Visualizing Time Series Data Using Web Map Service Time Dimension and SVG Interactive Animation

Timothée Becker

February, 2009

Visualizing Time Series Data Using Web Map Service Time Dimension and SVG Interactive Animation

by

Timothée Becker

Thesis submitted to the International Institute for Geo-information Science and Earth Observation in partial fulfilment of the requirements for the degree in Master of Science in *Geoinformatics*.

Degree Assessment Board

Thesis advisor	B.J. Köbben MSc and Ms Dr. C.A. Blok
Thesis examiners	Chair: Prof. Dr. M.J. Kraak (RTL), External examiner: Dr. G. Andrienko

INTERNATIONAL INSTITUTE FOR GEO-INFORMATION SCIENCE AND EARTH OBSERVATION

ENSCHEDE, THE NETHERLANDS

Disclaimer

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

Abstract

The constant increase in number and size of spatio-temporal (ST) datasets is challenging researchers to develop effective means for visually exploring and presenting the information they contain. Within the fields of Geovisualization and Exploratory Data Analysis, interactive animated maps have been pointed out as the only generic technique available to explore large ST datasets. They provide a complete view of the dynamic process under study, and often help to reveal its (subtle) spatio-temporal patterns. To become more broadly used, interactive vector animated maps must become less time-consuming to make and easier to disseminate. Internet technologies offer two big advantages in this direction: the possibility for interoperable distributed services and the ease of disseminating animations to specialists and wider audiences world-wide. With the global aim of improving vector animated mapping possibilities, the main objective of this research is to look into the possibility of combining two technologies that have never been combined before: animated and interactive vector graphics for the internet and distributed geo services. We adopt OGC's Web Map Service (WMS) framework for distributed services and Scalable Vector Graphics (SVG) to produce interactive animated maps that can be viewed in a Web browser. After determining what needs to be done to generate temporal animations from data stored according to WMS' recommendations, we designed and implemented an animated mapping prototype for exploring moving object dynamics with a case-study on iceberg dynamics. We base our design on reviews of animated mapping, moving object visualization and iceberg literature. As a result we present the TimeMapperWMS prototype which offers users worldwide the possibility to visualize the dynamics inherent to the thirty-three year Antarctic Iceberg time-series dataset. The most important visualization functionalities offered are a mechanism to select the temporal extent of the animations, temporal legends, a time-slider and speed control. Different animation types are destined to explore the changes in the iceberg population's distribution and the motion dynamics of individual as well as groups of icebergs.

Keywords

WMS time, Animated mapping, animated map, vector animation, time-series, spatio-temporal data, geovisualization, time-slider, moving object, interpolation, icebergs

Acknowledgements

I want to thank the following people back home: my parents, for their constant and unconditional support throughout my whole studies, David Chaille for the life-long lesson he taught me of how to organise my days of work – perhaps next time he can teach me to organise the months :), my brother Thomas for his support and patience, Ronny and Lena Moser and Daniel Leuenberger for their friendship and their so appreciated visit.

At ITC, I want to thank the following people: my supervisors, Barend Köbben and Dr. Connie Blok as well as Professor Menno Jan Kraak: Mr Köbben, for proposing such an exciting topic, for his guidance, his amazing patience and his hard work, Professor Kraak and Mrs Blok for their guidance, for always reminding me the academic requirements of an MSc research, for their valuable comments and for trying to make me stick to a schedule, Dita Anggraeni for all her help and her support, Clarisse Kagoyire for her support and her seemingly endless generosity, my classmate and friend Hoa Nguyen Thi Phuong for sharing the result of her work on iceberg data with me, Dr. Javier Morales for his help on the Unified Process, Dr. Rolf de By for his support with Latex and for designing the most challenging project of all teaching modules, the ITC and ITC Hotel staff for their amazing welcome and kindness, and finally, the entire ITC community, friends and cheerful faces, for making this experience unique, warm and delightful.

I also want to thank Erik Dahlström for his support as well as many members of the SVG community for their willingness to share their knowledge and code.

Contents

Abstract i

Acknowledgements iii

List of Tables ix

List of Figures xi

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Problem statement and objectives . . . 3
- 1.3 Research steps and research questions . . . 5
- 1.4 Methods . . . 7
- 1.5 Thesis structure . . . 8

2 Temporal animated maps: principles and recommendations 9
- 2.1 Introduction . . . 9
- 2.2 Geovisualization and Exploratory Data Analysis as background 10
- 2.3 Conceptual framework for space-time phenomena . . . 11
  - 2.3.1 Time as a dimension: the *Where – What – When* triad . . 11
  - 2.3.2 Exploratory task-levels focusing on time . . . 12
  - 2.3.3 Two types of time: linear and cyclic . . . 13
- 2.4 Describing dynamics : types of change, behaviors and patterns . 13
  - 2.4.1 Types of change . . . 13
  - 2.4.2 Spatio-temporal behaviors and patterns . . . 13
- 2.5 Principle of temporal animated mapping . . . 15
  - 2.5.1 Basic principle of animations . . . 15
  - 2.5.2 Basic characteristics: temporal scale and resolution . . . 15
  - 2.5.3 Basic needs for animated mapping . . . 16
- 2.6 Suitability and advantages of animated maps . . . 17
  - 2.6.1 Suitability of animated maps for spatial dynamics . . . 17
  - 2.6.2 Advantages of animated maps for spatial dynamics . . . 18
- 2.7 Perceptive and cognitive limitations of animated maps . . . 19
  - 2.7.1 Cognitive Load Theory for exploratory mapping . . . 21
- 2.8 Design requirements and recommendations for animated maps 21
  - 2.8.1 Rules transferred from static on-screen mapping . . . 22
  - 2.8.2 Temporal legends . . . 23

2.8.3 Interactive control of display time . . . 25
2.8.4 Solutions to the split-attention effect . . . 26
2.8.5 Zooming and temporal focusing . . . 27
2.8.6 Controlling the temporal scale/speed . . . 28
2.8.7 Emphasizing change: smoothness, saccades and dynamic variables . . . 28
2.8.8 Complementarity of small-multiples and alternative views 29
2.9 Conclusion . . . 30

3 Technical choices: WMS, SVG and RIMapperWMS 31
3.1 Introduction . . . 31
3.2 Why OGC's WMS as distributed GIService? . . . 32
3.2.1 Web Map Service in a nutshell . . . 32
3.2.2 WMS support and format for the Time dimension . . . 34
3.3 Why choose SVG as a vector graphics format? . . . 36
3.4 Interactivity and animation in SVG . . . 39
3.4.1 Short introduction to SVG's syntax . . . 39
3.4.2 Scripting interactivity and animations . . . 40
3.4.3 SVG and SMIL animation . . . 41
3.5 Compatibility between the WMS and SVG specifications . . . 44
3.6 Choosing a platform to extend: RIMapperWMS . . . 45
3.7 Conclusion . . . 46

4 Moving objects visualization and iceberg use-case analysis 49
4.1 Towards a visualization environment for moving objects . . . 49
4.2 Conceptual framework for moving object data . . . 50
4.2.1 Conceptualizing object dynamics . . . 50
4.2.2 Pattern types for moving object phenomena . . . 52
4.2.3 Visual techniques for movement, and data reduction . . 54
4.3 Case-study: Antarctic Iceberg movement visualization . . . 54
4.3.1 Iceberg formation and fields of application . . . 54
4.3.2 The NIC Antarctic Iceberg dataset . . . 55
4.3.3 Iceberg-visualization tasks using animation . . . 58
4.4 Analysis of Iceberg visualization use-case . . . 62
4.5 Summary . . . 65

5 Animated mapping visualization system design 67
5.1 Introduction . . . 67
5.2 Visualization options offered to the user . . . 68
5.3 Description of visualization functionalities . . . 69
5.3.1 Generic functionalities for animated mapping . . . 70
5.3.2 Moving object specific functionalities . . . 72
5.3.3 Iceberg specific visualization components . . . 74
5.4 How should the data be stored, converted and retrieved? . . . 75
5.4.1 Storage of spatio-temporal data . . . 75
5.4.2 Steps to convert the time component of the data . . . 76
5.4.3 The need for a temporal intersect . . . 78

5.4.4 Integration of additional georeferenced products 80
5.5 High level system structure 80
5.5.1 Conceptual representation of the system 81
5.5.2 RIMapperWMS' present structure 81
5.5.3 Extending the structure: the birth of TimeMapperWMS 83
5.6 The actual GetCapabilities and GetMaps: requests and responses 84
5.7 Summary 86

6 TimeMapperWMS prototype implementation 87
6.1 Introduction 87
6.2 Populating the database 87
6.3 Client-side visualization functionalities 88
6.3.1 Building animation behavior 88
6.3.2 Building temporal legends 91
6.3.3 Controlling time: time-slider and other functionalities . 92
6.3.4 Providing visualization options to the user 93
6.4 Serializing: generating interactive animated maps 94
6.4.1 Querying the database 94
6.4.2 Building animations from the data and storing time-stamp variables 94
6.5 Setting the temporal scale: a functionality that binds all 96
6.5.1 Setting the value of the temporal scale 96
6.5.2 Setting the start-times and durations of the animations 97
6.6 Integrating WMS map backgrounds and additional data 98
6.7 State of the implementation 98

7 Results, testing and evaluation 99
7.1 Introduction 99
7.2 System testing: functionality and limitations 100
7.2.1 How well do the prototype's functionalities work? 100
7.2.2 Solutions to improve responsiveness 100
7.3 Evaluating the generic interactive functionalities 101
7.4 Evaluating animation types for moving-object visualization . . 105
7.5 Using the system for other datasets and extending its capabilities 106

8 Conclusion and recommendations 107
8.1 Combining distributed services and vector animation 107
8.2 Designing a visualization environment for iceberg dynamics . . 108
8.3 Implementation of the TimeMapper prototype 109
8.4 Results and system evaluation 109
8.5 Recommendations 110

A Scenario to introduce the three basic needs in animated mapping 113

B Explanations on Köbben's SVG scripted animations 115

C Example of periodicity in an MCB visualization 117

D SVG and JavaScript code **119**
D.1 Introduction 119
D.2 Interpolated iceberg tracks 119
D.3 Animating attribute changes 120
D.4 Linear temporal legend 121
D.5 Pie-portion cyclic temporal legend 121

Bibliography **123**

List of Tables

2.1 Tasks focusing on *when* 12

3.1 Symbols for time periods 35
3.2 GetMap request parameters 36

List of Figures

1.1 Making use of generically applicable principles and technologies 4
1.2 Methodology of the research 7
1.3 Five core workflows of the Unified Process 8

2.1 The Where – When – What triad 11
2.2 Three types of temporal legends. 23

3.1 Three-tier basic WMS setup 33
3.2 Simple SVG graphics example 41
3.3 Example of the power of SMIL for morphed animations (from [55]) 42

4.1 Individual movement behavior and Momentary collective behavior 51
4.2 Iceberg identification from satellite images 56
4.3 Map of Antarctica 56
4.4 Use-cases for iceberg visualization 62

5.1 Projected user interface for animated mapping environment . . 69
5.2 Four designs of cyclic temporal legends 70
5.3 OGC-compliant database schema 77
5.4 Time conversion steps 77
5.5 The need for a temporal intersect 78
5.6 Applying a temporal intersect to spatio-temporal data 79
5.7 Three-tier architecture: Web-client-map server-database 80
5.8 Extending the structure of RIMapper 82

6.1 Serialization of animations, storing variables and setting animation speed 95
6.2 Offering the user flexibility in temporal scale settings 97

7.1 The TimeMapperWMS prototype 103
7.2 Motion dynamics visualization 103
7.3 The ‘small pie’ cyclic temporal legend 104
7.4 Moving cluster pattern 104

C.1 MCB-based visualization of bird migration speeds 117

Chapter 1

Introduction

1.1 Motivation

The need felt by cartographers to represent time along with their spatial depictions is hundreds or even thousands of years old [28]. Until the advent of more modern techniques, the use of graphical symbolizations was used to represent movement or change. We can for example state the use of simple symbols such as arrows to represent movement or of more complex depictions such as Minard's famous map. However, the on-going increase in capacity to record spatio-temporal information provided by modern technology (especially GPS and satellite-born remote sensing) has been propelling the development of new visualization techniques and tools.

Among other techniques, such as small-multiples, maps embedded with time-plots of attribute values, and the space-time cube, map animation is one of the techniques used in Geovisualization to explore and present spatio-temporal phenomena. It is believed to have a series of advantages among which we will present the following subset: Because the graphical depictions can be made to look like real-world states and because display-time is used to represent the passage or real-world time, animated maps (AM) offer an intuitive, easily understandable representation of real-world phenomena [13]. According to Andrienko and Andrienko [4], it is the only generically usable technique capable of offering overviews of spatio-temporal datasets with wide ranges of temporal values.[1] Animated maps in effect, especially when supplied with temporal legends, enable one to view information in its temporal context. The most frequently stated advantage of animated maps is probably their ability to help detect (possibly subtle) spatio-temporal patterns [30]. These advantages and a few more result in a wide range of application domains for animated maps. Harrower and Fabrikant [30] state that"the use of animated maps spans the spectrum of disseminating spatial knowledge to a wide audience (. . .) to data exploration" by experts using "highly interactive (. . .) tools."

[1] Some of the other techniques can be used for certain data types but none except animation can be used for all.

Despite this wide range of potential applications, a fairly small number of animated maps are presently being disseminated. The recent development of kml animations for Google Earth did however make this number increase. Harrower [28] predicts that the place where animated maps could be disseminated in large numbers is the internet and that if such an increase happens, it will follow the advent of "on-demand" animated mapping systems.

Animated mapping facilities are better developed for raster georeferenced images than they are for vector graphics and data.[2] For advanced forms of vector animation (i.e., involving some interpolation), programming or scripting is necessary: the "appearance of animated objects (...) can be rendered "on the fly" by the computer, which decodes and visually renders these vector-based animations" [28]. In the present thesis, we will be looking into vector graphics animations for the internet.

While some vector animation platforms exist, like ArcGIS or Tempest [19][3], they are characterized by three big deficiencies: the impossibility to publish the animations on the internet, a low level of interactivity[4] and, the absence of a feature to produce interpolated animations. Because of the latter, it depends on the data, for the animations to appear smooth. Because animated mapping platforms do not produce internet-publishable outputs, the most broadly adopted platform to build vector animated Web maps is Adobe Flash.[5] This is often done in a graphics-only setup which is very laborsome.

For animated mapping like for any vector GIS task, it is obviously an advantage of having the graphics hooked onto a database. The vector graphics can be animated using spatial and object attributes along with a temporal attribute. Such a setup offers to animated mapping the generic advantages of database management systems: flexibility, reusability, data integrity, data consistency, enhanced security, access for multiple users in multiple locations, possibly at the same time.

Besides broad dissemination possibilities, animated mapping can benefit from internet services by being integrated into distributed services. Interoperability and distributed services is a clear trend that today's Geographic Information Science (GIS) is taking [24]. The advantages of distributed services for animated mapping platforms partly descend from the qualities of database management systems (DBMS). A second set of more specific advantages result from the interoperability of standardized Geo Web Services.

Standards for interoperability proposed by distributed frameworks such as the Open Geospatial Consortium (OGC) offer advantages for data sharing, for com-

[2]The reason is that a time-series set of raster images covering the same region can easily be played in a rapid sequence. However, because of the heaviness of raster images, of the limited power of computers and bands of networks, good quality and long time-series of raster animations are still difficult to disseminate on the internet and are devoid of high level interactive control even when running on stand-alone computers.

[3]Google Earth is not an animated mapping platform. The animations need to be scripted.

[4]This is even truer about raster animations.

[5]See [27] as an example.

bining software components and for overlaying graphical outputs from different sources. As a result, with a minimum need for adapting data products and software components to each other, distributed services offer the possibility to overlay image products coming from multiple datastores and processed by multiple map server software (potentially all physically in different locations). This series of advantages cause distributed services to be "widely accepted in governmental agencies and educational institutions" as well as by "geospatial data producers, users, GIS vendors, and GIS professionals" [64].

As already said, animated maps are used to explore and present real-world processes. In many cases, geovisualization specialist are interested in explaining the dynamics present in these processes by relating them to other phenomena. Interesting geovisual work could result from the overlay of multiple dynamic phenomena or of a dynamic phenomenon with static map layers.

Many authors have stated the need for interactivity in animated mapping (for reviews and further references, see [30, 72]). This need for interactivity is rooted in perceptive and cognitive limitations of human users: due to the fugitive characteristics of the information displayed, animated map users tend to have trouble perceiving changes, and remembering them for long enough to make sense of them. Because of these perceptive and cognitive limitations, animated map theorists and designers recommend a series of design rules. Along with simplifying graphical depictions, providing the user with control over the amount of information displayed is the most often recommended solution [30, 72]. Through interactivity, the user must be able to control the animation in a way that enables him to grasp the changes occurring and remember them. For various Geovisualization purposes, interactive time-sliders are increasingly often being provided [19, 25]. We will see that they offer a very powerful and flexible way of browsing the data.

1.2 Problem statement and objectives

In summary, we have established that animated mapping could benefit from the binding of vector graphics to a database and from being integrated within a distributed services framework. In addition, the most promising way to disseminate animated maps is via the internet. The deficiencies of existing animated mapping platforms are important. Apart from the fact that none functions in a distributed services architecture, these platforms are characterized by (a) the impossibility to publish vector animations on the internet, and either one or the other of the following: (b) a low level of interactivity and (c) the absence of mechanisms capable of rendering interpolated spatio-temporal states.

With the global aim of improving animated mapping possibilities, the main objective of this research is to look into possibilities, both from theoretical and practical perspectives, of combining two technologies that have never been combined before: animatable and interactively controllable vector graphics formats for the internet and distributed geo services. At a theoretical level, we will

study the specifications of existing technologies and at a practical level, we will attempt to develop a prototype animated mapping system using these technologies. We hope to adopt solutions that could serve both exploratory and Web presentation purposes. At a functionality design level, we intend to follow recommendations found in literature on animated mapping to instruct the design of state-of-the-art animation and interactive functions.

With the limited time at disposal, we cannot expect to develop a complete animated mapping application. Thus, with the intention of developing solely a prototype, we need to restrict the set of functionalities to provide. The completeness of an animated mapping platform can be considered from three different perspectives all related to how the platform satisfies user requirements. (1) It can produce different types of animations: stepwise animations, interpolated animations, brushed animations, lagged animations, etc...(2) It can be used to visualize different types of real world phenomena (e.g., moving object data, political census data, health statistics ...) and different types of data (point data, line, polygon). (3) It can be used for different types of tasks going from exploration to presentation to the general public.

We will restrict ourselves in the design and implementation phases by making selections from all three of these perspectives. Firstly, we decided to develop an environment for moving object visualization. The second restriction happened as a result of a very interesting case-study dataset being offered to us, i.e., the NIC's Antarctic Icebergs dataset [49]. This dataset seemed more appropriate,

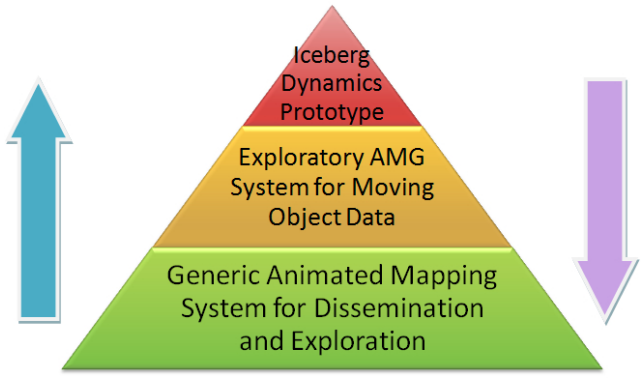


Figure 1.1: The principles and technologies that will be used to develop the prototype could serve to develop a generic animated mapping platform.

in a first stage, for phenomenon exploration than for presentation. Thirdly, by studying iceberg dynamics in literature, we intend to come up with what we will judge to be the most useful animation and interactive control functionalities and to propose an iceberg dynamics visualization prototype environment. This progressive focusing towards a visualization environment of a size that we can hope to design and implement is illustrated in figure 1.1 and particularly represented by the flow direction of the blue arrow. The design of the animation and interactivity features that we intend to develop will be informed by a review of animated mapping literature and visualization concepts for moving

objects data as well as by our analysis of the use-case on iceberg movement. The purple arrow shows that the prototype developed can be considered, if we may say so, as the tip of the iceberg of a generic animated mapping system that could be developed using the same principles, the same architecture and the same set of technologies.

1.3 Research steps and research questions

To attain the aforementioned objectives, knowledge needs to be gained, choices need to be made and problems need to be solved on a variety of topics. The following research questions, grouped by theme, will guide us through our research.

On animated mapping

After setting a theoretical background for animated mapping within the broader field of Geovisualization, we intend to discover the principles, applications and requirements for animated vector mapping. How do animated maps work? What visualization tasks are they useful for or what are their advantages? On the other hand, what are their limitations? Fond of their knowledge and experience, what do animated mapping theorist and designers preconize or recommend for developing effective geovisualizations using animation?

On technical choices

OGC is the official body for standardization and interoperability of distributed geo Web services and their specification for visual products is Web Map Service (WMS). We intend first to verify that OGC's WMS is indeed the best framework to adopt for our purpose. Preliminary research showed that the WMS implementation specification contains a recommendation for the storage and sharing of temporal components. We will try and determine from a theoretical point of view, whether the WMS specification for storage and sharing of time is suitable for our purpose.

In our research for a vector graphics format that could be used to animate and interactively control the temporal dimension of animations as well as be published via the internet, Adobe Flash and the W3C's Scalable Vector Graphic (SVG) were identified as the main competitors. We will show why we choose SVG rather than Flash. What are the advantages and limitations of SVG in general? What is its structure? How well documented is it? How well can it be bound to a database? Further, what are its specific advantages for animated mapping? How does SVG handle the temporal aspect of its animations and how flexible is to develop different types of animations?

Once these questions on the main technology components have been answered, we will try and determine how well the two technologies can be combined from a theoretical or specification point of view. Can we expect them to be compatible? In particular, does the way in which WMS recommends to store time and the way SVG designs usage of time seem to fit together well?

As we cannot expect to develop a complete animated mapping prototype for moving objects following the WMS framework, we hope to find an existing implementation that we could extend. What are the already existing WMS packages that we could extend to yield SVG interactive animated maps? Further, if such a package exists, what needs to be done to extend it?

On moving object visualization and iceberg dynamics

In our attempt to develop a visualization environment for moving objects, we need to develop an approach to understand the phenomena at hand. How can movement of objects be conceptualized? What kinds of patterns can we expect to find in such movement data? Which types of behaviors can be observed using animated maps and which of the pattern types that can be found in movement data can they help to detect?

After presenting general knowledge on icebergs and our specific dataset, we will attempt to answer specific questions about iceberg dynamics exploration. What iceberg visualization tasks can we hope to successfully apply animation to? Is it suited to assess the evolution of the distribution of icebergs through time? Can it be used to visualize the motion dynamics of icebergs? Finally, is it suited to visualize iceberg dynamics in relation to other natural phenomena?

On conversions between time formats

The compatibility of the ways in which time is stored in WMS and used in SVG should not only be assessed in theory. Our prototype will serve as a practical test-case for assessing the effective compatibility of the two time formats. What are the steps needed to convert the time-stamps from real-world time to display time attribute values?

On designing a visualization environment

Having increased our knowledge on animated mapping, moving objects visualization theory and iceberg visualization tasks, we enter the phase of system design. Based on a vision of the final system's output and behavior, we will try to determine the following: What should be the architecture of the system? What should be its components and how should these components behave and interact with each other?

We mentioned already three important features that advanced animated mapping applications should in our opinion include: spatio-temporal interpolated animations, temporal legends and interactive control of the temporal dimension. Within the set of features that we consider developing, these three will be given a special place and attention.

1.4 Methods

To answer the research questions posed above, a methodology was developed. Figure 1.2 offers a schema of these methods. Our first task will be to review literature on spatio-temporal phenomena visualization and animated mapping. Next, we will study the candidate technologies that we intend to combine and

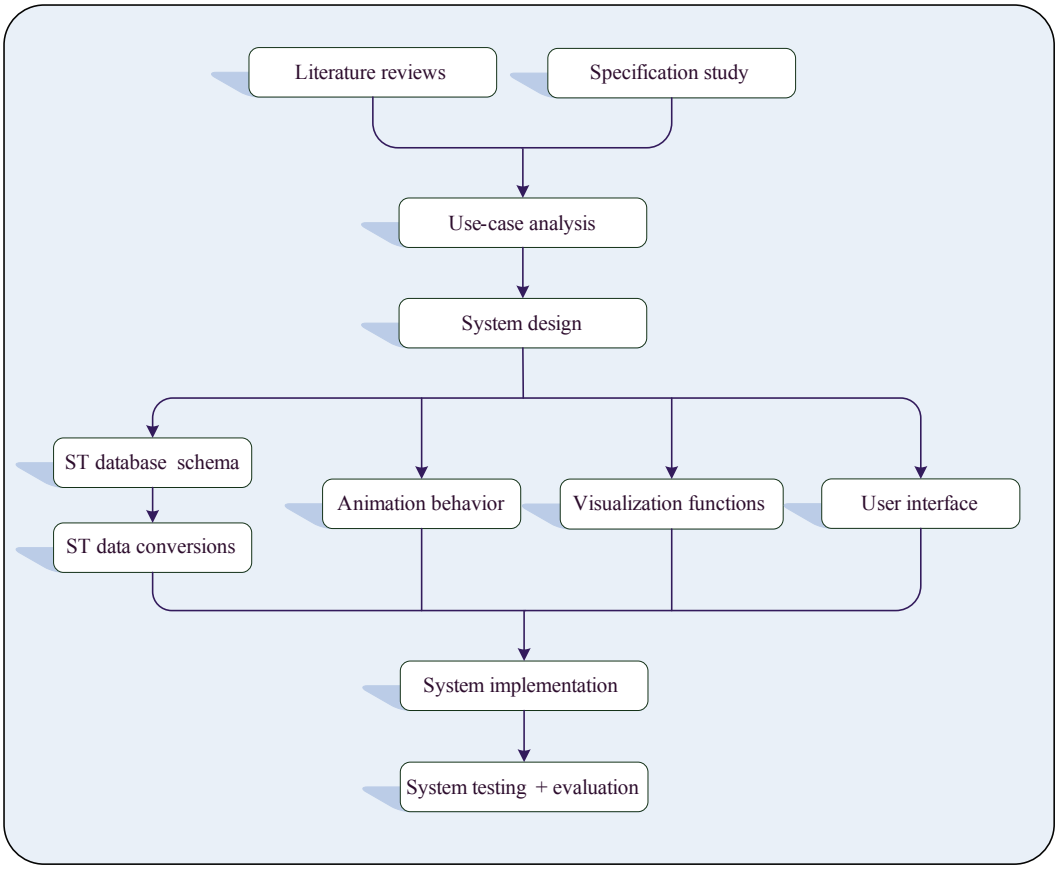


Figure 1.2: Methodology of the research

review the official documentation provided for them. Narrowing down on the type of dynamic phenomena that we intend to visualize, we will review concepts to understand and visualize moving object data. Our case-study or use-case on iceberg movement visualization will then be studied and analyzed. Using the result of this analysis, we will design an animated mapping prototype for ice-

berg dynamics visualization. The design outline will inform the implementation of the system which will be pursued using database technology and appropriate programming and scripting languages.

The design of the system we intend to propose will be based on all that will have been established in the first three chapters. For our system development, we follow the steps described in the method called the Unified Process [8]. The five

Figure 1.3: Five core workflows of the Unified Process (from (8))

steps "requirements, analysis, design, implementation and testing" are shown in figure 1.3. Our process will be initiated with the presentation of a use-case involving different field specialists interested in iceberg visualization. The use-case requirements will be analyzed to further instruct the design of the system.

1.5 Thesis structure

Including this introduction and our conclusions, this thesis is composed of eight chapters.

In the next chapter, which ultimately aims at pointing out principles and recommendations for temporal animated mapping, we start by introducing our topic as part of the field of Geovisualization in general and of space-time phenomena exploration and presentation in particular. In Chapter 3, we justify our choices of WMS and SVG as technical solutions for our implementation, we determine how well the two technologies can be combined together and finally choose an existing WMS implementation generating SVG output that we will attempt to extend.

A second phase of our research begins with Chapter 4, its study of a framework for moving object visualization and the presentation and analysis we make of our iceberg use-case. Chapter 5 outlines the architecture centric design that we draw of our prototype and explains each component in detail. The following chapter logically is the implementation, in which we report on the practical work done. In Chapter 7, we start by presenting the visualization results we got on our case-study and report on the testing of the prototype's functioning. Further on, we evaluate the system in terms of how well its components could be used for a more generic animated mapping platform. Finally, we will give recommendations for further developments and conclude on our work.

Chapter 2

Temporal animated maps: principles and recommendations

2.1 Introduction

The goal of this chapter is double: give a theoretical background and framework to our study and point to a set of design recommendations for developing an animated mapping environment. This study is limited to temporal animated mapping and does not cover the use of animation for other purposes (e.g., fly-bys, progressive building of the map layers, etc.).

The constitutive parts are the following: We start by setting a broad background for animated mapping within the wide field of Geovisualization. Then, we adopt theoretical frameworks for the study of spatio-temporal phenomena. These will help us to better understand the temporal dimension of animated mapping and to establish a relationship between real-world behaviors and the patterns we can detect in them. We then genuinely enter the field of animated mapping and present its principles, its advantages and its limitations. We end with a comprehensive list of design rules and recommendations for temporal animated mapping.

Since the environment we intend to develop is based on vector graphics and not raster, our recommendations are slightly inflected towards vector animation. However, most of the principles, advantages and limitations of raster and vector animated maps are the same. What may differ slightly is the types of controls that we provide the user with.

A fair part of the literature we will use here is written with a focus more on knowledge sharing than on data exploration. Our interest regarding knowledge sharing is to contribute to the development of an animated mapping platform capable of generic output. Thus, we argue that its functionalities should be basic. We will focus our approach slightly towards exploration because of the

orientation of the prototype we intend to develop. However, a generic knowledge sharing application and a generic exploratory environment should have similar basic features.

2.2 Geovisualization and Exploratory Data Analysis as background

As Harrower and Fabrikant [30] say, "many of today's significant challenges, such as resource management and environmental monitoring, depend upon capturing, analyzing, and representing dynamic geographic processes." Animated mapping is a technique which is increasingly often used by practitioners of a variety of fields. We can consider that in the center of all these fields lies Geographic Visualization, also termed Geovisualization.

Geographic Visualization is defined by MacEachren and others [44] as "a form of information visualization in which principles from cartography, geographic information systems, Exploratory Data Analysis, and information visualization more generally are integrated in the development and assessment of visual methods that facilitate the exploration, analysis, synthesis, and presentation of georeferenced information."

In their introductory chapter "The power of geographic visualization", Dodge and others [15] explain that this discipline "works by providing graphical ideation to render a place, a phenomenon or a process visible, enabling the most powerful human information-processing abilities – those of spatial cognition associated with the eye-brain vision system – to be directly brought to bear." Andrienko and Andrienko [4] also insist on the predominant importance of visualization. They assert that "in order to be able to think about data, the mind needs to perceive the data." In that sense, they consider that "one picture is worth much more than a collection of numbers." Dodge and others [15] further insist that visualization is a "cognitive process of learning through the active engagement with graphical signs." The statements we chose underline the fact that a geovisualization researcher analyses a representation of reality instead of reality itself.

MacEachren and others' definition also shows that geovisualization is not only a discipline involved with exploration and analysis of georeferenced information but also with its "presentation". However, it should be clear that geovisualization techniques are applied to "reveal novel insights that are not apparent with other methods of presentation [15]".

While being connected to almost all the disciplines stated in the definition we gave, our research is particularly connected to Exploratory Data Analysis. We will see that the technique of temporal animated mapping is often used to explore large multidimensional datasets about which little is known. It is recommended [4] for revealing the structure and behavior patterns inherent to spatio-temporal phenomena.

2.3 Conceptual framework for space-time phenomena

More than a decade ago, Edsall and Peuquet [19] made the statement that "the incorporation of time into GIS has introduced challenges in all realms of GIS research." Several of these challenges have been satisfactorily solved today, like, for example, the storage of spatio-temporal data within databases. Others, like the visualization of such data, are presently the object of intense research in geoinformatics and this present work aims at being a part of this. To represent time and phenomena unrolling in time coherently, it is necessary to refer to a well defined conceptual model of time.

2.3.1 Time as a dimension: the *Where – What – When* triad

Peuquet's [67] developed a "conceptual framework for the representation of temporal dynamics in geographic information systems" that can be considered an important building block of today's research for the integration of time into geovisualization. The key element of the framework is the where – when – what triad (as shown in figure 2.1).

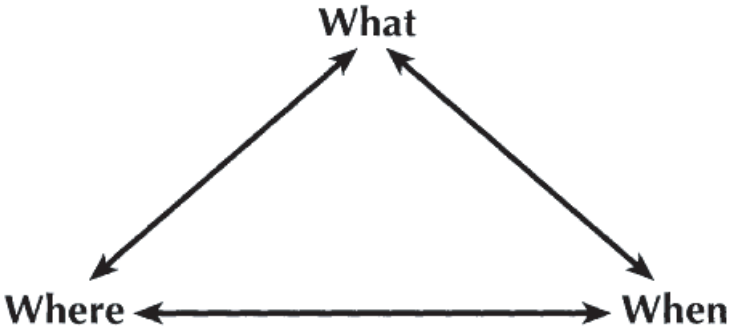


Figure 2.1: Where – When – What triad (reproduced from: (67))

The where – when – what triad contributed to the development of a conceptual schema considering time not merely as a data attribute but as a *dimension* in itself. Geographic phenomena can now be approached from three different angles enabling three "basic kinds of questions" [67]:

1. when + where → what: Describe the object or set of objects (what) that are present at a given location or set of locations (where) at a given time or set of times (when).
2. when + what → where: Describe the location or set of locations (where) occupied by a given object or set of objects (what) at a given time or set of times (when).

3. where + what → when: Describe the times or set of times (when) that a given object or set of objects (what) occupied a given location or set of locations (where). [67]

2.3.2 Exploratory task-levels focusing on time

In their systematic approach in "exploratory analysis of spatial and temporal data," Andrienko and Andrienko [4] are inspired by Koussoulakou and Kraak [37] to adopt a framework similar to the one Bertin developed on exploratory tasks. In the same way Bertin distinguishes between different spatial "reading levels" [11], other authors [37, 7] distinguish between such reading levels for the other data dimensions. Andrienko and Andrienko [4], for example, distinguish between elementary tasks and synoptic tasks (earlier termed general tasks by Andrienko and others [7]. Elementary tasks "refer to individual elements." Synoptic tasks "involve the whole reference set or its subsets." For the latter type of tasks, they give the following example: "describe the variation of the proportions of children over the whole country."

Andrienko and others [7] adopt Peuquet's basic kinds of questions and apply to them the distinction of reading levels. The number of combinations becomes large as for each of Peuquet's basic kind of question, the elements present on both sides of the "→" are divided into elementary and synoptic (or general). Thus, all together, twelve different combined approaches exist which shows well how complicated things can get when the temporal dimension is introduced to geographical analysis. With the objective of simplifying this approach and because they are interested primarily in the temporal aspect of the phenomena, Andrienko and others decide to focus on the basic kind of question which are centered on time. They come up with four exploratory scenarios and definitions that we have entered into table 2.1.

	elementary *what* + *where*	general *what* + *where*
elementary *when*	Describe characteristics of this object (location) at the given time moment.	Describe the situation at the given time moment.
general *when*	Describe the dynamics of characteristics of this object (at this location) over time.	Describe the evolution of the overall situation over time.

Table 2.1: Elementary and general levels of exploration for tasks focusing on *when* (adapted from: [7])

2.3.3 Two types of time: linear and cyclic

Geographers who discussed the notions of time generally agree that time can be considered in two different ways: as a linear continuum or as repetitive cycles. In linear time, the time reference is absolute and measurable by clocks. In cyclic time, absolute time remains present in the way time is measured but the main temporal reference is one cycle. This way of thinking about time is fruitful to study repetitive patterns inborn in natural and human processes. Nature bears cycles such as days, years and moon cycles. Societies further built other cyclic temporal structures for organizing their activities (e.g., weeks, months, hours, minutes, etc). Let's look at transport as an example of human activity to show the advantage of the cyclic time perspective. While studying transport patterns, a researcher will get more insight about spatial behaviors if he thinks of them as projected onto a daily temporal reference (one day = one cycle) than if he thinks about them as simply projected on a uniform and unstructured timeline. For transport, weekly, monthly and yearly cycles could be equally relevant depending on the specific behavior that one tries to analyze.

2.4 Describing dynamics : types of change, behaviors and patterns

In the present section, we will present two ways of considering dynamic phenomena. The first way originates from the idea that any phenomena happening in time can be viewed as one of three types of change. The second way, centered on Andrienko and Andrienko's [4] notions of behavior and patterns, attempts to describe the tendencies present in the dynamics of real-world phenomena.

2.4.1 Types of change

Real-world processes can all be viewed as the product of some form of change. Andrienko and others [7] made use of Blok's [12] work on change and propose three categories of changes:

1. Existential changes, i.e. appearances and disappearances
2. Changes of spatial properties: location, shape or/and size, orientation, altitude, height, gradient and volume.
3. Changes of thematic properties expressed through values of attributes

2.4.2 Spatio-temporal behaviors and patterns

Andrienko and Andrienko [4] introduce the conceptual notion of behavior to observe the characteristics of spatio-temporal datasets. They define behavior as "the set of all characteristics corresponding to a given reference (sub)set,

considered in its entirety and its particular organization with respect to the reference (sub)set." Since behavior is related to "all characteristics", this notion only applies to the synoptic or general level and not to the elementary one. The authors further clarify that "behavior" is "a generalization of such notions as distributions, variations, and trends."

We find slightly misleading that Andrienko and Andrienko include distributions in their behavior category. The common use of the word behavior is related to action or at least to something happening in time. To the contrary a distribution , to something describing a single state in time. We see further on that this inclusion of static states in their definition is intentional as they provide us with the following example: "find spatial clusters of districts with a high proportion of children." They further specify that the behavior under study is "spatial cluster of high values."

While we consider the notion of behavior to be useful, we want to reject the part of this concept that is not related to time. We understand Andrienko and Andrienko's idea of having a general term both for static and dynamic phenomena. We also understand that many static states, such as distributions, are the result of a process and actually of a behavior or a set of behaviors. However, our intuitive understanding of the term behavior, which is in harmony with the definitions provided by the Oxford English Dictionary [1][1], motivates us to examine the value of the generalization made by Andrienko and Andrienko. Because in visualization and in particular in animated mapping, dynamism is the result of many static states observed in a sequence, we consider it is useful to keep dynamic descriptive terms such as "trends and variations" from static descriptive terms such as distributions. For these reasons, we decide, in our framework, to restrict the use of 'behavior' to describe dynamic processes.

Detecting spatio-temporal patterns is the most widely stated advantage of animated mapping. Andrienko and Andrienko [4] define pattern in general as "a construct that reflects essential features of a behavior in a parsimonious manner." The relation posed by the authors between behavior and pattern is a very useful one. While a behavior relates to the "set of all characteristics" (centered on the phenomenon), a pattern is a summary mental construct (centered on the understanding of the observer) that can be expressed in language. In this study, we are primarily interested in spatio-temporal patterns. As an example of such a pattern, we could present a case of household income. An analyst might observe that the centralized distribution of high incomes evolves into a pattern with many smaller isolated areas of high income. We can see here that spatio-temporal patterns can often be described as the evolution of two or more spatial patterns through time.

In many cases, the goal of exploratory data analysis is to find a set of patterns that represents, or approximates, the behavior [4] under study. Patterns can be explicitly described by a researcher or can be present as a mental schema

[1]None of the six main definitions given describes a static phenomenon. The fifth definition defines behavior as "the manner in which a thing acts under specified conditions or circumstances, or in relation to other things." The term "act" shows that behavior happens through time.

in his mind. While observing a temporal animation, a researcher has a set of patterns in his mind. A particular pattern can be triggered by the behavior of the phenomenon he is observing. For instance, in wild animal data exploration, a researcher might notice that many individual animals that are first dispersed in space converge to a region of shade or water at the same time. *Convergence* is the pattern in the research which is triggered. After getting acquainted with all the behavioral aspects of the phenomenon, the researcher can attempt to summarize the behavior with a set of patterns.

2.5 Principle of temporal animated mapping

In the previous sections, we have set up a conceptual background for the visualization of spatio-temporal phenomena without looking into any specific technique for the actual visualization. In the present section, we start by reviewing the basic principle of animation and the basic characteristics of animated maps (temporal scale and temporal resolution) and continue with three basic requirements for animated mapping (time to get acquainted, temporal legends and interactivity).

2.5.1 Basic principle of animations

The basic principle of temporal animated maps is simple. Harrower and Fabrikant [30] define animations as "sequences of static graphic depictions (frames), the graphic content of which, when shown in rapid succession (...), begins moving in a fluid motion." One single frame, that we might think of as a paused animation shows a graphical representation of the real-world much like any static map. To represent change, the technique makes use of display time to represent the passage of real-world time. Real-world events present in the data are displayed in the same sequence as they happened and real-world dynamics are observable in the map-display. Temporal animated maps therefore show the unfold of real-world processes in a very intuitive way, which inspires Blok to say that they "mimic real-world dynamics" [13].

This definition is valid both to raster and vector animations. However, while raster animations are simple to achieve (show a series of raster images of the same area at a rapid pace), vector graphics animations needs programming. Harrower [28] claims the technique consists to "numerically define the appearance of animated objects (e.g., size, direction, and velocity) so that they can be rendered "on the fly" by the computer, which decodes and visually renders these vector-based animations in real time."

2.5.2 Basic characteristics: temporal scale and resolution

Just as the spatial scale of a map is the ratio between map distances and real-world distances, the temporal scale of an animation depicting a real-world phe-

nomenon is the ratio between display-time and real-world time [30]. Harrower and Fabrikant [30] give the following example: “five years of data shown in a ten second animation would have a temporal scale of 1:157 million.” We want to make it clear to the reader that we are not talking here about the rate at which frames are played. An animation can have a very *high frame-rate speed* at the same time as a high temporal scale (high temporal scale means slow rate of change in the animation).

Some authors refer to the speed of an animation instead of its temporal scale. The notion of speed is more obvious than that of temporal scale. However, the speed of an animation cannot be objectively defined and a user is bound to describe his actions with words like faster or slower. Because temporal scale is more objective than speed and because *faster/slower* are more intuitive, we suggest that the notions of temporal scale and that of speed (and its related terms: faster/slower) should remain in use.

Most animated maps keep a constant temporal scale throughout the animation although, as we will see when we talk about event-based animated maps, the interactivity provided to the user enables him to view different parts of the animation at a slower or a faster pace, thus making the temporal scale vary.

The temporal resolution of an animated map, also called temporal granularity can be defined as “the finest temporal unit resolvable. [30]” For animations including an interpolation mechanism, we need to make a distinction between the temporal resolution present in the dataset and the temporal resolution of the actual animations. The temporal resolution of vector objects in a database may or may not be regular. Concerning vector graphics animations, if there is no interpolation, the temporal resolution will correspond to the temporal resolution present in the data. Alternatively, if there is an interpolation mechanism, the temporal resolution may then be thought of as infinitely small.

2.5.3 Basic needs for animated mapping

To introduce what we consider to be the three basic needs in the design of animated maps and animated mapping tasks, we developed a scenario showing the cognitive process triggered by a simple example of a static map reading task. This scenario can be found in appendix A. Here we will only present its main conclusions.

An analyst has a set of basic needs in the first phases of exploring a spatio-temporal dataset using the technique of map animation. Basically, to make sense of what he is seeing, he needs time and ways to get familiar with all three dimensions of the map (where, what, when). To satisfy these needs, three basic requirements for the design of animated maps and animated mapping tasks have been pointed out in literature:

1. The need for time to get acquainted with all three dimensions. Because of continuous change, the amount of information to get hold of is much greater than in static maps.
2. The need for temporal legends to build an appropriate temporal schema. Kraak and others [39] state that "for users to understand a temporal animation, (...they must ...) apply an appropriate temporal schema that allows them to interpret meaning inherent in the sequence and pacing of the animation. They conclude that the "animated maps should be accompanied by a legend that prompts an appropriate schema," i.e., one or several temporal legends.
3. The need for interactivity to give himself time to freely explore the attribute and temporal dimensions of the data without overloading his cognition. Specifically for exploratory purposes, Andrienko and others [6] advance that ""pure" animation is insufficient for supporting exploratory analysis of time-referred data. An analyst should have at her/his disposal powerful and convenient facilities to control display time within the presentation."

We will study these features more in detail, along with other requirements or recommendations, in section 2.8. Harrower and Fabrikant [30], as well as Slocum and others [72], report that authors agree on the needs for interactivity and for temporal legends. Therefore, in the forthcoming sections, we will never make distinctions between animated maps with or without interactive controls and with or without temporal legends. We assume that in all cases where they are helpful, they are provided.

2.6 Suitability and advantages of animated maps

2.6.1 Suitability of animated maps for spatial dynamics

Review papers as well as specialized papers [66, 13, 30, 72] affirm that animated mapping is a useful technique for knowledge sharing despite the skepticism and critics of certain authors. Furthermore, their use for exploratory purposes is undisputed. Andrienko and Andrienko [4] assert that to fully and visually explore a massive spatio-temporal dataset whose temporal dimension contains a wide range and large number of values, "animation may be the only reasonable choice." Except for representing the trajectories of moving objects, where the space-time-cube can be an effective technique, the main technique to which animated mapping is compared to is that of small-multiples. Specialized authors generally agree that "map animations and small-multiples are best used for different tasks. The former being more useful for inspecting the overall trend in a time series dataset, the latter for comparisons of various stages at different time steps" [30]. In the following sections, we show what advantages

animated maps possess and eventually show how the approach compares with small-multiples in specific.[2]

2.6.2 Advantages of animated maps for spatial dynamics

The principle of animated mapping, which we saw earlier, enables viewers, to visualize representations of real-world dynamics in an intuitive way. In this section, to introduce a more complete set of the advantages of animated maps, we will base our approach on Shneiderman's [70] well known Visual Information-Seeking Mantra whose punch-line is: "Overview first, zoom and filter, then details-on-demand."

Shneiderman's principles suggest that any visual exploration task should begin with an overview, or overall representation of the data. Ogao and Kraak [58] advance that temporal animated maps offer "scientists the opportunity to deal with real world processes as a whole rather than as instances of time." Although Andrienko and Andrienko [4] consider that animated maps do not fully meet the requirement of overview tasks (because all the referrers cannot be seen in one glance), they conclude that animations remain indispensable tools because no other generically applicable[3] technique can provide a better or even equivalent overview of large spatio-temporal datasets and of the behaviors they record.

The next stage of the Visual Information-Seeking Mantra is "zoom and filter." Zooming in the spatial dimension is analogous to focusing on specific extents of the time-line. One might want to explore what is the behavior of a phenomenon in a particular area during a particular time-extent. The fact that any stage of the depiction is immediately preceded by a stage representing a real-world moment that existed immediately before and symmetrically succeeded by a stage that existed immediately after it makes Blok [13] say that the viewer can see information in its "temporal context." One of the functions of temporal legends is to assist the map reader in this task.

Spatio-temporal behavior observation and related pattern detection are among the most important tasks made possible by map animations. This task can be considered to cover the whole spectrum of visual exploration. Such behaviors and patterns can appear to be fairly complex. They are related to at least two dimensions of space as well as to the temporal dimension. They might even include an attribute dimension which can itself change with time. Many authors defend the capacity of temporal animations to reveal subtle spatio-temporal patterns that are "not evident in static representations," sometimes "even to expert users who are highly familiar with the data" [30]. Several experiments, such as those done by Dorling and Openshaw [17], MacEachren and others [44] Andrienko and others [6] or Blok [13], to name but a few, give evidence of this.

[2]For reviews and practical experiments on comparing small-multiples with animated maps, the reader might refer to [26, 30, 72].

[3]For some types of objects, i.e. moving objects, the space-time-cube technique can be applied efficiently but with some limitations

The last stage of Shneiderman's mantra is "details on-demand." One of the great advantages of animations over series of small-multiple maps is "their ability to display micro steps in complex systems" [30]. Such a possibility might enable to detect small changes that would otherwise have been missed but in general, it is precisely these micro-steps which make animations represent processes in an appealing and often intuitive manner. In addition, for animations aided by an interpolation mechanism, it is possible to focus into the temporal dimension further than a true reflection of the data. For cases where spatial and temporal interpolation can be assumed to come close enough to the behavior of the real phenomenon, such a mechanism can be useful for a scientist to view micro-steps mostly to relate the state (attribute, position or shape) of an object with its environment or other objects.

Although in early stages of exploration, information seeking tasks might resemble passive or perhaps curious observation of the information displayed, in more advanced stages, it often involves a more inference-based approach. At these stages, the expert might formulate – in writing or mentally – fairly precise hypotheses. Interactive controls of animation enable him to test a large number of such hypotheses in a short time, further enabling him to formulate new ones ...

Often, animated mapping will be only one of the techniques used by a researcher. The knowledge gained, the patterns detected and the hypotheses formulated with the help of the animated display, can lead the expert to identify another technique as a potentially effective means for acquiring new knowledge. For this reason, environments interactively connecting an animated map display with multiple alternative representations are often used.

In the end, we saw that animated mapping had advantages to offer at all stages of "visual information seeking" as enumerated by Shneiderman's mantra. We will now conclude this section with a remark on interactivity and temporal legends. We have already said that we consider both to be inherent to sound animated map design. Both are intimately related to most – if not all – advantages stated here.

2.7 Perceptive and cognitive limitations of animated maps

Several authors have mentioned the perceptive and cognitive limitations of animated maps. Among them, Sara Fabrikant and Mark Harrower have studied the question in depth [20, 21, 29, 30]. In his article "The cognitive limits of animated maps", the latter integrates concepts from psychology studies on the cognitive aspects of the use of animated graphics. The framework he adopts is that of the Cognitive Load Theory (CLT). The basic principles introduced from CLT are those of long-term memory (LTM) and working memory (WM). While LTM "stores knowledge and skills on a permanent basis", WM "performs tasks associated with consciousness and actively processing incoming stimuli" [29]. We

will present hereafter the four most important cognitive and perceptive limitations related to animated map reading tasks: change-blindness, cognitive overload, retroactive inhibition and split-attention.

Change-blindness

Experiments with animation showed that observers sometimes fail to notice large changes in the graphics. Harrower and Fabrikant [30] advance that this "change-blindness effect operates even when viewers know that (changes) will occur." This points out that solutions should be found to tackle change-blindness.

Cognitive overload

The biggest and most generic problem posed by animated maps is "cognitive overload." Harrower and Fabrikant [30] say "Regardless of the map-use goal, unlike static maps that do not change, the individual frames of an animated map are on-screen briefly and there is little time to examine fine details. In other words, there are obvious cognitive and perceptual limits that must be understood and used to inform map design. We believe exceeding these limits – which is easy to do with today's massive and complex datasets coupled with powerful computer graphic cards – is likely to leave the user frustrated or unsure of what they have seen" [30]. Studying this same problem, Harrower [29] concludes that "the problem is not that map viewers are incapable of seeing the changes occurring on the animated map, it is that they have well-documented trouble remembering what they saw and integrating it into their knowledge schemata."

Retroactive inhibition

Another limitation treated by Harrower which is closely related to that of cognitive overload, is termed "retroactive inhibition". It relates to the fact that "what comes first in the animation often has to be remembered and understood in order for what comes next to make sense." The act of remembering, in CLT terms can be expressed as the passage of information from working memory to long-term memory. ""Retroactive inhibition" occurs when there is insufficient time for this to happen, resulting in a kind of cognitive jam."

Split-attention effect

The fourth limitation that we will cover here is "split-attention". Split-attention is what occurs when a user needs to simultaneously attend two different cognitive objects that are separated in space or in time [29]. It is an effect well known by cartographers because of the frequent use of legends: the information in map

legends often needs to be frequently referred to for the information mapped to be understood. For temporal animated maps, the problem is amplified because of dynamism. If the map viewer needs to look at the temporal legend during an animation, he will not see part of the animation and possibly even lose track of the dynamic elements that he was attending to. We will discuss this more in upcoming sections.

2.7.1 Cognitive Load Theory for exploratory mapping

Harrower derives guidelines for the design of animated maps from the principles of CLT as well as from related knowledge within the cartographic field. However, the author cautions us that the work he reviewed on animation "is clearly aimed at students in a classroom." It might therefore not be directly applicable for "highly experienced and motivated map readers." To characterize students, we could say that they might not have the possibility or be willing to spend much time on an animated visualization. The animated maps are therefore expected to be understandable with a rather small investment of time and, hence, with little interactive exploration. Because of the characteristics of this public, to augment the effectiveness of the animated maps, Harrower suggests techniques like "looping", "slowing the rate of playback or providing built-in pauses" and, in general, techniques characterized by a lesser level of interactivity than those traditionally destined to the more specialized groups of users. Field experts and specialists in Geovisualization, on the other hand, are usually provided with tools with a higher level of interactivity and are characterized by a willingness to spend more time exploring the dynamics present in spatio-temporal datasets.

Our focus here is on the use of animated maps for exploratory purposes. Nevertheless, we will see in the next section that the concepts of the Cognitive Load Theory and the ideas put forward by Harrower, Fabrikant and other authors can be used to inform the design of an animated mapping platform. In particular, we will show how change-blindness, cognitive overload, retroactive inhibition and split-attention can be reduced by various features of a well designed interactive mapping interface.

2.8 Design requirements and recommendations for animated maps

The purpose of this section is to offer a set of design recommendations for an animated mapping platform. It contains what we consider to be the most useful generic functionalities for animated mapping proposed in literature. As announced, this section is a in depth continuation of the three basic needs that we treated in subsection 2.5.3. Here, we have regrouped the functionalities proposed into eight categories. The first contains rules that transfer directly from static on-screen mapping. The second treats of temporal legends, the third of

controls provided to the user for controlling display time. The fourth offers solutions to the split-attention effects. The fifth presents recommendations for zooming into the spatial dimension and for focusing into the temporal dimension. The sixth presents the usefulness of a temporal scale/speed control functionality. The seventh exposes useful techniques to emphasize change in animations and finally, the eighth shows the complementarity of animated displays and small-multiples as well as alternative representations of the data. For some of these categories, we will show how the particular design element helps to solve problems mentioned by Harrower and explain the benefit brought by the element from the perspective of the Cognitive Load Theory.

2.8.1 Rules transferred from static on-screen mapping

The graphical elements of animated temporal maps often don't differ greatly from those present in static maps. In effect, many animated maps' graphics look just like static maps, whether they make use of special symbols to represent dynamism or not.

Although the symbols used to represent real-world items are similar, the graphics of animated maps should be less complex than those of static maps in order to compensate the augmented cognitive load brought by dynamism. The most obvious change between static and animated mapping might thus be to adapt the symbology used. Rensink [68] identifies four as the number of "novel items that can be held in WM" at once. Classes for choropleth animated maps, for instance, should probably not exceed a number close to four or five and it might be advantageous to use even less when possible as Acevedo and Masuoka [2] showed in their experiment on urban growth animated mapping.

In static temporal mapping, the visual variable shape is commonly used to represent movement or change. Graphical symbols such as arrows or lines are used to represent a moving tendency or successive fronts of a phenomenon. For animated mapping, although the temporal and thus dynamic characteristic of the data is already explicit in the animation, similar use of graphical symbols is often made and considered useful to help the viewer understand the change he is seeing.

Along with using a simplified symbology, other ways to clear the display from unnecessary clutter are switchable data layers and filtering. In on-screen mapping, whether it is static or animated, the possibility of switching on or switching off data layers is a recommended feature. Data filtering, in a similar way, enables the animated map designer to remove noise and items that the map reader doesn't want to observe.

Hence, a complete animated mapping environment should offer the user the ways to quickly: aggregate data (to represent less data classes), use static symbols representing dynamism (such as arrows, e.g., for wind or ocean currents) and to switch data layers on and off.

2.8.2 Temporal legends

Temporal legends for orientation in time

Adding temporal legends (TL) to animated maps and animated mapping environments is recommended. Kraak and others [39] explain that legends help the viewers to locate the information displayed in the temporal dimension and "apply an appropriate temporal schema that allows them to interpret meaning inherent in the sequence and pacing of the animation."

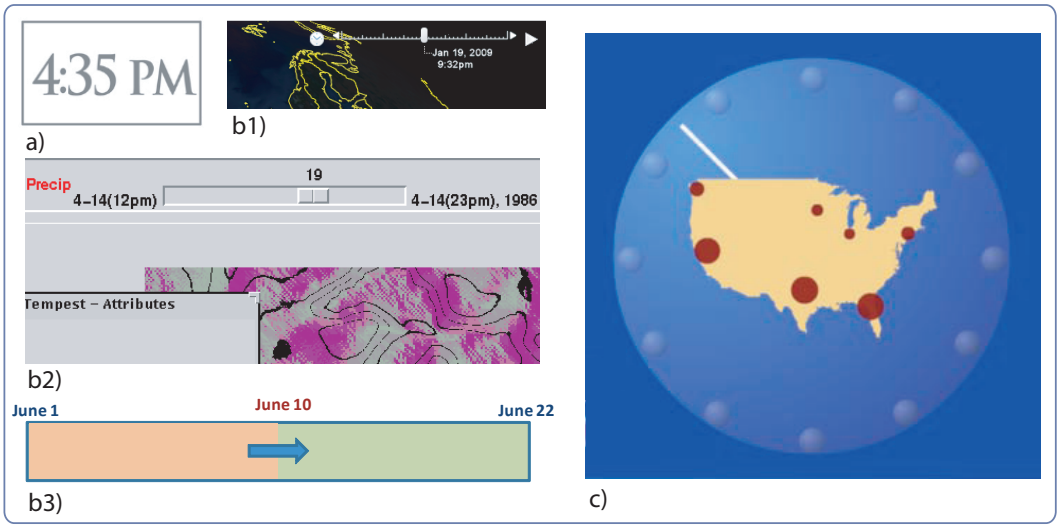


Figure 2.2: Three main types of temporal legends: a) Digital clock (from (30)), b) Time-bar (b1 from Google Earth (25), b2 from (19) and b3 adapted from (30)) and c) Cyclic temporal legend (from (46))

The three most frequently used types of temporal legends are: digital clocks (or "text" TL), time-bars and time-wheels (also termed cyclic TL). These are exemplified in figure 2.2. Besides research on graphical temporal legends, sonic temporal legends have also been suggested and developed. Research is still in its infancy and we will not discuss it further.[4] We will now review the main principles and advantages of these different legends, starting with the digital clock.

Digital clock

The digital clock is a dynamic text item that precisely tells the map viewer what real-world time the presently viewed graphic state corresponds to. The text actually constituting this information can be made of any combination of string

[4]The reader may refer to Kraak and other [39], [30] and Slocum and others [72] for reviews and further references on sonic temporal legends.

characters commonly used and understood by the user population to write out time (e.g., 13:56:08 for a clock time, 2008-12-25 13:56:08.136 for date and time precise to the 1/100 of a second).

Time-bar (Linear time legend)

The time-bar (Fig. 2.2-b) is a graphic temporal legend whose generic characteristics are a timeline with a moving symbol representing the real-world moment corresponding to the frame displayed on screen. The symbol representing the present is sometimes called the *present vehicle-sign* [39]. Time-bar legends can be designed in various manners. One solution, such as shown in example b3 of figure 2.2, the legend is made of three components. On the left hand side is an area representing the time that has already elapsed in the animation, in other terms, the real-world past. On the right hand side appears an area representing the real-world future. The third element is the line or symbol separating past and future, i.e., the present.

Cyclic temporal legends

An example of a cyclic legend is given in figure 2.2. Its principle is very similar to that of an analog clock. Much like the hand of a clock, the present symbol advances to demonstrate the passage of time. The advantage of cyclic temporal legends is to visualize behaviors that are related to temporal cycles among which the most common in practice are years, weeks and days.

Taking up the example on transport that we saw earlier, the analyst wants to get insight on how transport intensities vary with daily hours. He will benefit from visualizing the time span as a cycle as it will help him notice the relation existing between the intensities and the time of day – represented by the position of the present symbol in the cyclic legend. Exploratory visualizations using cyclic legends would most often benefit from combining it with one or both of the other legend types.

Combining different types of temporal legends

The choice of legend type depends, according to Kraak and others [39], on “the nature of the spatio-temporal phenomena displayed (. . .), the nature of the temporal queries that users are expected to make, and the knowledge schema concerning spatio-temporal entities” that the designer wants to prompt.

As we foresaw, it appears often to be beneficial to combine two or even three types of temporal legends. In effect, while the time-bar is very helpful to tell the viewer how far from the beginning and the end of the temporal extent loaded is the presently displayed state, a digital clock, possibly very close to the time-bar is useful to tell him exactly what that time is. In fact, such an association is very common. For time-wheels, it would often be needed for the user to locate

the time cycle he is presently visualizing in a broader time-span. The time-bar is ideal for that.

To conclude on this first of the sections related to temporal legends, we want to mention that they often serve a double purpose: not only to they help the user orient himself in time but, through interactivity, they are made into a time navigation tool [39].

2.8.3 Interactive control of display time

To introduce the need for powerful interactive control over the temporal dimension of a dynamic data depiction, we will draw a scenario of an exploratory process from the cognitive point of view. In his exploratory task, the user progressively makes sense of more and more information. To constructs an understanding of the dynamic phenomenon under study, he needs to regularly "shuttle" information from working memory to long-term memory [29].

The newly acquired knowledge improves the user's understanding which in turn enables him to improve the knowledge schema that he applies to the visualization task. He for example expects to find particular behaviors and has newly formed, more precise, hypotheses that he needs to test. In a cyclic manner, an understanding of the phenomenon is built for general trends as well as for detailed ones.

To allow shuttling of information from WM to LTM and solve the problems of cognitive overload and retroactive inhibition, authors unanimously recommend the provision of interactive control for animated maps. Such controls can help the user to reduce the cognitive strain provoked by the dynamic graphics. The question thus arises of the type of interactive controls to provide for animated mapping and in particular for exploratory purposes.

With the development of animated mapping techniques and of supporting technologies, different types of interactive controls have been provided to users. After the age of VCR-type controls and forward-stepping, the present stage of computers seems to allow more powerful tools. The interactive time-slider of common software like Windows Media Player as well as similar items provided in Google Earth or specialized software like Tempest [19] show a powerful and promising way of interacting with the temporal dimension of data.[5] NNN An important difference needs to be made between the usefulness of a slider for raster animations and for vector. To our knowledge, no existing software for raster animations allows the user to play the images in a smooth sequence by moving the slider backward and forward. The big emphasis that we put on the time slider might therefore apply only to vector animations (for a few more years).

[5] Similar interactivity was added on cyclic temporal legends. We will however not discuss such features for reasons that we will expose in the next section.

The principle behind the time-slider in animated mapping is to add interactivity to the time-bar temporal legend. The present vehicle sign is made interactive and can be clicked and dragged. For vector graphics, which are lighter than raster, dragging the time-slider enables the user to browse through the animation with a lot of flexibility. With the single and simple action of moving the mouse from left to right and right to left, the user can:

1. Jump to a chosen time,
2. Instantly choose a temporal subset or temporal region he wants to explore,
3. Loop over this chosen temporal subset,
4. Quickly pass from viewing short temporal extents to broad ones,
5. View the animation in forward and backward mode,
6. Tailor the speed at which he views particular depicted events,
7. Pause and start playing the animation again to allow information shuttling from WM to LTM when necessary.

The list of actions enabled by the temporal slider is thus large. Most of these actions cannot be achieved as quickly, simply and with as much flexibility with other types of temporal controls. However, some of these might be imperfectly practiced with a time-slider and even for vector data, the realization of such powerful interactivity might not always be possible because of computer memory limitations.[6] Nevertheless, it appears to be the single most useful control for an exploratory animated mapping environment. -NNN

2.8.4 Solutions to the split-attention effect

The split-attention problem was explained earlier (section 2.7). The type of split-attention we will try and offer solutions for is solely related to the user's attention being split between elements in the dynamic display and the – dynamic – temporal legends. Some authors studied the use of sonic temporal legends along with graphical ones to solve this problem but, despite the potential effectiveness of such solutions, we will not treat them here. Regarding graphical solutions, the position and integration of temporal legends in the map display will be reviewed. Then, we will present a hypothesis about why interactive time-sliders might be particularly useful for solving the problem of split attention.

Temporal legends placed in a separate display area reserved for graphical user interface items is problematic because it is more likely to be distant from the dynamic elements the user is attending to (see Fig. 2.2 b2)), thus increasing split attention effect. Embedding or superimposing temporal legends over the map

[6] In cases where this would not be possible because of too massive datasets, solutions like data aggregation, spatial and temporal focusing or falling back on plain animation would be possible.

itself were proposed as solutions (see the famous Google Earth GUI in Fig. 2.2-b1) by Kraak and others [39]. Very large temporal legends were designed, for example by Mitbo and others [46] (see fig. 2.2-c) probably with the objective of making the user subconsciously aware of the position of the present sign.

We now want to present a hypothesis on the usability of time-sliders to decrease split-attention for vector animations. Our hypothesis is that a time-slider is not only graphical user interface feature but that it also links to a tactile interface via mouse gestures. We argue that it is a powerful interface to a tactile-related mental image of the temporal dimension. In a preliminary phase, the user can get familiar with the relation between the mouse movements on the one hand and the motion provoked in the time-slider on the other hand. Such a relation can be memorized and the user can approximate the movement in time that he is triggering, solely by knowing how much he moves the mouse. Approximately knowing where the presently viewed state stands within the temporal extent has two and perhaps three cognitive advantages. First of all, we may argue that the problem of split-attention is partly solved because the main referrence becomes tactile and not visual anymore. The user needs to look at the temporal legend less often. Secondly, when he needs to look at it, the diversion of attention lasts less long because he approximately knows where to find the present vehicle-sign. Finally, an additional advantage might result from the use of a second sensory channel. The effective working memory capacity of the user might be increased.[7]

In conclusion, various solutions were reviewed from literature as well as proposed by us to help solving the problem of split-attention. However, it must be stated that experimental testing has not yet been reported on to assess the effectiveness of any of these solutions.

2.8.5 Zooming and temporal focusing

In animated maps, the changes occurring can be complex and numerous for the users' cognitive abilities. Another problem with animation and especially with interactive control of the temporal dimension is that computers might not be powerful enough to neatly mimic the dynamism of the data. To solve these problems, focusing on subsets of both the spatial and the temporal extents often reveals to be useful. We needn't present spatial zooming here and will treat only temporal focusing.

Temporal focusing is less common than spatial zooming just as animated maps are less common than static ones. It is analogous to spatial zooming in the sense

[7] In his review of CLT literature and in particular of how effective adding sound to animations is, Harrower [29] reports: "The amount of information that can be processed using both auditory and visual channels exceeds the processing capacity of a single channel; thus, limited WM may be effectively expanded by using more than one sensory modality." We can argue that this also applies to the sensory channel of touch. CLT authors Kalyuga and others [33] point out, in Harrower's words, that "dual-mode presentations do not reduce extraneous cognitive load or make the presentation inherently less complex; rather, they increase effective working memory capacity."

that it helps the user to focus on a subset of the data, not spatially but temporally. A temporal subset of the data is chosen to be visualized in a dynamic way.

Except for reducing computer and human memory strains, temporal focusing has the potential advantage of facilitating a proper temporal schema in the reader's mind. In effect, choosing moments for the beginning and end of the visualization enables the user to more quickly understand and remember what moments of the real world phenomenon the different moments of the animation correspond to.

2.8.6 Controlling the temporal scale/speed

We already introduced the notion of temporal scale and explained its relation to the speed of an animation. As Harrower and Fabrikant say, the scale of most temporal animated maps does not vary through the animation. Animations in which the temporal scale changes to match increases and decreases in the intensity, complexity or interest of the real-world change viewed are called event-based animations [72]. For an exploratory environment, we argue that complicated design involving temporal scale change would not be part of the set of basic components but of a more advanced set. Similar speed-variation effects can be achieved via a time-slider.

2.8.7 Emphasizing change: smoothness, saccades and dynamic variables

In the present section, solutions to change-blindness and to difficulties to remember and thus make sense of what one has seen will be treated. Various techniques have been proposed in literature and used in practice with the goal of having one of three following effects: make the change easier to see, make the change easier to understand or make the change easier to remember.

Smooth change versus saccaded change

Smooth change of dynamic graphics can have both positive and negative effects on the ability of a user to interpret and to make sense of an animation. Change can be difficult to interpret because of characteristics in the data. For example, too few states of a continuously changing phenomenon might have been recorded by instruments inducing a low temporal resolution. Taking the example of moving objects, it might not appear obvious to the viewer which object moves where from one moment to the next. Applying spatio-temporal interpolation to such moving objects can be an effective way to help the viewer interpret what he is seeing. Therefore, in this case, smoothness, via the application of interpolation, improves the visualization of a phenomenon.

In other cases, smoothness of change makes it difficult to make sense of the change and to memorize it. Famous experiments [71] showed that animation viewers can be blind to very important changes in a display, even if they are aware that some change will occur. Making animations less smooth or in other terms, voluntarily reducing the temporal resolution and making the changes happen in a more abrupt way can help readers remember the changes witnessed and hence make sense of the phenomenon.

Attracting attention with dynamic visualization variables

In a previous section, we introduced the notions of temporal scale and temporal resolution as main characteristics of temporal animated maps. Another framework to describe animations and the changes they contain was developed by DiBiase, MacEachren and others [14, 42, 43]. They called ”dynamic visual variables” a set of animation characteristics constituted by duration, rate of change, order, display date, frequency and synchronization. Later, Blok [13] used this framework and renamed a more limited set “dynamic *visualization* variables” (DVV) which we consider slightly more appropriate.

This framework is useful to study the characteristics of animations. However, it is not as useful for vector-based exploratory temporal animated mapping applications as it can be for non-temporal animations. In effect, in an exploratory context using simple animation techniques, most of these variables would be fixed as the animated display simply reflects the data and its temporal steps. Nevertheless, use can be made of dynamic visualization variables to emphasize change. We consider, for example, using a flashing symbol to attract the viewer's attention on particular events. We therefore consider using the DVVs as special effects for attracting attention more than as a framework explaining the characteristics of our animations.

2.8.8 Complementarity of small-multiples and alternative views

As we mentioned, remembering what one has seen in a dynamic display is difficult for many animated map readers. Series of small static maps showing successive chosen representations of the process can be very useful to help the viewer build a mental model of the dynamics he is observing. Such series of maps are called small-multiples. The biggest advantage of small-multiples may be for comparing the states of one or several phenomena at different moments of time. Animation is not well suited for such a task but it is more effective, to study the dynamics of change and to show many small changes [72].

Non spatial representations of datasets, like small-multiples, can be very usefully combined with animated maps. Spatial depictions are effective means to visualize the distribution of phenomena in space. Animation of such depictions is useful to represent the evolution over time of such phenomena. However, the attribute dimension (i.e., the 'what' of the where-when-what triad) can often

not be well represented on a map display and this is even truer about animation. For this reason, alternative views of the data are often recommended. For instance, if a researcher is interested in the detailed evolution of population density in a series of administrative entities, an animated choropleth map alone is not a suited solution. It has been demonstrated that change in color at fixed locations is not an effective means to visualize attribute change [17]. In this case, a useful alternative view of the data would be a line-plot of the population densities with time on the x axis, density on the y axis and multiple lines representing the administrative entities. Useful guidelines for representing spatio-temporal phenomena can be found in Monmonnier's [47] article "Strategies for the visualization of geographic time-series data."

2.9 Conclusion

In the present chapter, we have established animated mapping within the field of Geovisualization and provided frameworks to understand animated maps and the types of tasks they can be used for. We used our study of their principles, advantages and limitations to propose a set of recommendations for the design of a visualization environment using animated map displays.

The basic features that such an environment should have are temporal legends and a way to interactively control the temporal dimension. The time-slider is recommended for such a task and argued to be a very powerful and flexible interactive tool for exploratory animated mapping. Other necessary features are simplified graphical representations, control of the temporal scale (or speed) of the animation and selection of the temporal extent of the data to load. In addition, dynamic visualization variables can be used to emphasize change and series of small-multiples as well as alternate views of the data are recommended as complementary to animated maps. As announced, the recommended set of features is generic and would be suitable to build animated mapping applications for data exploration as well as for presenting the data to wider audiences on the internet.

Chapter 3

Technical choices: WMS, SVG and RIMapperWMS

3.1 Introduction

The main objective of our research is to look into possibilities, both theoretical and practical of combining interactive vector graphics animation and Distributed GIServices. In the present section, we are first going to justify our technical choices of Distributed GIService framework and graphics format. To do so, we will show that WMS and SVG possess the required capabilities for our setup and that they are equivalent or superior to any comparable frameworks for the same purposes. Then we will determine whether, from a theoretical – or specification – point of view, it is possible to combine the two chosen components.

The basic requirements for the two components are as follows:

- On the server-side, we need a standard compliant, widely implemented, distributed service framework possessing a specification not only for the spatial and attribute components of the data but also its temporal component.
- On the client side, we need a vector graphics format supported on the Web that allows to interactively manipulate animations.

After this, we will briefly study four existing WMS implementations which already render SVG maps but without yielding any animation. We will explain why the RIMapperWMS implementation is the best suited of the four candidates to be extended for our purpose.

3.2 Why OGC's WMS as distributed GIService?

New possibilities for sharing geospatial data and geoprocessing capabilities came forth with the uprise of internet technologies. Web clients could communicate with map servers which in turn were linked to data stores. Yet, it is only with the efforts towards interoperability and standardization that multiple map products could be combined together. The Open Geospatial Consortium (OGC) is the leader and centralizing organization in this interoperability enterprise [63]. The specifications for standardization are developed in an open process involving private parties, large geosoftware companies and academic institutions.

The resulting recommendations enhance the sharing of geospatial data and geoprocessing software components because standardization makes it easier to combine products. An increasing number of private and governmental agencies decide to comply to OGC's standards because of the enhanced possibilities. Products distributed in the OGC framework are either freely shared or sold following traditional commercial ways.

The joint effort is responsible for a number of Open Web Services (OWS) specifications among which the most widely adopted is the Web Map Service (WMS) specification [36].

3.2.1 Web Map Service in a nutshell

OGC's Web Map Service, also popular under its WMS acronym, is the standard for mapping and visualization. The implementation specification is mature and well documented [61]. This is demonstrated by the fact that it has been adopted by the International Standardization Organization (ISO) as one of its standards (ISO 19128) [31].

Like all computer networking standards, WMS specifies how the interfaces of services should be defined. So doing, the exchange mechanisms between the three main components of Geo Web services, web clients, map servers and data servers (see Figure 3.1) are standardized. Service developers remain free to store or process data according to methods that suit their internal needs and resources.

The big advantage of WMS is that "software conforming to the WMS specification, using ordinary Web browsers, is able to automatically overlay map images obtained from multiple dissimilar map servers, regardless of map scale, projection, earth coordinate system or digital format" [59]. Furthermore, one ore more of these map images may well be the result of complex geoprocessing. This leads to important time and financial gains.

After introducing what WMS offers to service providers and thus to end users, lets briefly review how WMS works? There are three basic types of requests that can be asked to WMS implementation: the `GetCapabilities` request,

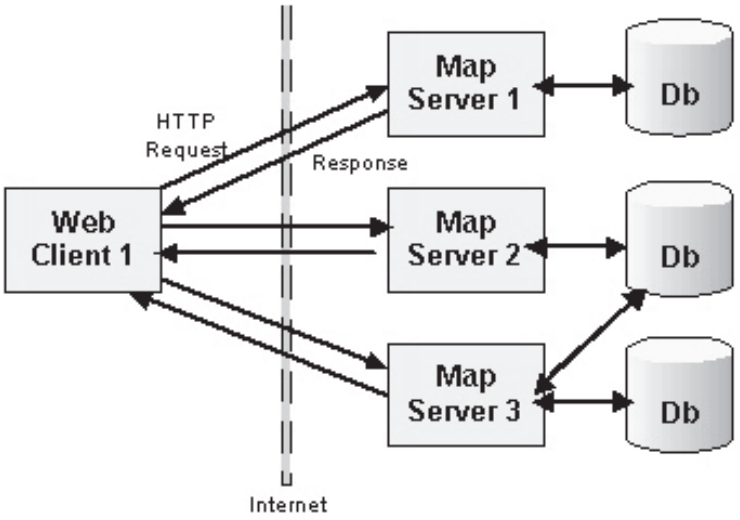


Figure 3.1: Three-tier basic WMS setup (from (59))

the `GetMap` request and the `GetFeatureInfo` request. The first two are compulsory for OGC compliance and the last is optional. Köbben and others [36] inform us on what these requests do:

GetCapabilities The `GetCapabilities` request is "used by client software to ask for the capabilities of the service: what layers are available, what projection system can the maps be delivered in, what output formats can be requested, etcetera."

GetMap Based on the response to a `GetCapabilities`, "the `GetMap` is issued to ask for an actual map. Because the client knows the possibilities of the service from the `GetCapabilities` response, it can issue its request with specific parameters asking for example for one or more layers of information."

GetFeatureInfo The optional `GetFeatureInfo` mechanism is provided by "services that advertise layers of data as *queryable*." It is used to "find attribute values in the underlying data."

Each request is entered by the client in the form of a Uniform Resource Locators (URL) which contains appropriate parameters. The map server replies to the requests in the following ways: To a `GetCapabilities`, it responds an XML document specifying the required information. To a `GetMap`, it renders a map in the specified graphic format, possibly including legends. To a `GetFeatureInfo` request, which queries for values of attributes for specific locations in the map, the map server reports the attribute values back.

3.2.2 WMS support and format for the Time dimension

Many projects exist which build animations using the WMS framework for at least part of the process (for example, see [48]). Most of them end up using Google Earth or common media player software (such as Windows Media Player or Quick Time) to render the animations. The present interest for binding animations to distributed services is shown by the fact that OGC published Eric LaMar's [41] project to extend the WMS specification for animation services on the same Web page as the WMS specification.[1]

We have shown that OGC's WMS is presently the most suited candidate for building a map server based on standardized distributed services. The next step, which is the subject of the present subsection, is to determine whether WMS's specification offers an appropriate standard for the sharing of data containing a temporal dimension.

The current version of WMS Implementation Specification [61], version 1.3.0, includes support for the temporal dimension of data. A third level subsection of the specification is named "Temporal CS". It is part of the specification on "Coordinate systems" which does not seem logical. For this matter, it is our impression that the 1.1.1 specification [60] was better structured and used a more appropriate terminology: the broader section was termed "Common Request Parameters" and the unit on time was called "Time Dimension". These changes and this confusion seems to show that perhaps the norm for time is not totally mature yet.

However, the information and further references offered for the "Temporal CS" seem complete and highly relevant for our purpose. It says that "A WMS may announce available times in its service metadata [(`GetCapabilities`)], and the GetMap operation includes a parameter for requesting" georeferrenced graphic results for particular times. Two "normative" annex documents are referred to: Annex C, called 'Handling multi-dimensional data' defines the options of requesting single values, lists of values or intervals for different types of multi-dimensional data. The *dimensions* in this document refer to "Time, Elevation and sample dimensions." Annex D, called 'Web Map Service profile of ISO 8601' specifies the "format of a time string."

For animated mapping, we are obviously interested in the Time dimension introduced in annex C as one of the multiple dimensions. `GetCapabilities` outputs and `GetMap` URL requests can contain a dimension element for which the set of mandatory parameters are *name*, *units* and *extent*. Optional parameters also comprise: a default value for the dimension, whether multiple value requests are allowed and whether nearest value rendering and current time values are possible. Most of these parameters are generic for all 'dimensions' but the 'current' parameter is obviously specifically temporal.

[1] Although the issues dealt with in this project are not the same as those we are interested in (because it is centered on raster data), the fact that OGC publishes an unofficial project on its main implementation page shows the existing interest for animations in the WMS framework.

The possibility to request and render time intervals is the most interesting feature of the dimension element. It allows a user to specify the temporal extent of the animation he wants to view. The desired temporal resolution for the extent requested can be further specified. The following line shows an example of a temporal interval.

```
2000-07-01/2000-07-31/P1D
```

The temporal interval shown here starts on the first of July 2000 and ends on the 31st. The `P1D` expression shows the periodicity of the measurements. We will see more on the format in which this information is conveyed below.

A framework for offering and requesting time related data is thus available and we can now review the format in which temporal parameters should be written. Annex D "specifies the encoding of moments and periods in time to allow the Web Map Service to support temporal data descriptions and requests." As its title suggests, the standard is based upon ISO 8601:2000 recommendation. It extends this standard by "defining a syntax for expressing in a single string the start, end and periodicity of a data collection." We will now show what the basic syntax elements are for specifying time moments and periodicity.

"All times should be expressed in Coordinated Universal Time (UTC)" although local time issues are taken care of by the standard. The basic way of expressing time uses a string to specify century, year, month, day, hour, minute, seconds and optionally a decimal point followed by fractions of seconds. Here below figures a schema and an example of such a string:

Schema: ccyy-mm-ddThh:mm:ss.sssZ

Example: `2009-01-28T13:53:41.007Z`

Cases for time AD are taken care of by using the '-' sign and to represent time from a distant past, the number of digits for the year is extendable. For example, -1500000000 corresponds to 150 million years AD, the Jurassic period.

The standard indicates "the time resolution of the available data" in a very compact way which uses letters to abbreviate temporal characteristics. Table 3.1 shows examples of the expressions and their meaning.

P1Y	→	Data for every year
P1M10D	→	Data for every 1 month plus 10 days
PT1.5S	→	Data for every 1.5 seconds

Table 3.1: Symbols for time periods (adapted from [61]).

Now that we have seen the syntax, we can explore what this would look like in the communication language between clients, map servers and data servers. The following is an example of a dimension element for time as it could figure in a `GetCapabilities` document:

```
<Dimension name="time" units="ISO8601" default="2003-10-17"
>1996-01-01/2003-10-17/P1D</Dimension>
```

This dimension element specifies that there is a time dimension in the proposed data, that its units are as recommended in the ISO 8601 standard, the default date (if the client does not specify which date he wants is the 17th of October 2003. The full extent available is from 01.01.1996 to 17.10.2003 with a daily temporal resolution (P1D).

In a GetMap request, the parameters found in table 3.2 might be specified for an animation in the mpeg video format.

Table 3.2: GetMap request parameters incl. TIME (adapted from (61)).

Parameters	**Explanation**
`VERSION=1.3.0`	Version of the WMS specification
`REQUEST=GetMap`	Type of request
`LAYERS=ozone`	Name of layer
`CRS=CRS:84`	Coordinate reference system
`BBOX=-180,-90,180,90`	Bounding box in lat/long
`WIDTH=600`	Width of the output image
`HEIGHT=300`	Height of the output image
`TIME=2000-07-01/2000-07-31/P1D`	Time interval and periodicity
`FORMAT=video/mpeg`	Output graphical format

3.3 Why choose SVG as a vector graphics format?

In search for a graphical format suitable for the animated mapping platform we intend to develop, we started by establishing what were the requirements or our needs. Firstly, the graphical format should support vector shapes and be well suited for cartographic purposes. Secondly, it should be well supported on the internet and run in web browsers because the internet is the means we chose to disseminate animated maps whether the final user is the general public or field experts. Thirdly, the graphics engine should be capable of multiple types of animation, and, fourthly, it should be possible to program functions for the user to be able to interactively control the temporal dimension of the animation.

Before studying the candidate graphical formats, we want to remind the reader of our choice of building our system following a thin-client architecture. The reason for this choice is that the setup should be usable both for data exploration and dissemination of animated maps on the WWW. For exploration purposes exclusively, it would have been possible to develop a desktop program working in a thick-client distributed architecture, only using the internet to access data. However, since we also intended to use the Web to disseminate animated maps, it made sense to go for an architecture suitable for both purposes at once, hence, a thin-client setup.

Two main vector graphics format were envisaged to render the type of interactive animations that we need: Adobe Flash (SWF) (originally developed by Macromedia) and Scalable Vector Graphics, usually referred to as SVG. Andreas Neumann [54] did a comparison of SVG and SWF in 2002. The main differences between the two formats appear well in this document but the author judged that the detailed comparison of features is outdated. Without going into a detailed comparison of the capabilities of these two formats, we will present them and expose the reasons why we chose to work with SVG instead rather than with Flash.

Flash, whose web publishing output is actually SWF, was the first "widely usable vector graphics format" for the internet and "is still the most popular." Flash graphics can be drawn and animated using a powerful authoring tool whose name is also Adobe Flash. However, this environment is not very flexible and cartographers often prefer to use the scripting capabilities of ActionScript. This scripting method is also necessary to add interactivity to the graphics. Another framework, Adobe Flex, provides possibilities to bind the vector graphics to a database. The format is widely supported in web browsers provided that the user installs the Adobe Flash Player plug-in.

However, despite all these advantages, Flash also has serious limitations for cartographic use. First of all, Flash is a proprietary format developed by one single company whose biggest category of customers are advertisers. This implies that cartographers might not have enough influence to propose or prevent new developments that would serve or harm their interests. Secondly, Flash is a binary format which is not human readable. Thus, except for Adobe's costly authoring tools, there is no effective means to develop high quality products. In addition, its binary characteristic also makes it closed to search engines. Text items present in maps cannot be found by this means. Finally, unlike the authoring tools, the SWF format itself is poorly documented [3].

Regarding cartographic use of Flash, Neumann and Winter [56] consider that it has been used by cartographers to develop "useful interactive cartographic applications" but that, after more than a decade of existence (the statement the authors made in 2003 remains true today), "there are still few online examples of Flash-based mapping applications." However, in contrast with SVG, Flash has been used to publish interactive animated maps on the internet. In effect, Mark Harrower and his students developed a series of animations with a rather low level of interactivity [27]

To present SVG's qualities, we will start by saying that, despite exceptions such as the one we have just presented, it is generally preferred to Flash by cartographers who want to publish their work on the internet and by Web GIS applications developers. To check this statement, one could have a look at the large number of cartographic papers presented at the annual SVG conferences [62]. Neumann and Winter [53] defend that "with the rise of SVG, there is for the first time a technology at hand that allows to represent all graphical elements producable by graphics and cartographic-software with the additional advantages of high interaction possibilities and animation, all based on open and standardized file formats and programming languages." Furthermore, Peng and Zhang [65] consider that it could become one of the "key standard technologies to facilitate the development of Web Geoprocessing Services." But why do cartographers like SVG so much?

SVG is developed at the W3C[2] as an open standard by a joint effort of participants from a variety of backgrounds. These stake-holders include cartographers which implies that although SVG was not especially designed for cartography, "the working group had cartographic applications in mind from the beginning" [56]. Coordinate systems, for example, are well supported in SVG. The openness of the process through which SVG is developed is not the only way that the format is open. The SVG specification is very well documented [77]. SVG is XML based, human readable and SVG text elements can be found by Web search engines. Its XML structure makes it particularly easy to integrate within complex Web projects.

SVG's syntax is clear, simple and powerful. It can be edited by any text editor. Graphical authoring tools also exist but these are not as powerful as the ones developed by commercial companies (such as Macromedia, which developed Flash, now owned by Adobe) and do not include possibilities for animation and interactivity. SVG's syntax is particularly suited to create vector shapes from objects contained in databases [81] which is one of our main concerns in this research. We will later present the way SVG is scripted.

However, SVG, like Flash, is not free of problems. The biggest tooth ache of the SVG community does not concern the specification itself, nor the authoring tools, but the browser implementations. To be viewed by Web users, SVG graphics need to be supported by his Web browser. The only Web browser that supports close to the full SVG specification is Opera, a browser specialized in mobile services. At the time of writing, SVG is still not supported by the most widely spread browser, Microsoft's Internet Explorer. Mozilla Firefox natively supports static SVG and interaction but no animation. Plug-ins exist for both these browsers and enables a user to benefit from almost the full SVG specification. Apart from this browser situation, SVG has some limitations that are related to the fact that, as we said, it was not developed specifically for cartography. Neumann and Winter, as well as Dunfey and others [18], discuss some of these shortcomings but offer solutions to work around them.

[2] W3C stands for World Wide Web Consortium, which is the standardizing organization for the Web

Weighing the pros and cons of Flash and SVG, our choice went to the latter because of several of the advantages we just presented. Among these, the facts that it is generally preferred to Flash by cartographers and that cartographers' opinions are considered in new developments, that it is more open in many ways, that free or cheap solutions are available to develop high quality complex applications and that its XML structure promises high interoperability with other applications, were among the arguments that we considered. However, the determinant argument is probably the adequacy of SVG's syntax for binding graphics to a database and this is particularly true about the syntax used to animate vector objects. We decided that these advantages outweighed SVG's limitations, including that of the browser situation. One reason that justifies this decision is the fact that the output of the prototype we intend to develop is destined to field experts rather than to the general public. We consider that experts having a strong interest in the phenomena that might be visualized with our platform will be sufficiently motivated to download another browser or install a plug-in. It must however be said that if the present project had been developed for economic purposes and not for academic research, our choice might have been different.

3.4 Interactivity and animation in SVG

There are many ways to provide interactive control over SVG graphics and there are two main ways vector shapes and their attributes can be animated. In this section, after a short introduction on SVG's syntax, we will present the interactive possibilities offered by scripting along with the first type of animation: scripted animation. After that, the second animation technique possible in SVG, which uses the SMIL language will be treated.

3.4.1 Short introduction to SVG's syntax

Before treating interactivity and animation, we need to give the reader a basic idea of how SVG is coded. The following code (adapted from an example found in [56] shows a basic SVG file containing four elements. Figure 3.2 below shows the graphical result one would view in a Web browser if the file was loaded.

```
<?xml version="1.0" standalone="no"?>
  <!DOCTYPE svg PUBLIC " //W3C//DTD SVG 20010904//EN"
    "http://www.w3.org/TR/2001/REC SVG 20010904/DTD/
    svg10.dtd">
  <svg width="500" height="400" xmlns=
    "http://www.w3.org/2000/svg">
    <desc>SVG Hello World example</desc>
    <circle cx="200" cy="200" r="100"
      style="fill:yellow; stroke:red;  stroke-width:10" />
    <text x="150" y="350" style="font-size:40pt"
```

```
      >Hello World</text>
    <polygon points="170,250 215,150 220,100 300,210 190,260"
       style="fill:blue ;
       stroke:lime ; stroke-width:5"/>
  </svg>
```

One can see how simple it is to script a polygon in SVG. Three of the four elements have a `style` label that contributes to describe characteristics of the shape. The `style` label is one of many attributes that shapes can have. We will come back to these attributes repeatedly.

3.4.2 Scripting interactivity and animations

Interactivity comes in many different forms in SVG: event-handling, hyperlinks, scripting, SMIL events and more [56]. Here, we will only discuss the advantages provided by scripting. Although "SVG is designed to be language independent," [56] it is very often used in combination with ECMA-Script (which is the standardized version of JavaScript). Because SVG is XML, every element and each of its attributes "can be accessed and manipulated using script" [56]. This offers developers great flexibility and power to program the interactive behavior they desire.

ECMA-Script can be used to animate SVG graphics. Köbben developed an interactive animation simulating flood risks in Kathmandu [35]. His application makes blue polygons symbolizing water visible or invisible as a clock runs. They are visible at real-world times when it is predicted that water will cover an area and invisible when it is predicted that water will not yet have reached or already have subsided from the area. Leaving interactive control aside, the code actually provoking the animation is very simple. It is constituted of a time engine and of a mechanism to make the flood layers visible or invisible.

To elaborate a proof of concept for our prototype on moving objects, we modified Köbben's application. While his application simply set the visibility attribute of layers to "visible" or "hidden", our application set the position of an object to mimic its movement. The code below shows the few lines, which, bound to the time-engine, animates the object:

```
function setPosition(time)
    {
    var i = 0
    for (i = 0; i < numberOfPoints; i++)
        {
        if (t_in[i] <= time && t_out[i] > time)
           {
           svgDocument.getElementById("MovingPoint")
              .setAttributeNS(null, "cx", xPos[i]);
```

```
            svgDocument.getElementById("MovingPoint")
               .setAttributeNS(null, "cy", yPos[i]);
            }
         }
      }
```

The setPosition function of the above code states that, if the current time of the clock is larger than (or equal to) the starting time (t_in) of a position the point took and it is smaller than the end time (t_out) of the same position, the x-y position of the SVG object called “Moving Point” is set to values called “xPos” and “yPos” (found in database columns). One disadvantage of this way of representing the movement of objects is that if only few positions of the trajectory were recorded, the object will appear to move in leaps. To fix this, supplementary position coordinates would need to be calculated applying spatio-temporal interpolation.

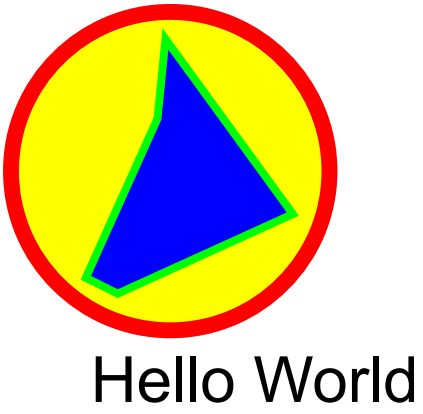


Figure 3.2: Simple SVG graphics example

3.4.3 SVG and SMIL animation

The Synchronized Multimedia Integration Language (SMIL) is another XML-based language developed and recommended by the W3C. Like SVG, it is well documented in a detailed specification [79]. SVG is called a called the host language in the way it makes use of SMIL. This status allows SVG developers to “amend or extend SMIL Animation in appropriate circumstances” [82].

SMIL’s declarative syntax is extremely compact and powerful. The following code shows a simple `animate` element.

```
<animate attributeName="width"
  from="100px" to="800px" begin="click" dur="7s" />
```

This `animate` element could be embedded within an SVG shape such as a rectangle. Starting at the moment the user clicks the shape (event), a smooth

animation will showing the width of the shape increasing from 100 pixels to 800 pixels in a duration of 7 seconds. The fact that the syntax is *declarative* implies that it is generally not necessary to understand how SVG rendering engines achieve the animation [82]. It is enough for someone using SVG to know the result he desires and to declare it following using proper code.

There are five different types of declarative animation in SVG from which four are borrowed from SMIL. The fifth, `animateTransform` was designed specifically to transform shapes in SVG. A detailed specification for SVG animation which shows how all elements work and how SVG hosts SMIL can be found on the W3C's website [78]. Below figures the five types of animations followed by the definitions given by Watt and others [82]:

- `animate`—A general-purpose animation element that allows time-based or event-based changes in scalar values of SVG attributes
- `animateColor`—Allows you to modify the values of color-related attributes and properties
- `animateMotion`—Allows animations that move an object along a motion path
- `animateTransform`—Allows you to create animated transforms, for example, by adjusting the value of a transform attribute
- `set`—Allows stepwise changes of attribute or property values

The general `animate` element can often be used instead of the more sophisticated functions by specifying which attribute should be animated.

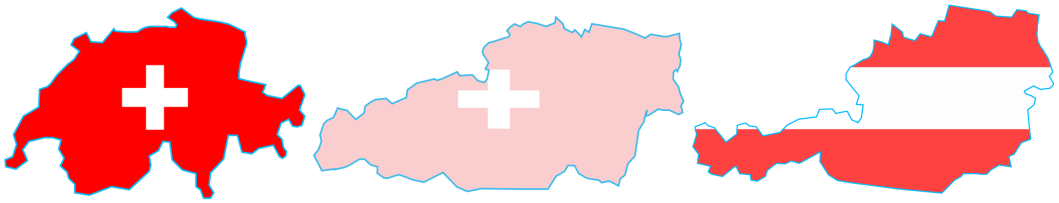


Figure 3.3: Example of the power of SMIL for morphed animations (from (55))

We did a combined study of the possibilities offered by these functions and the different types of animation that have been used in vector animated mapping. This study convinced us that the set of animation possibilities is complete. All graphical attributes of SVG, including spatial attributes, can be animated in controllable ways. We have already reported that Neumann and Winter [56] consider SVG is adequate for static mapping, which means that all of Bertin's [11] visual variables can be rendered. The most difficult elements to animate are polygons changing shapes. This type of shape merging is called *path-morphing* in SVG. Neumann and Winter [55] provide an example showing

the power of SVG SMIL animation for morphs. Figure 3.3 above shows snapshots of a morphing animation in which a polygon representing the territory of Switzerland is morphed into a polygon representing Austria.

Apart from all the visual variables being animatable, effects using the dynamic visualization variables[3] introduced by DiBiase, MacEachren and others [14, 42, 43], i.e., duration, rate of change, order, display date, frequency and synchronization, can all be realized using SVG and SMIL animation.

For many animated mapping cases, the biggest advantage offered by SMIL over scripted animation might not be the simple and powerful syntax but the possibility for interpolation. In effect, in the examples we saw earlier, the width of an object regularly increased and the shape of a polygon smoothly changed through the animation.[4] Interpolation can even be modeled to be non-linear and follow a "cubic Bezier spline function." While complex Bezier functions are beyond the scope of this thesis, we will study in depth and use the possibilities offered by interpolation and benchmarks to design our prototype implementation.

To further understand how animation works with SMIL and SVG, let's look at it from a temporal point of view. SVG files have an internal clock that is switched when the file finishes to load. All the temporal commands can be considered as projected on a time-line starting at that moment. There are two generic temporal attributes in SMIL animations: `begin` and `dur` and . The `begin` attribute specifies the event or moment when an animation should start running. If omitted, the animation will start when the file finishes loading. The `dur` attribute specifies how long this particular animation should take to unfold. In addition, different techniques exist to specify what the animation engine should do during that lapse of time. Every animation is linked to an attribute and values must be entered to tell the rendering system what to do. There are several ways in which one can specifies these values; we will briefly review two. The `from` and `to` attributes can be used when only two values need to be entered. A supplementary `by` attribute can be added. Another way is using the `keyTimes/values` method.

```
<circle cx="200" cy="200" r="100"
   style="fill:yellow; stroke:red;  stroke-width:10" >
     <animateTransform attributeName="transform"
        type="scale"
        keyTimes="0; 0.2; 0.5; 0.8; 1"
        values="1.5; 4; 4.5; 4; 1.5" dur="5s" />
</circle>
```

Using this method, one denotes `values` for an attribute which correspond to specified moments (`keyTimes`). It is particularly useful for data driven anima-

[3] As explained on p.29, the authors mentioned here had named the class 'dynamic visual variable'.

[4] It is possible to make this evolution match attribute and temporal benchmarks as we will see.

tions! We will explain more in detail how to use it in the second part of our work.

Many characteristics and possibilities of SVG and SMIL animations have not been covered here, such as animation chaining or multiple animation on a single SVG element. SMIL animation is easier to use and more powerful than scripted animation but scripting is more flexible. We foresee that in many cases, the two animation techniques might be advantageously used together, the latter patching up the limitations of the prior. One of our main objectives is to look into possibilities to interactively control the time dimension of animations. We cannot review here the mechanism that we foresee to achieve such high level interactivity. This is one of the main issues we will need to tackle in our implementation attempt. We can however state that the authors of SVG Unleashed [82] assert that the capabilities of SMIL can be supplemented by scripting which should enable us to gain access over the temporal logic.

3.5 Compatibility between the WMS and SVG specifications

As we have seen, the WMS specification for time is complete and, likewise, SVG offers a variety of animation types. SVG can be used to animate all cartographic visual variables related to the spatial and attribute dimensions of the data. The dynamic representations can be designed so as to make use of all the dynamic visualization variables identified for animated mapping. The final question to determine whether our two technologies can be used for our purpose is: *can the WMS Time dimension and SVG animation be combined*. This question further develops into the following: *does the WMS specification allow SVG as a graphical output format?* and, *from a practical point of view, is it possible to convert the time stamps in the ISO 8601 format to a format usable within SVG animations*.

The first question is easy to answer. Yes, SVG is one of the two "graphic element formats"[5] that the WMS specification allows and recommends. Flash is not mentioned as one of the graphic element formats allowed.

To assess the practical compatibility of SVG animation with the extended ISO 8601 standard, lets examine how the two recommendations for time compare. We will first focus on SMIL animations and then shortly discuss scripted animations as well.

Time stamps stored to record real-world processes would obviously be in real-world time. To the contrary, time in SVG SMIL is display time. We saw that the central time features in SVG are the `begin` and `dur` attributes. We also saw the very useful `keyTimes` attribute (`keyTimes` / `values` method) for animations having multiple data driven steps. A conversion is needed to transform

[5] "Graphic element format" is the name given to vector graphics format in the WMS specification.

real-world time stamps into values that can fit into the SVG SMIL animation `keyTimes` attribute. Lets see whether this task seems possible to achieve...

We will start by giving an example to try and convert moving object data into a SMIL animation. For a real-world phenomenon, we have t values for each recorded state of the object (attribute, position or change in geometry). We want to use these values within a SMIL animation using the `keyTimes` / `values` method which should be in the form of listing that follows. Real-world time stamps such as `2009-01-12T14:42:10Z` should be converted in a way that they can be integrated in a SMIL animation.

```
<circle id='Moving Object' r='10' >
   <animate attributeName='cx' keyTimes='? ; ? ; ? ; ? ; ?'
      values='list of x positions' dur = '?' />
   <animate attributeName='cy' keyTimes='? ; ? ; ? ; ? ; ?'
      values='list of y positions' dur = '?' />
</circle>
```

There are two different animate elements, one for the x attribute spatial attribute and one for the y. The `begin` attribute is omitted here which means that the animation will start as the file finishes loading. As for the `dur` attributes, their values should be the same and should reflect the temporal scale - or the speed - at which the user wants to visualize the animation. It is thus not directly related to the temporal extent in the data. The keyTimes attribute is the attribute in which the real-world time stamps must be converted. In the `keyTimes` method, the first 'keyTime' must always be 0 and the last always 1. How can this conversion be achieved? This is a question we will try to solve in chapter 5.

Now that we have shown what can be done to combine WMS Time dimension (WMS T) and SMIL animation, lets see what would be necessary to use scripted animations. This latter type of animation is more flexible than SMIL. The `keyTimes` / `values` method is not used and the time moments need not be converted into a display time format. In the work done by Köbben [35] in his exploration of the use of SVG scripted animation for mapping, the author developed a time engine which use a time format requiring fewer conversion operations from real-world time to display-time.[6]

3.6 Choosing a platform to extend: RIMapperWMS

In the previous sections, we have shown that, in theory, WMS and SVG can be used and combined to develop a standard compliant animated mapping environment for the internet. The next step in our contribution is logically to tryout this combination practically. Because the time available for this research would

[6]The reader may refer to appendix B to see parts of the code and explanations.

not permit to develop a complete system from a to z, we went in search for an already existing WMS system that could be extended to output SVG interactively controllable animations.

Four WMS implementations retained our attention in our search, UMN Map Server [45], Geo Server [23], SUAS MapServer [10] and RIMapperWMS. A brief study of these setups showed that among the four, only RIMapperWMS treats SVG as an interactive vector graphics format. The other three, as Köbben [34] already said about Map Server and Geo Server, treat the SVG output as "just another graphics format" which implies that, like for "GIF and JPEG output, the maps are basically pictures only, with no interactivity." This means among other drawbacks, that the geographical objects cannot be selected and manipulated in useful ways and, most importantly for our purpose, cannot be animated on an object basis (only on frame-by-frame basis).

Unlike the other implementations, RIMapperWMS takes advantage of the possibility to combine SVG with ECMA-Script to produce interactive outputs. As Köbben says, using these possibilities, it is "possible to build an SVG map, or rather an SVG application, that includes its own graphical GUI." Such a setup, with a built-in graphical user interface appeared to be excellent way of designing a thin-client animated mapper for the Web. In addition, RIMapper is already implemented with a database backend, which is one of the architectural requirements that we had. As potential risks, we need to mention that Köbben [35] considers the setup to be "not well scalable" because "it outputs the whole data extent in one client-side SVG file."[7] Because the advantages of the setup easily outweigh the risks and because solutions to limitations could further be developped, we decided to try and extend RIMapperWMS to output SVG interactive animated maps. In chapters 5 and 6, we will present more in detail how RIMapperWMS is built before designing the needed extensions for storage of standard-compliant time stamps on the database side and for animating the graphics on the client-side.

3.7 Conclusion

In conclusion, we can say that the Open Geospatial Consortium's WMS standard is the most suited Distributed services framework for our purpose because OGC is the standardization organization for Geo Web Services, because the WMS implementation specification is mature and well documented and because it also offers a standard for describing and sharing temporal moments and intervals. The SVG specification, on the other side, seems to be the most promising vector graphics framework to develop an animated mapping platform and this for several reasons we introduced. Firstly, and perhaps most importantly, unlike Flash, its main competitor, it is recommended by the WMS specification as one of the vector formats to be used. Secondly, the declarative structure of

[7] This problem could however be solved by using the Asynchronous JavaScript and XML (AJAX) technique.

its SMIL animation engine seems to be well suited for data driven animations. Thirdly, SVG is well suited for cartography and developed to support cartographers' needs. Finally, it is based on an open and well documented standard.

We also showed that WMS Time dimension and SVG could theoretically be combined to output animated maps. The two types of animation possible in SVG (i.e., SMIL and scripted animations) can be achieved using as time components, converted real-world time stamps originally stored in a format compliant with the WMS Time dimension standard.

Finally, in search for an existing WMS implementation yielding SVG interactive maps, we identified RIMapperWMS as the most promising platform to extend. The RIMapper setup, although it does not yet produce animated results, is designed to output SVG vector-based mapping application with built-in graphical user interfaces.

Chapter 4

Moving objects visualization and iceberg use-case analysis

4.1 Towards a visualization environment for moving objects

After identifying requirements for exploratory animated mapping in chapter 2 and determining which technical frameworks to use to develop a suited visualization environment in chapter 3, we now enter the second phase of this research. As the time available is too short to develop a generic animated mapping platform, we need to focus our attention on a particular application. With the objective of developing a prototype animated mapping platform, we will follow system development steps as they are presented in the Unified Process (UP) framework [8]. Figure 1.3 (p.8), that we already saw, shows the five workflows recommended in UP system development: requirements, analysis, design, implementation and test.

One of the main tasks the animation technique has been recommended for is moving object visualization. We therefore intend to develop a prototype application for the visualization of movement dynamics of individualized entities. As Dodge and others' [16] article shows, various types of moving object data exist. While we plan to develop an application which could be used for all these movement types, we will further focus our design and implementation to suit the needs of a chosen case-study.

The case-study we decide to develop our animated mapping system for is iceberg movement visualization. Iceberg dynamics is a phenomenon of great interest for several natural and applied sciences and has implications on human activities such as navigation and oil-mining. Because these dynamics are related to external natural processes, seasons in particular, we can expect to find in interesting patterns in iceberg tracking data such. A huge data collection is made freely accessible by the (American) National Ice Center (NIC) [49] containing all recorded iceberg positions since the end of the 1970s.

The present chapter is structured in three parts. First, a framework for moving object data is presented. Second, the ice-berg case-study is introduced - which can be considered to be a use-case, as understood in UP terminology. Third, the iceberg use-case is summarized in requirements and these requirements are further analyzed. This objective of the final requirement analysis is to inform the design of the system, object of the following chapter.

4.2 Conceptual framework for moving object data

4.2.1 Conceptualizing object dynamics

In their article Designing visual analytics methods for massive collections of movement data, Andrienko and Andrienko [5] provide a conceptual framework for understanding movement data and related patterns. This framework is particularly suited for iceberg movement exploration and we will therefore review some of its main concepts.[1] We will start by "considering the structure and properties of movement data", which leads to two different approaches to the dynamics inherent to moving objects. Then, we will "define potentially significant types of patterns" that we might encounter for example in our visualization of iceberg movement.

The dynamics inherent to the movement of multiple entities can be considered from two different perspectives. On the one hand, one can conceptualize the overall dynamic behavior as a composite of all the individual dynamics and on the other hand, the overall dynamic behavior can be seen a time-series of characteristics describing all objects. As the authors state, these two perspectives "are essentially different". Different names are therefore given: when the analyst focuses on individual trajectories, they talk about *individual movement behavior* (IMB) and when he focuses on snapshots in time of collective characteristics, they talk about *momentary collective behavior* (MCB). Going from these two different perspectives to a "holistic view of the movement characteristics of multiple entities", they call the overall behavior *dynamic collective behavior* (DCB). Lets see how the authors define these terms...

Individual movement behavior or IMB is formed of "the changes in position and other movement characteristics of an entity over time." An IMB has characteristics such as "the path, or trajectory, travelled in space; the distance travelled; the movement vector; and the variation of speed and direction." Figure 4.1 shows a conceptual representation of the notion of IMB for a single entity.

Momentary collective behavior or MCB represent the "movement characteristics of a set of entities at some single time moment." These char-

[1] Dodge and others [16] attempt to develop a generic framework or *taxonomy* for patterns in moving object data. However, because their framework aims at a high level of generality capable of integrating all types of movement data (including eye-movement!), it is less well applicable to our topic.

Figure 4.1: Illustration of the notion of individual movement behavior (IMB) (left) and momentary collective behavior (MCB) (right) (adpted from (5))

acteristics comprise "the distribution of the entities in space, the spatial variation of the derivative movement characteristics, and the statistical distribution of the derivative characteristics over the set of entities."

Dynamic collective behavior or DCB introduces to "a holistic view of the movement characteristics of multiple entities over a certain time period." DCBs can be the result of viewing an IMB over a set of objects or of viewing an MCB over time.

In section 2.4.2, we criticized the use made of the word *behavior* by Andrienko and Andrienko [4] to describe something static. This criticism may apply partly here again. They use this term here again in *momentary collective behavior* to address, among other "movement characteristics" the "distribution of entities in space". The other *movement characteristics* mentioned are "derivative movement characteristics". The authors give the following examples of such items "speed, direction, acceleration, turn, and so on." The class MCB comprises these characteristics and *distribution*.[2] The inclusion of the latter in a behavior class might be criticized in the same way as we did earlier because *distribution* is by definition static. However, the 'M' of MCB means *momentary*, this usage of terms can hardly be criticized. To keep things clear, we propose to adopt the following rule for our framework. When we use the class MCB to describe a distribution, we will add the suffix '-D'. MCB-D thus means *momentary collective behavior of type distribution*. For the other types of MCBs, the term can be used indiscriminately.

Now that we have solved this terminology problem, we can very simply explain how the MCB concept can be used. The MCBs are used to "find and measure similarities and differences between MCBs at different time moments or between MCBs of different groups of entities." In the case of MCB-Ds, we therefore might compare the positions (one of the momentary collective behaviors) of a set of objects forming a cluster with the positions of another analog set of objects – or with the same set of objects but at a different time.

[2] Andrienko and Andrienko state that very often, because of the frequent large size of datasets, summary representations of these characteristics will be shown.

4.2.2 Pattern types for moving object phenomena

As established earlier, a pattern is a description of a behavior. While he pursues a visualization task, "general pattern types exist [in the mind of an analyst] as mental schemata." Furthermore, Andrienko and Andrienko argue that, since "the analyst looks for constructs that can be associated with known pattern types," "these schemata drive the process of visual data analysis". Once such a construct is detected, the analyst can further study it using the same or different techniques.

Andrienko and Andrienko point out to two different types of patterns: patterns describing movement characteristics inherent to the data and patterns describing a connection between movement behaviors and properties of external phenomena. The first type of pattern is called *descriptive* because it describes the movement of the objects. The second type of pattern is called connectional because it attempts to explain the behavior movement by connecting it to other phenomena. Different types of patterns can be detected when visualizing the dynamics of a phenomenon. The authors of our article identify four *generic pattern types*: "similarity, difference, arrangement and summary".

Descriptive patterns from IMBs

"The behavior of the IMB over the set of entities can be described by means of similarity and difference patterns, that is, as groups of entities having similar IMBs that differ from the IMBs of other groups of entities." For instance, we might have a group of objects that are immobile in an area while another group of objects pass by the same region. Within the two groups, we find similar *individual movement behavior* while between the groups, the IMBs are different. We therefore have different *dynamic collective behaviors*. Here below is a list of the different ways in which IMBs can be similar:

- Similarity of overall characteristics
- Co-location in space
- Synchronization in time
- Co-incidence in space and time

Descriptive patterns from MCBs

Now regarding *dynamic collective behavior* from the perspective of MCBs, besides similarities and differences, the arrangement pattern type can also be encountered. In Andrienko and Andrienko's framework, *distribution* is an instantiation of the general pattern type *arrangement*. They for example consider "an increase in the number of entities in some part of the space and a decrease in other parts" an instantiation of the arrangement pattern. They list and give

definitions of eight different pattern types that can be observed while studying MCBs. In parenthesis figures the general pattern type that is related:

1. Constancy (similarity): the MCB is the same or changes insignificantly during a time interval.
2. Change (difference): the MCB changes significantly in a given time interval.
3. Trend (arrangement): consistent change in the MCB during a time interval.
4. Fluctuation (arrangement): irregular changes in the MCB during a time interval.
5. Pattern change or pattern difference (difference): the behavior of the MCB during a first interval differs from that during a second interval.
6. Repetition (similarity): occurrences of the same patterns of types 1, 3 or 4 or the same pattern sequences at different types of intervals.
7. Periodicity, or regular repetition (similarity and arrangement): occurrences of the same patterns or pattern sequences at regularly spaced time intervals.
8. Symmetry (similarity and arrangement): opposite trends or pattern sequences where the same patterns are arranged in opposite orders.[3]

For visualization purposes, several of these pattern types are of interest. Besides the basic *similarity* and *difference*, we mentioned earlier that we were particularly interested in detecting and describing the pattern types that are here termed *trends* and *periodicity*. An example of periodicity pattern can be found in appendix C.

Connectional patterns

Andrienko and Andrienko also provide a framework for the description of connectional patterns and make use of the terms “correlation, influence and structure.” Although the authors use the term *correlation* in a broader sense than the usual statistical correlation term, the two first terms are commonly understood. The last, *structure*, is defined as “the composition of a complex behavior from simpler ones.”

But what are the DCBs related to? “Properties of space, time, entities, external phenomena, and events” can be advanced to explain the dynamics of moving objects. Among properties of time, cyclicity is the one we will be most interested in. As for external phenomena, environmental influences may be advanced to explain a particular behavior pattern of a given moving object.

[3] List adapted from [5].

4.2.3 Visual techniques for movement, and data reduction

The authors of our article review the visualization techniques available to study DCBs from IMBs and MCBs. For the case of IMBs, the space-time-cube (STC) and animated displays are the two techniques available.

With the space-time cube, it is in effect possible to visualize space-time paths to "estimate the positions, speeds, directions, and other movement characteristics at different times" for IMBs. But the authors caution that the benefits of this technique "fade away with an increase in the number of moving entities, the length of the time period, or the geometric complexity of the trajectories." The number 10 is advanced as an already high number of trajectories to be visualized in a space-time cube and this technique is therefore not to be recommended for iceberg trajectory studies. We want to add that the STC cannot be used to visualize any other type of dynamic phenomenon than trajectories and that therefore it cannot be used to explain IMBs by connecting them to other types of phenomena.

To visualize DCBs from MCBs, the use of the STC and a "movable plan" [38] might be useful although the effectiveness of this new technique still needs to make its proof. More commonly, small-multiples and map animation can be used to visualize MCBs. The two latter techniques, as we said previously, can well be combined.

Because modern datasets of moving objects often comprise large numbers of objects and long trajectories, data analysts regularly encounter the need to reduce the number of items visualized on-screen. Andrienko and Andrienko identify "aggregation, filtering and clustering" as potential approaches to effectuate this data reduction. They review these approaches specifically for moving object data. Reviewing these approaches is beyond the scope of the present research on visualization.

4.3 Case-study: Antarctic Iceberg movement visualization

4.3.1 Iceberg formation and fields of application

Icebergs are the result of the separation – or calving – of big masses of ice from an ice-shelf. In Antarctica, the ice-shelf surmounts the Southern continent. The ice, resulting mainly from snow compaction, flows in a "downhill manner" due to gravity. Ice-tongs are the most important iceberg calving regions and are located in specific areas. It is important to distinguish icebergs from sea-ice. While icebergs come from the ice-shelf, sea-ice is merely frozen ocean water. After their calving, icebergs travel mainly following ocean currents [9].

Several fields of application study iceberg. Glaciologists study icebergs to get knowledge about ice dynamics and the relation between iceberg life-cycles to

other phenomena. Climatologists are interested in iceberg numbers and life-cycles as climate indicators (past, present and future). Oceanographers are interested in icebergs because they reflect ocean dynamics and affect the biosphere by shading areas where they get grounded. Navigation and petrol companies need to be aware of iceberg positions and dynamics because of the threat they represent for ships and mining-platforms. In addition, as the shortage of drinking water is predicted to be the biggest challenge humanity will have to face in the future, iceberg towing has been pointed out as one of the ways we could provision ourselves with clear water. Finally, because of the wide range of interests related to icebergs, because of the high level of importance iceberg studies nowadays and because of the special equipment necessary to study icebergs, multi-disciplinary scientific research teams exist to address issues relevant to several application fields. We will call the members of such teams emphiceberg specialists.

After presenting the iceberg dataset that we have available, we intend to introduce a use-case that should further be analyzed to inform system design. From an actor perspective, there are at least as many use-cases as there are fields of application. The final system should address one or several of the requirements set belonging to these different fields of application. After reviewing what visualization tasks can be solved by animated mapping, we will decide which of these use-case(s) we will try and develop our visualization environment for.

4.3.2 The NIC Antarctic Iceberg dataset

Two institutes started recording Antarctic iceberg positions in the late 1970s, the (American) National Ice Center (NIC) and the Brigham Young University (BUY). The data is made freely available by the NIC [49] and thus offers more than 30 years of iceberg positions as well as other characteristics. The data was recorded by the two institutions using a variety of remote sensing platforms and sensors.[4] Icebergs are detected by backscatter contrast and systematically named. The NIC Web site provides images of each identified iceberg (see for example [50]. Figure 4.2 shows for example (among others) iceberg 'A-38' and another iceberg at a time shortly preceding its calving from the ice-shelf. Only very large icebergs can consistently be tracked and the dataset therefore only contains icebergs whose sizes exceed the range of 10 nautical miles (18.5 kilometers). We could have feared that such a dataset would not be representative of the whole Antarctica iceberg population. However, Jansen and colleagues [32] state that half of the ice calved off the Antarctic ice-shelf is due to the most massive icebergs, i.e., with a "major axis greater than 28km." Therefore, we argue that the population of icebergs with a major axis of 18.5km or larger can be assumed to be fairly representative of the whole population.

[4] Ballantines and Long [9] inform us that the NIC uses an infrared imager, a radiometer (Advanced Very High Resolution Radiometer (AVHRR), Synthetic Aperture Radar (Radarsat SAR) but most of all, it uses scatterometer sensors (Operational Linescan System (OLS). The NIC also "obtains iceberg positions provided by BYU who use SCP enhanced resolution images derived from the SeaWinds scatterometer on board the satellite QuikSCAT."

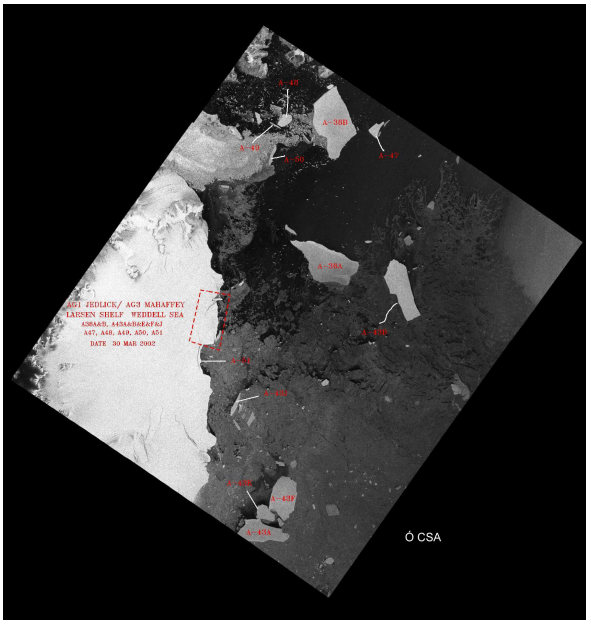


Figure 4.2: Iceberg identification from satellite images (Source: (49))

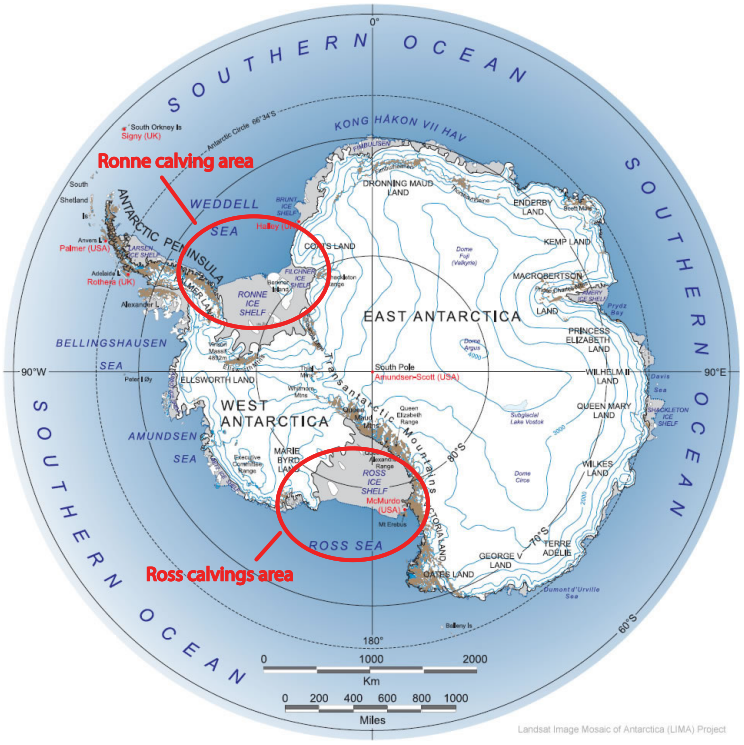


Figure 4.3: Map of Antarctica showing the areas where major calving events occurred in 1999 and 2000 (adapted from (22).

The icebergs with the largest major axis in the dataset is 172 km long and the one with the biggest surface covers 9274 km^2.

The dataset contains the following information: an arbitrary number identifier, an iceberg ID, a time-stamp of the moment that particular tuple of information was recorded, x-y positions in lat/long, the size of the iceberg (given both in approximate short and long axes and in area) and finally the satellite that recorded that particular iceberg instance.

The temporal aspects of the iceberg dataset that are most interesting for our purpose are the longevity of the icebergs, the temporal resolution and the temporal precision at which the data was recorded. The lives of icebergs commonly last from a few years to 15 or even 20 years. The temporal resolution of the data resulting from the NIC and BYU combined effort is variable. The laps between two iceberg instances commonly goes from 1 or 2 days to 15 days, with regular cases of up to 50 days. This temporal irregularity and the general low temporal resolution might limit the results we can expect to get when exploring the objects' dynamics. The temporal precision of the recordings is unfortunately low as well. Only the day at which the icebergs were spotted are entered, no hours or any more precise time measurements. We assume that it would be possible for the data providers to enter more precise time-stamps and hope perhaps to get such data at some point.[5] The temporal irregularity and the generally low temporal resolution as well as the low temporal precision of the data records all impede the exploration of movement dynamics. For observing distributions, this seems to be less problematic.

We want to give our special thanks to our colleague and friend Hoa Nguyen Thi Phuong [57] for offering to share with us the work she did cleaning the dataset. In effect, the original NIC dataset contained many duplicate recordings that were not trivial to trace. Errors of other types were also removed.

The NIC dataset has been used in the past by researchers with various objectives. Stephen and Long's [74] studied the life-cycle of a single iceberg. Ballantyne and Long [9] analyzed the increase in calving events happening from 1976 to 2001. They mainly try to determine the reasons behind the increase in iceberg numbers that occurred in the years preceding their study. They establish that the biggest part of the increase can be explained by major calving events that occurred "from the Ronne and Ross Ice Shelves" (see fig. 4.3 for map) but that improvements in recording techniques is also responsible for part of the increase. In their study, the authors also report on the movements of the icebergs after the time they calve. It appears that the ocean currents around Antarctica are fairly complex. Time constraints unfortunately do not allow us to review documentation on this matter.

The dataset represents a unique source of information for many research fields and need for more effective visualization techniques has been identified [75]. In

[5] We have attempted to contact the data providers at NIC but up to date, we have not received an answer.

the next section, we will review some of the challenges that research on icebergs is facing and we will point-out those which animation could help to address.

4.3.3 Iceberg-visualization tasks using animation

We identify three approaches related to the dynamics of icebergs in which visualization techniques could be applied. While the two first approaches can lead to the discovery of descriptive patterns, the last can lead to discovering connectional patterns.

The first approach is related to the distribution of the population of icebergs in space and its evolution through time. It is related to existential changes (appearance and disappearance of icebergs) and to changes in collective distribution. The second is related to the dynamics of icebergs, to 'how they move' in space (change in absolute position) and 'how they move' with regard to each other (change in relative position). The third approach comprises attempts to explain iceberg behavior by correlating it to external phenomena. It is split into two topics: explaining calvings and disappearances, and explaining iceberg movement.

While treating these three approaches, we will put in between '( )' requirement with strange names which will help us in the next steps, analysis of these requirements as well as system evaluation.

Evolution of iceberg distribution

Two iceberg behavior types influence the distribution of the population of icebergs: the first is icebergs appearances and disappearances and the second is iceberg movement. Iceberg appearances bear the name of calvings. Two different types of calving exist: calving from the ice-shelf - in our case the glacier tongs of Antarctica and calving resulting from the split of an iceberg into two or more separate entities. The 'disappearance' of an iceberg from the database usually indicates that its size became too small for it to be tracked but disappearance can also indicate that the iceberg was lost by the tracking team for various reasons.

The difference between iceberg appearance and iceberg disappearance logically affects the total number of icebergs present in the ocean. Climatologists are interested in the evolution of the total number of icebergs as an indicator of climate-warming. Animated mapping is less effective than other methods to study the evolution of the overall number of icebergs. However, it can be useful to study where and when the calvings and disappearances occur. However, in a display cluttered with large numbers of moving icebergs represented, such existential changes might not be easily noticed by the animated map reader. It might therefore be useful to emphasize these existential change events (requirement 'exist').

Apart from studying the appearances and disappearances, animation can help to visualize where and when one can find clusters of icebergs. Where and when do we have high concentrations of icebergs (requirement 'highConc')? This question is of particular interest for oceanographers and also for navigation companies. Are these clusters formed at regular times of the year from one year to the next? Are there times of the year with high/low concentrations of icebergs and where are these high/low concentrations localized (requirement 'timesYear')? Furthermore, once clusters have been identified, one might want to study how these clusters evolve in time (focus on the region) (requirement 'evolTclust')? The evolution of the distribution of icebergs typically requires a *momentary collective behaviors* (MCB) approach.

In addition, climatologists, navigation companies and iceberg specialists can be interested in comparing the distributions of icebergs between two or several years. This can be done in several ways. An obvious way would be to aggregate the data for each year and provide summary representations of that aggregated data. Another way involves a special animation technique that Slocum and others [72] term *brushing*. The technique is said to "involve choosing a subset of a time series," for instance, the 21st of December of each year composing the dataset (requirement 'compareYears'). With animation applied to it, the viewer can rapidly examine the trends, variations and regularities existing in the successive years. In fact, this can be done in several two ways, choosing, like in our example, to visualize only one day a year or by 'summarizing' the data over a given period (e.g., one the month of December) for every year.

Motion dynamics of icebergs

Several fields of application can gain from increased knowledge about how icebergs move. Questions such as the following may be asked on the trajectories and movement of single icebergs:

- What is the complete trajectory of an iceberg during its 'lifetime'? How much does one iceberg travel during one year (absolute movement)?
- Are their trajectories regular or are they chaotic?
- What are the dynamics or variations in speed in the movement of an iceberg? (requirement 'dynSingle')

The two first questions do not necessitate animation to be answered; static mapping would be well suited. For the third question, animation will be useful although animated mapping is not the best method to assess speed of moving objects.

Animation becomes useful to assess motion dynamics within space-time clusters of icebergs. Once space-time clusters are identified, we can attempt to answer the following questions:

- Do all icebergs in a space-time cluster follow similar trajectories? Do space-time clusters of icebergs split, seeing one group going one direction and another group going another direction? (requirement 'trajClust')
- How do icebergs move with regards to each other (relative movement)? Are their speeds similar or different? (requirement 'relMvm')

These questions might seem similar to those we posed when introducing distribution evolutions. However, the approach needed is quite different. While in the previous section we were interested in overall distributions, here we are interested in the dynamics of identified icebergs, whether they are alone in space-time or whether they evolve in a group. The approach needed here is that of *individual movement behavior* (IMB). Animation appears to be a potentially useful visualization method to answer the second set of questions posed here.

Two severe data limitation unfortunately apply for the visualization of motion dynamics. As we mentioned earlier, the temporal precision of the recordings is one day. This implies that some of the dynamics patterns that we might detect may be the result of the imprecision of the temporal attribute of our data.

Exploring iceberg calvings and disappearances

The two categories of iceberg behavior identified, i.e., iceberg appearance/disappearance and iceberg movement are both largely related to external phenomena. What are the external phenomena that influence these behavior and what patterns do these influences follow?

Various internal and external parameters play roles in the calving and disappearance of icebergs. A wide range of factors have been identified in literature to explain both the calving and reduction in size of icebergs. Grossly, the set of factors advanced can be considered to be similar for both phenomena because both calving and disappearance are the results of processes such as weakening, melting and erosion of the ice. The set of factors advanced include the following items [9, **?**, 76]:

- Ocean currents
- Wind currents
- Iceberg to iceberg or iceberg to ice-shelf friction
- Basel friction
- Tidal currents
- Water salinity
- Water temperature
- Solar irradiation
- Air temperature

Research is still ongoing to assess the weight of these factors on iceberg calving and size reduction. Our concern here is whether vector animations can be of use to researchers to further model these weights. Raster-based animations have been applied to illustrate processes such as of calving, iceberg-ice-shelf friction or the effect of tidal currents. [51, 73] However, we believe that that, as exploratory tools and not merely as illustrative tools, map animations are essentially useful to visualize phenomena in which both the spatial and temporal dimensions play an important role. To study the weight of the factors listed above, we want to argue that animation alone is not an effective technique. In effect, most of these factors have a mild short term effect and it is only through long time intervals that their effects can be measured. Since in animation, change detection depends on human memory capacities, we argue that this technique would not be effective. The analyst would therefore need to remember the approximate values of the factor fields that individual or even multiple icebergs traversed. Since this appears almost impossible, we propose that instead of visualizing multiple icebergs in a spatio-temporal context, a different visualization technique or even a mathematical model should be applied. As a visualization technique, we could propose that the attribute values of the factors should be visualized through time for single icebergs or small groups of icebergs (by using line-graphs or other appropriate visual techniques). The cumulated effects of the factors could therefore be apprehended following a historical approach.

However, if vector animation is not to be recommended to show the weight of factors on calving or size reduction, it may still be of use to visualize the appearances, disappearances of icebergs in their spatial and temporal contexts (requirements 'contxCalvDiss') as well as view the dynamism inherent to iceberg size variation (requirement 'dynSize'). It may also be useful for researchers attempting to identify icebergs whose calving, size reduction and disappearance histories would be particularly worth studying.

Explaining iceberg movement by external phenomena

Veitch and Daley [76] say that "currents were found to be the most important driving force for drift. Deep steady currents were found to be relatively important for large icebergs and wind driven currents were relatively important for smaller bergs." Two other factors however play important roles in iceberg dynamics: the height of the ocean bed and the presence of sea-ice[6]. Icebergs tend to get "grounded" when they enter in contact with the ocean bed and they get stuck when the sea-ice is thick.

Ocean currents, wind currents, sea-ice and ocean-bed height thus are the main factors which need to be taken in consideration to explain iceberg spatial dynamics. Of the four, the three first are dynamic phenomena and the last is static. It would be interesting to visualize iceberg movement congruently with

[6]Sea-ice has a different origin than icebergs. While icebergs originate from the ice-shelf, sea-ice is ocean water which froze in contact with cold air.

all four or, alternatively with one or the other (requirement 'moveRelPhen'). IMB and MCB approaches could both be applied for iceberg motion, depending on the explanatory phenomenon visualized.

4.4 Analysis of Iceberg visualization use-case

In the previous section, we presented a series of tasks and discussed whether animation could be an effective means to go about them. Figure 4.4, schematizes the system that we intend to design, the actors and related use-cases. As

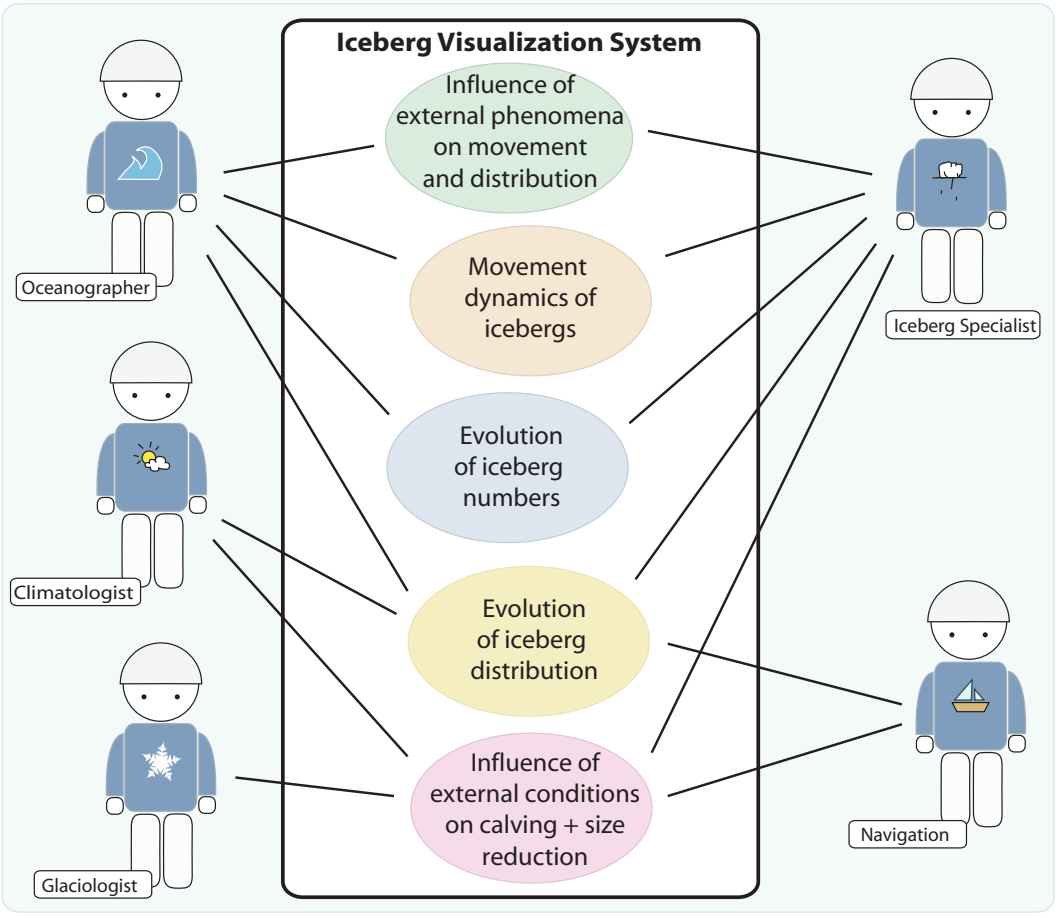


Figure 4.4: Use-cases for iceberg visualization involving actors from five different fields of application

we have seen, the use of animation for iceberg visualization tasks can potentially interest specialists from all the application fields we listed. We therefore propose that our prototype will serve a variety of users from the fields of glaciology, climatology, oceanography and navigation. In fact, we saw that multidisciplinary teams for iceberg studies already exist.

The present section deals with the second workflow of UP, which is requirement analysis. Requirement analysis is to be accomplished by focusing on system components destined to satisfy the requirements. The analysis is split into five parts. We first present a series of requirements that are generic to all animated mapping tasks.[7] Then appear four main visualization tasks and related requirement analysis. These main tasks are split-up in the same way as the iceberg visualization tasks we just saw.[8] The reader may have noticed that in the previous sections, we wrote '(requirement 'requName')' next to the tasks that could be solved with the aid of animation. As we do not intend to process the data in any complex manner for this research, tasks involving data aggregation and summarizing are not treated below. We regrouped the 'requirements' seen above according to the system functionality or component that we intend to use to address them.

Summary of generic requirements for temporal animated mapping

1. **Visualize objects:** The objects need to be visualized.
   → *Apply proper symbolization for icebergs, i.e., point symbol.*

2. **Orientation in space:** The user needs to have some idea of distances and directions that the icebergs are moving.
   → *Provide a representation of the spatial context in which the objects evolve. For the iceberg case-study, we will include a map of Antarctica and possibly parallels and meridians.*

3. **Zooming:** The user may want to zoom on a particular region.
   → *Provide a zooming mechanism.*

4. **Specify time interval:** The user needs to be able to choose the time interval of the data he will visualize.
   → *Provide a mechanism to specify the temporal extent of the animations.*

5. **Orientation in time:** The user needs to be able to orient himself in time.
   → *Provide temporal legends digital clock, time-bar and cyclic temporal legend.*

6. **Speed control:** The user needs to be able to control the rate-of-change of the animations.
   → *Provide a mechanism to control the temporal scale/speed of the animations.*

7. **Interact with temporal dimension:** To explore the data and its inherent dynamics, in addition to animation, the user needs to be able to control the temporal dimension of the animations.

[7] This is a short version of the basic requirements that we presented for temporal animated mapping in Chapter 2.

[8] For some visualization requirement analysis, we may repeat the presentation of items that we already treat as basic requirements. We do this for the reader to see which tasks the basic features assist.

→ *Provide simple temporal controls such as play, pause, stop; more specific controls such as looping and an interactive time-slider.*

8. **Complement animations with small-multiples:** As small-multiples maps are good for comparing different states of the data, for sharing the information and for remembering specific states, offering a possibility to generate small-multiple maps in an animated mapping system is highly recommended.
→ *Provide a small-multiple generation functionality. The user should be able to pause the animation on any state and easily generate a 'snapshot' of that state. This snapshot should conveniently be inserted in the series of snapshots made by the user with information on the time-stamp and without unnecessary GUI items.*

Main task 1: Visualizing evolution of iceberg distribution

9. **Requirement 'existEmph':** emphasize existential changes such as calving and disappearance events.
→ *Make use of a temporally varying visual variable (such as color change) or of a dynamic visualization variable (such as the frequency of blinking applied to a point object) to emphasize the moments when icebergs appear or disappear.*

10. **Requirement 'highConc':** help viewer to know where and when high/low concentrations of icebergs occur (distribution).
→ *The 'where' problem is taken care of by the spatial representation. Since we are interested in collective behavior and not really in individual trajectories, the animations can be simple reflections of the point position data in the database. The icebergs can remain represented by simple point symbols, no need to view trajectories, no need for interpolation.*

11. **Requirement 'evolClustT':** study evolution of clusters through time.
→ *Animation can be applied and especially controlled with a time-slider to visualize the evolution of the cluster shapes and positions.*

12. **Requirement 'timeYear':** help the viewer to relate the images displayed to times of the year.
→ *The when problem calls for assistance from temporal legends. In particular, to relate the moments to times of the year, a cyclic temporal legend is recommended. Digital clock and time-bar should be included as well.*

13. **Requirement 'compareYears':** compare two or several years of data by focusing on chosen moments of the year.
→ *The request for an animation using the technique of temporal brushing should specify the subset or periodicity of the data that should be rendered. Animation and especially time-slider control can then be used to explore the distributions.*

Main task 2: Visualizing Motion dynamics of icebergs

14. **Requirements 'dynSingle', 'trajClust' and 'relMove':** study trajectory dynamics for individual icebergs and icebergs as parts of groups; study relative dynamics between icebergs (relative speeds, converging or diverging paths).
→ All these requirements can be provided a solution for by two features added to simple animation of the data: interpolation[9] *and dynamic path mapping. Both have the same objective, reinforce the ability of the viewer to see the movement as continuous instead of a map with points changing positions.*

Main task 3: Explaining iceberg calving/disappearance by external phenomena

15. **Requirements 'contxCalvDiss' and 'dynSize':** view spatio-temporal contexts of calvings and disappearances as well as the dynamics of their size reduction.
→ Viewing calvings and disappearances is taken care of already above.
→ Visualizing the evolution of the sizes of icebergs can be achieved by making use of a traditional cartographic visual variable changing through time. Size seems like the most appropriate might be applied instead/as well because they might be easier to visually grasp.

Main task 4: Explaining iceberg movement by external phenomena

16. **Requirement 'moveRelPhen':** visualizing iceberg movement in relation to four external phenomena (ocean currents, wind currents, sea-ice and ocean bed height).
→ Provide appropriate dynamic or non dynamic representation of the explanatory phenomena.

4.5 Summary

To conclude this chapter, we will shortly sum up what we have seen. We first introduced a framework for visual analytics of moving objects. We then presented a case-study on Antarctic icebergs which we treat as a use-case for the development of an animated mapping prototype specialized for iceberg movement visualization. We identified a set of requirements from a visualization point of view and further analyzed these from a system perspective.

The most useful and specific functionalities for an animated mapping system are the following: For the user to orient himself in time, (a) temporal legends

[9] We will treat the problems related to the use of interpolation in the next chapter.

are necessary. Furthermore, for him to be able to navigate the temporal dimension, the most useful item is a (b) temporal slider. (c) A looping function may also be of use. Several types of spatial animation were identified as useful: (d) Simple animation of points reflecting the data are used to study the evolution of the overall distribution (MCB-D). (e) Animations using temporal brushing is applied to compare the distributions of the objects between different years. (f) Interpolated animations with dynamic animation paths are used to gain knowledge on the dynamics of individual icebergs and the relative dynamics between two icebergs or two groups of icebergs. (g) In addition, animation applied to traditional cartographic visual variables such as size can be used to study the dynamics of non positional attributes of icebergs. Finally, to attract the viewer's attention on special events such as calvings or disappearances of icebergs, (h) special use of dynamic visualization variables may be applied.

Despite a variety of animation types and differences in the additional visualization features proposed, the resulting visualizations all follow the same simple principle ... The specified animations are loaded along with appropriate temporal legends. The animations can be played or controlled using the temporal slider. The orientation, control and animation requirements identified in the present chapter will be used to inform the next step in our system development, *system design*.

Important data limitations related to the temporal characteristics of the dataset were identified. These limitations might limit the knowledge that can be one gain from iceberg motion dynamics visualizations.

Chapter 5

Animated mapping visualization system design

5.1 Introduction

The function of the present chapter is triple. Firstly, it offers solutions to our main objective of combining the WMS Time Dimension framework with an animatable vector graphics format for the internet. Secondly, it describes the user interface and the functionalities of our visualization environment, which should generate interactively controllable animated maps for the exploration of moving object data. Thirdly and as a consequence of the two prior, it will serve to instruct the implementation steps of our prototype.

The proposed system should do three things. It should enable storage of the data in a database, receive requests from a Web-client and give the appropriate responses. The behavior of the system concerning request reception and responses should be as follows: (a) Firstly, the system should receive `GetCapabilities` requests and (b) return an XML `GetCapabilities` response to the Web-client. Secondly, the system should (c) receive a `GetMap` request, (d) retrieve the data needed for the request (e) transform it into a graphical output exhibiting the requested set of elements (objects, time-interval, animation type, legend, interactive functions) in an appropriate format and finally (f) respond this "map product" to the Web-client.

The structure of this chapter follows a top-down approach. We will first describe in detail the visualization possibilities that the system should offer the user. Secondly, we will describe the behavior of the animations and interactive functionalities. Thirdly, we will expose how the data should be stored and how its temporal component can be converted for use in the vector graphics animations. The fourth step is to examine the overall structure needed for our system. We will present RIMapperWMS' structure and show what elements need to be added. Finally, we will examine the parameters that should compose the `GetCapabilities` and `GetMap` requests.

5.2 Visualization options offered to the user

Figure 5.1 shows a projected layout for our visual environment. It is composed of two main areas. On the left stands the animated map display which includes temporal legends and the time-slider. On the right hand side stands a general graphical interface for the user to make visualization mode choices and to enter parameters.

To produce his first (Get)map, the user is invited to make a series of choices. These options are represented by the tick and text-prompt boxes and the temporal scale slider of the 'Visualization choices GUI' (VCGUI) of figure 5.1. We plan for the user to make his visualization choices as follows:

1. First, the user must choose (1) the visualization type. He has the choice between a mode to visualize distributions (the MCB-D mode), a mode to visualize motion dynamics (or IMBs) and a mode to compare successive years of data at particular dates (brushed animation method). If he chooses to visualize MCB-Ds, he further needs to specify whether he also desires to view existential (1.1a) and attribute (1.1b) changes.[1] If the user makes the second choice, destined to explore motion dynamics of individual and groups of objects, the system will automatically show linearly interpolated positions of objects. The user should still specify whether he wants to view the motion tracks of the objects (1.2a – i.e., dynamic paths following the entities), and, like for MCB-Ds, whether he wants to view existential (1.2b) and attribute (1.2c) changes in the data. If the user makes the third choice, i.e., brushed animations, he further needs to specify the periodicity (1.3a) of the animations and the recurrent date (1.3b) for each period (e.g., does he want to view data of one chosen day of every year?).

2. The second group of choices relates to cyclic temporal legends (digital clock and time-bar legends are made permanent items). When the user decides to include a cyclic temporal legend (2.1), he further needs to specify which version of cyclic legend he desires. (2.2), the span of the cycle (2.3 – year, week, day, or irregular such as 1 month and 10 days, etc.) and the moment of the beginning of the cycle that the cyclic legend should show (2.4).[2] For brushed animations, we consider cyclic temporal legends to be not very useful and therefore plan to deactivate this choice.

3. The third item that needs specifying is the bounding box (3). What is the spatial extent that the viewer wants to visualize. For iceberg visualization, we propose to pre-fill the boxes with values covering the whole area covered by the Antarctic Iceberg dataset.

4. The fourth set of choices the user must make relates to temporal parameters: first the temporal extent (4.1) and then the animation speed (4.2).

[1] for our case-study on icebergs, the existential changes correspond to calvings and disappearances and the attribute changes correspond to changes in size of icebergs.

[2] More explanations on these features and choices follow.

For the temporal extent, the beginning and end dates of the interval need to be entered. For the animation's speed or temporal scale, we hope to offer the user control with a slider. This speed slider would control the temporal scale of the animation. The value of the temporal scale would show in a text box that we placed within the map display because it is information and not a control in itself. Since temporal scales are not intuitive values, we also offer the user a possibility to get an idea of what it means by providing a text box showing the total display duration of the animation (also in the map display area).

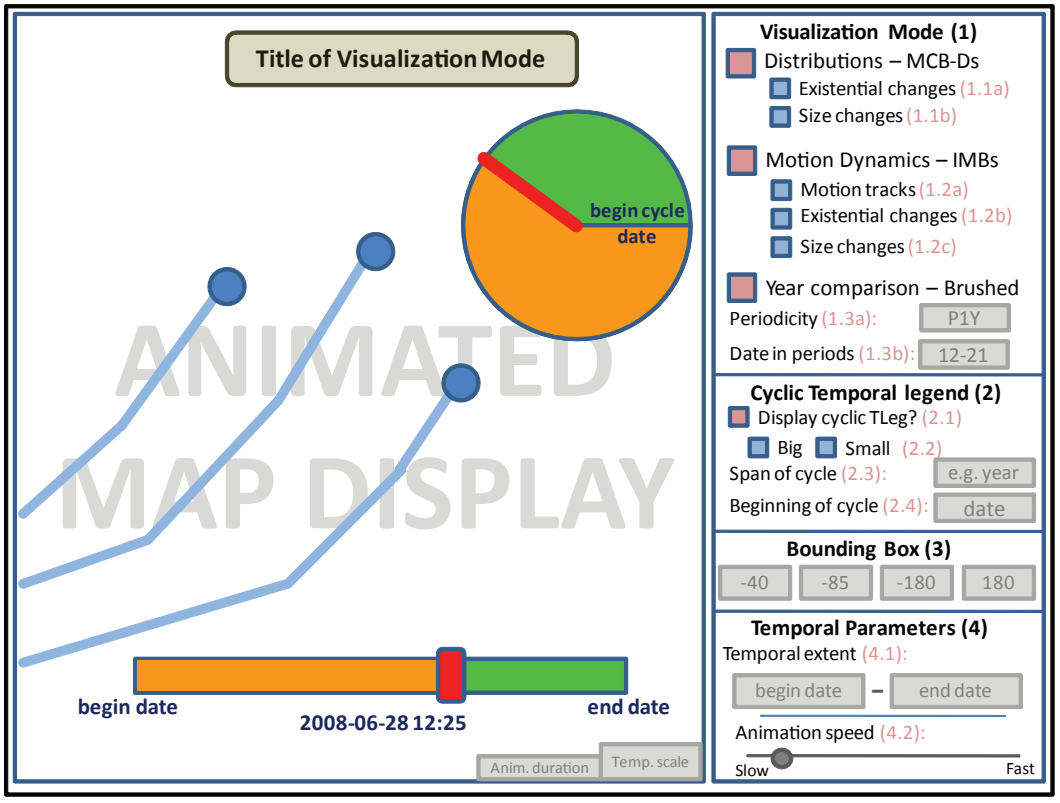


Figure 5.1: Projected user interface for animated mapping environment

Once all the necessary initial selections have been made, the GetMap request can be sent and the visualization may begin. The user will further want to change or refine his choices and he shall be able to do so by simply changing the visualization options and parameters in the VCGUI.

5.3 Description of visualization functionalities

In this section, we describe the functions offered as visualization choices to the user. This description includes graphical, animation and interactivity elements.

It is split-up into three parts. First, we describe the components that we consider to be generic for a vector-based animated mapping system. Secondly, the components of functions specific to moving object animated visualization are treated. The third set – very small – comprises those visualization elements that are specific to iceberg visualization. They are in fact mainly instantiations of generic functions described on more generic levels.

5.3.1 Generic functionalities for animated mapping

Temporal legends

The first set of generic functionalities that we treat is the temporal legends and we start with the time-bar. As can be seen in figure 5.1, we follow Harrower's [30] design of time-bar legend (seen on p.23). We consider the present vehicle-sign to be part of the time-bar because even without interactivity, it may be useful to have an attention attracting symbol representing the present of the animation. The dynamics of the time-bar are as follows: While at the beginning of the animation, the rectangle representing the past has no width, it progressively grows and covers the 'future' rectangle. The present vehicle-sign moves from left to right of the time-bar at the same speed. In addition to these graphics, time-stamp strings corresponding to the real-world moments of the beginning and end ('begin date' and 'end date' in figure 5.1 of the temporal extent should figure at the beginning and end of the time-bar.

The digital clock shall be placed under and in the middle of the time-bar. It shall show the real-world time of the animation in a dynamic way. The precision chosen for the time shall be meaningful. It might not be necessary to view seconds if we are visualizing a temporal extent of months or years.

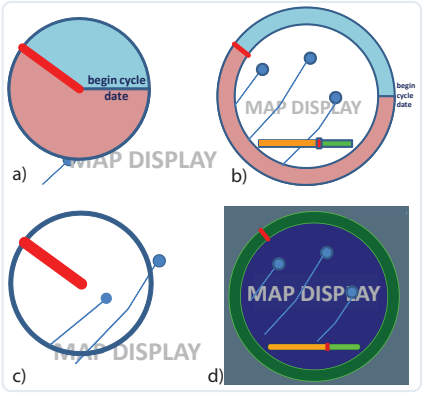


Figure 5.2: Four types of cyclic legends: a) the small pie-portion, b) the big pie-portion, c) the small hand type and d) the big hand type.

We propose four different graphical designs of cyclic temporal legend. This number might seem large but the coming considerations will justify it. These legends can be described according to two variables: their size and whether

they possess changing pie-portions or not. We start by describing those with pie-portions and then those without.

The first is as shown in figure 5.2a. It is a disc with a growing pie portion representing the past.The second, (fig. b), follows the same design but is much larger and surrounds the map display. It is based on the idea that very large cyclic legends may favor subconscious orientation within the cycle. The design of these two cyclic temporal legend (growing pie-portions) is, as far as we know, original. It is derived from that of the time-bar. At first, we were thinking of making the past and future portions of the circle the same colors as the time bar past and future areas. However, one cycle may not correspond to the temporal extent of the loaded animation and thus of the time-bar. To avoid confusing the viewer, we therefore decide to use different colors.

Because using two different color schemes for the bar and cyclic legends may overload the user of reference systems, we thought of an alternate, simplified, design. Removing the pie-charts, we only keep a “rotating hand”. These simplified legends, both of small and large types can be seen in figure 5.2c and 5.2d. A definite advantage of the small cyclic legend with only a hand (type c) is that it can be proposed without any background, which makes it less cumbersome in the map-display.

The pie-portion cyclic legends offer an advantage over simple hand legends. The user can decide at what time of the real-world time cycle the visualization cycles should start. For instance, in iceberg studies, a specialist might be interested in viewing seasons from what he considers their start to what he considers their ends. If for example, the iceberg calving season starts in December, he may want to start his cycles at that time or a little earlier. Having cycles start at another time would require him to do some unnecessary thinking. In legends a and b of our figure, the angle is of 90 degrees, which corresponds approximately to the 5th of November in a yearly cyclic legend.

It should be made clear that the hand rotates at identical speeds for both types of legends. It is only the moment of the cycle (e.g.,year) when the pie-portions start and end growing that can be set.

Time-slider

We have already abundantly discussed the time-slider. Its principle and advantages were treated in section 2.8.3. Its action is extremely simple. It must set the moment of the animation to the place the user drags the slider to. The interactive control function of the time-slider is combined to the time-bar temporal legend. In this sense, no additional graphics are added. The present vehicle-sign can be considered to be part of both the time-bar and the time-slider.

Temporal extent

To specify the temporal extent, the user must enter dates for the beginning and the end of the real-world time span he desires to visualize. It was mentioned that we could limit the user's choices of temporal extents both to make his choice easier and to simplify other settings (cyclic temporal scale). However, we defend that it is useful to offer the explorer with the freedom in his choice of the time extent. The reason is that he may be interested in the life of a particular iceberg or of a group of icebergs (e.g., all the icebergs that calved during a major calving event). Animations corresponding to the visualization choices made should be loaded for this time span.

Temporal scale/speed

For the design of our function destined to control the speed or the temporal scale of our animations, we propose to center our design on temporal scales and not speed or animation durations. Dealing with temporal scales instead of speeds has the big advantage, as explained earlier (see p.28), of being objective. Provided that an absolute zero of time is chosen, the animations loaded could very well have time-stamps corresponding to real-world times. These time-stamps can then be multiplied by the temporal scale to become time stamps in display time. We propose to control the temporal scale via a 'Tscale-slider'. Ideally, we would like to enable the user to change the speed of the animations without reloading the animations. Controlling the temporal scale as we propose, instead of controlling the durations of animations seems to be a good way of achieving this. In addition, it also offers the advantage of not needing to recalculate all the individual animation durations.

As announced, to deal with the fact that temporal scale values are not intuitive, we propose two solutions that we shall both adopt. First, the Tscale-slider should not have Tscales as indicative values but just the adjectives 'slow' and 'fast'. Secondly, to inform the user on what a temporal scale value corresponds to in his visualization experience, an information box tells the user in what display-duration the real-world time interval will be viewed.

5.3.2 Moving object specific functionalities

The following paragraphs are on visualization functions specific to moving object data. The elements treated could be grouped in four design categories: (1) the functions of the general GUI for the moving object visualization platform, (2) the graphical representations of objects, (3) animation types and (4) additional animated visualization elements (e.g., dynamic attributes, showing real/interpolated states, . . .).

GUI for visualization choices

As shown in figure 5.1, the 'Visualization choices GUI' contains check-boxes that the user can click, prompt text-boxes that the user can fill and a speed-slider to manipulate. The most fundamental choices (visualization types, temporal extent and bounding-box) should set the parameters of the `GetMap` request. Once sent, this request triggers high-level programs which serialize the data and the graphical user interface elements. The less fundamental choices trigger interactive functions client-side, some of which may have powerful visualization effects.

Graphical representation of objects

Logically, for visualizing moving objects data, the main symbol needed is one representing the objects themselves. Typically, point objects are represented by symbols such as discs, squares, triangles or any other more meaningful shape. Size, color, stroke of objects can be chosen for visualization or meaning-related purposes.

Animations to show the distribution of objects – MCB-D

As we saw, three main animation visualization types were proposed: plain animation of the data, brushed animations and animations showing the motion dynamics. We will start by presenting the functions underlying the first.

We assume the data used for our visualizations is in the following format: for each recorded position of the objects, a tupple is stored showing x-y positions and a t temporal value. For plain animations of the data, meant to show the distributions of objects, we adopt the MCB-D visualization mode explained previously. The objects are displayed in a given each position from the date of the time-stamp corresponding to that position until the date of the following time-stamp (and thus the following position).

Animations for motion dynamics visualization – IMBs

To visualize motion dynamics, we propose to integrate two design elements. The first is to interpolate the positions of the objects in space-time. When the animations are run, the objects are shown in successive positions, some of which correspond to real data (space-time) positions and some are the result of interpolation. The second is to show the dynamic tracks of the objects. The objects appear to be followed by their tracks.

Because we interpolate positions, the user needs to be informed that not all positions correspond to true records of object positions. We propose to inform the user about this in the two following manner: As the object passes real data

positions, these positions are marked by a dot. These discrete dots – in the same color as the dynamic tracks – remain visible after the passage of the object.

Comparing distributions at regular time periods - Brushed animations

The principle of brushed animations has already been explained. While in the above type of animation, all the data positions are loaded, in this type of animation only a subset of the positions is taken. The subset is defined (in addition to the temporal extent), by the periodicity and by the repetitive date in each period that should be visualized. For instance, the repetitive date of the 5th of February (Middle of the summer in Southern hemisphere) of each year (periodicity) should be loaded for a time extent going from 1976 to 2008.

Showing attribute changes

As we saw, moving object spatio-temporal data may also contain attributes describing evolving qualities of the objects. In the iceberg case for example, their size-changes are recorded. We propose to animate appropriate traditional cartographic visual variable to represent these changes. If for example, an object goes through critical phases in which it is safe, threatened or in serious danger, the visual variable hue may be animated from green to red through yellow. The changes may be discrete or continuous.

Emphasizing existential changes

A frequent type of change observed in moving object data is existential change. As seen for icebergs, objects may appear or disappear. We propose to attract the user's attention on these happenings using dynamic visualization variables. We are thinking in particular of applying the dynamic visualization variable frequency and to blink the symbol representing the object off and on. Doing so, we should be aware that the uniformity of the temporal scale is interfered with. However, such changes might disturb the visualization if the viewer is interested in other aspects of the phenomenon. This underlines the importance of offering the possibility to switch such emphasis functions off.[3]

5.3.3 Iceberg specific visualization components

Two types of iceberg-specific visualization choices need to be made. The first set is derived from the more generic functions that we have seen above. In addition, as it would be more meaningful to visualize the icebergs in their contexts, we intend to integrate elements of their environments in our visualizations.

[3] If these effects are disturbing even when the user is trying to visualize the events they represent, animating visual variables would also be an option.

Visualization choices for icebergs

We said that a symbol needs to be chosen to represent the objects visualized. We propose to use colored discs to represent icebergs. These discs may or may not have strokes. The second choice which needs to be made relates to which visual variable to animate to represent changing iceberg sizes. We propose to animate the variable *size*. This might be done in a discrete way, classifying the sizes or in a continuous way, with or without interpolation. We propose for the choices specific to iceberg visualization not to be offered to the user as this would require our prototype to be a complete mapping application. To simplify our task, we propose that these design variables be directly entered into the system.

Visualizing iceberg environment

Most cartographic visualizations need an appropriate context. We established earlier that for the visualization of iceberg phenomena, we should include a map of Antarctica, parallels and meridians. We propose to adopt an azimuthal projection of the region centered on the continent.

In addition, it may be of interest to visualize the dynamics inherent to our data in relation with other dynamic phenomena such as ocean currents, wind currents and the presence of sea-ice as well as with constant environmental qualities such as ocean-bed heights. We hope to find time-series data to represent the three dynamic phenomena and a map of bathymetric heights. If we didn't find time-series data for them, if its quality is not sufficient or if data on these dynamic phenomena were too difficult to animate, we hope that static representations of some of the dynamic phenomena (ocean and wind currents – if they are sufficiently regular throughout the year) and temporal general knowledge on the presence of sea-ice (according to seasons) might be useful fall-backs.

5.4 How should the data be stored, converted and retrieved?

5.4.1 Storage of spatio-temporal data

'How should the spatio-temporal data be stored?', 'How should it be converted for use within the client-side animated graphics format?' and 'How should the system respond to a temporal request or in other terms, for a specified temporal extent, which spatio-temporal data objects should be retrieved from the database?' are the main questions we will offer solutions to here. Because the result of the time-stamps conversions will affect the way we store the data, we will start by treating the conversions steps. In addition, as we just saw, we intend to congruently visualize additional data. How should this data be com-

bined with the iceberg data? Would it be possible to use another WMS engine in order to demonstrate the interoperability of our system?

5.4.2 Steps to convert the time component of the data

Both client-side vector graphics format that we studied (i.e., Flash and SVG) have an internal clock running in which display-time elapses. All changes happening in the animations happen in this display-time. Typically, seconds is the main time unit used and the animations start at time 0 seconds. On the data side, we have real-world time-stamps. In addition, these time-stamps are in the ISO 8601 extended format which means that they are in the form of a single string including different time units (e.g., 2009-01-12T14:50:58Z). In the iceberg case, these time units include years, months and days. A conversion is necessary to go from ISO 8601 to a time format that can be integrated in the animations referring to seconds.

The conversion we propose is composed of the following three steps. These are illustrated in figure 5.4).

1. **Convert time stamps to a single unit:** Convert the multi-unit time stamps (years, months, days, ...) to single unit time stamps. Typically, for modern times, this is done by calculating what the multi-string time stamp corresponds to in *seconds from the arbitrary date 1/1/1900*.
2. **Subtract the start-time value:** Since the client-side animations start with time 0 seconds, the next step is: for a given time extent, subtract the value of the starting time from all time stamps. The new time stamps are now in seconds from 1/1/1900 reduced by the start time.
3. **Multiply by temporal scale:** Finally, we multiply the new time stamps by the temporal scale. The temporal scale is chosen by the user in function of the 'speed' at which he desires to vision the animations.[4]

Only the two last steps depend on the visualization choices made by the user. They depend on the temporal extent of the animation and on the temporal scale chosen.

The first step does not need to be done at runtime and we thus propose to store the time in two formats in the database, i.e., one in ISO 8601 extended format and one in in seconds from 1/1/1900.

The database schema for storing moving object spatio-temporal data should follow OGC's specifications. For spatial attributes, the x and y coordinates of an object's position should be stored following OGC's simple feature geometry of type point. For the temporal attribute, the ISO 8601 standard should be

[4] As we saw in chapter 3, more complex time conversions may be useful to use more elaborate time manipulation functions in SVG.

Table name ->	Icebergs					
Attribute name ->	SrNr	ID	TIME_ISO	TIME_SECs1900	GEOM	AREA_KM2
Data type ->	integer	string	string	integer	wkt	integer
First tupple example ->	1	A01	1986-10-08	308457609	POINT(-56,-34.2)	3859
	2	A01	1986-10-15	308492932	POINT(-55,-32.3)	3859
	3	A01	1986-10-17	308528255	POINT(-53.7,-35)	3087
	4	A01	1986-10-20	308563578	POINT(-53.7,-35)	3087
	5	A01	1986-10-25	308598901	POINT(-53.4,-34.4)	3087
	6	A01	1986-10-29	308634224	POINT(-53.9,-33.8)	2401
	7	A01	1986-10-30	308669547	POINT(-53.3,-33)	2401
	8	A01	1986-11-08	308704870	POINT(-53.3,-32.8)	2401
	9	A01	1986-10-11	308740193	POINT(-51.7,-31.6)	823

Figure 5.3: OGC-compliant database schema for spatio-temporal point object data.

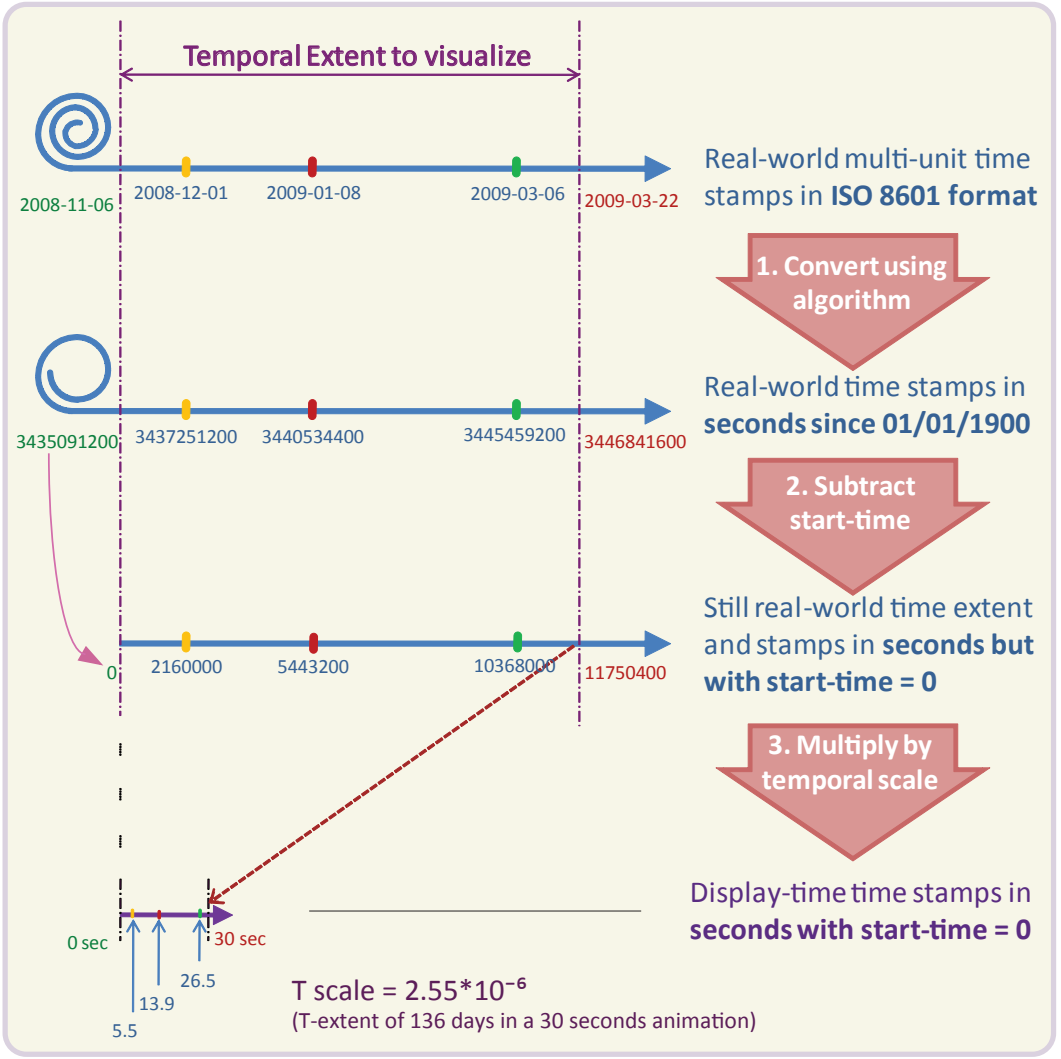


Figure 5.4: Time conversion steps from ISO 8601 to animation display-time seconds

adopted. Figure 5.3 shows what the 'Icebergs' table of our database could look like.

The second attribute (ID) holds the name given to each iceberg by NIC. The second holds the time-stamps in the ISO 8601 format. The third attribute is a conversion of the latter into seconds since 1/1/1900. The fourth attribute follows OGC's simple feature geometry for point objects.

To populate the database with the Antarctic Iceberg dataset, we will first have to convert the values of the x, y and time attributes to this specification.

5.4.3 The need for a temporal intersect

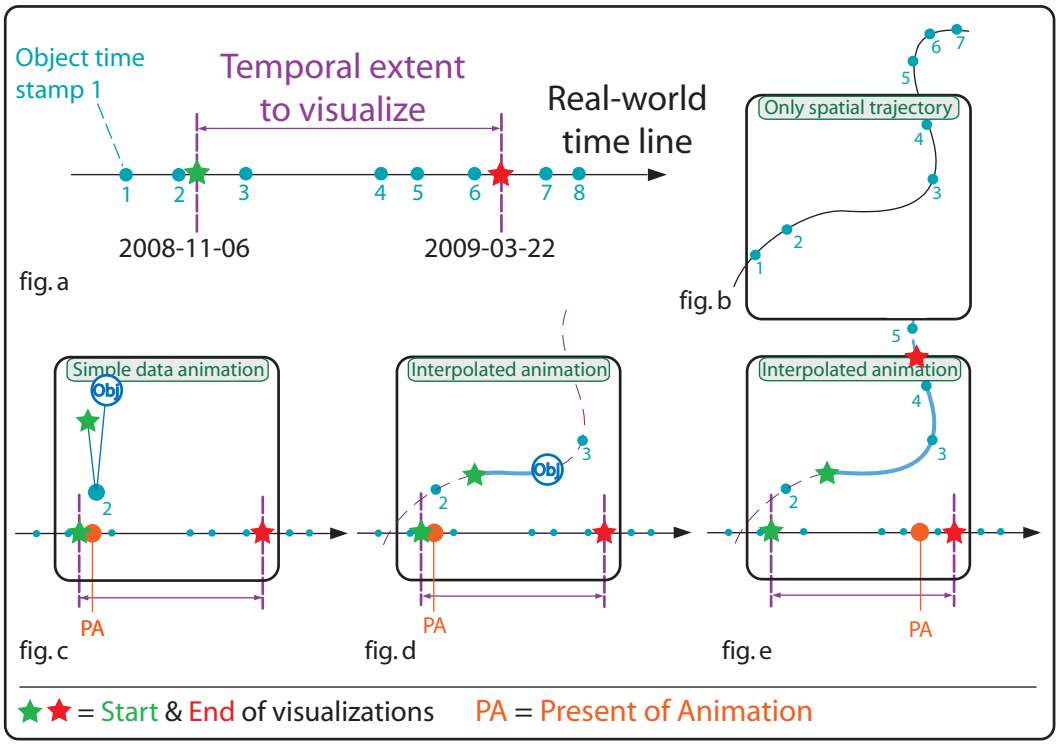
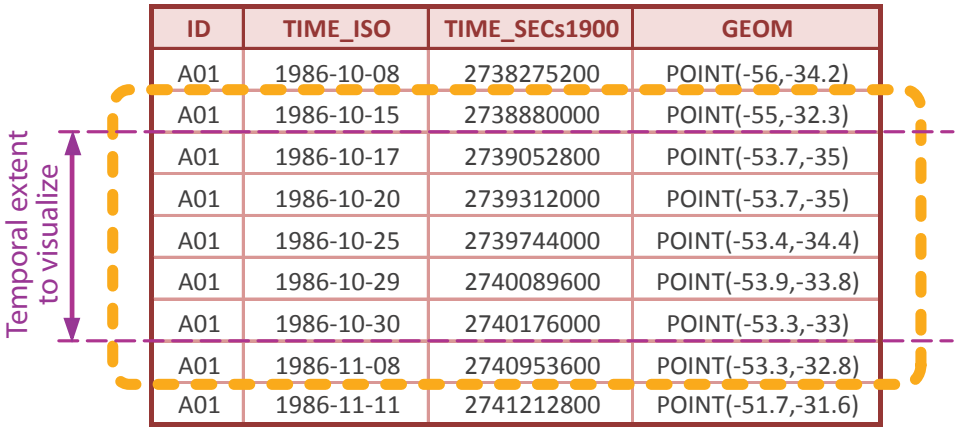


Figure 5.5: The need for a temporal intersect. In b, c, d and e, the blue points mapped (with numbered labels) show real data positions whereas the 'Obj' labels show object positions at the present of the animation.[6] The green and red stars respectively represent the start and end of the animation.

Map Server engines providing non temporal information apply a spatial-intersect between the bounding box requested by the user and the spatial distribution of the objects. Similarly, to parse a temporal request, our Map Server needs to apply a *temporal intersect* to the data.

Figure 5.5 illustrates the need for a temporal intersect. The beginnings of the animations, at display time 0 should also show states reflecting data positions

that are not within the time interval specified. In a), a real-world timeline is shown with the time stamps of one single object. Lines delimitating the temporal extent show the period that shall be visualized. Time stamps 3, 4, 5 and 6 are within the temporal extent. Figure b shows the trajectory of our object mapped without consideration for the temporal dimension (except for order). Figure c represents a stepwise animation of the data at start time. It attempts to show that the user needs to view the position of the object before the time of its first time-stamp within the temporal extent. In effect, although time-stamp 2 is not within the temporal extent, the object is shown in that position at the beginning of the animation. Otherwise, the user may think that the object did not come into existence before time-stamp 3. This also applies for



ID	TIME_ISO	TIME_SECs1900	GEOM
A01	1986-10-08	2738275200	POINT(-56,-34.2)
A01	1986-10-15	2738880000	POINT(-55,-32.3)
A01	1986-10-17	2739052800	POINT(-53.7,-35)
A01	1986-10-20	2739312000	POINT(-53.7,-35)
A01	1986-10-25	2739744000	POINT(-53.4,-34.4)
A01	1986-10-29	2740089600	POINT(-53.9,-33.8)
A01	1986-10-30	2740176000	POINT(-53.3,-33)
A01	1986-11-08	2740953600	POINT(-53.3,-32.8)
A01	1986-11-11	2741212800	POINT(-51.7,-31.6)

Figure 5.6: Applying a temporal intersect to spatio-temporal data

interpolated paths. In figure 5.5d, shortly after the beginning of the animation, the object is situated in an interpolated position between data-stamps 2 and 3 and its path has been interpolated and animated for all positions starting from the calculated position that the object is likely to have had at a time corresponding to the beginning of the animation.

The way a temporal intersect is implemented is similar to that of the spatial intersect. For each object existing during the time-extent, its time instance immediately preceding the time-extent and its time instance coming immediately after the time-extent should be retrieved (that is, if the object didn't start or end its existence during the time-extent). This principle is illustrated in figure 5.6. The orange dashed box shows the temporally selected data that we would use to build an animation for a chosen time extent (here from the 16th of October to the 3rd of November 1986).

Coming back to figure 5.5, frame e shows that although time-stamps 5 and 6 are within the temporal extent, the position of the object at those moments might not be within the spatial extent. In conclusion, to depict the positions of the objects for the requested bounding box and the requested temporal extent, both a spatial and a temporal intersect need to be applied. Although slightly complicated to realize, this shall have a positive impact on the size of the animation files loaded and thus on the memory load put upon the client.

Applying the time conversion steps presented in the previous subsection, real-world time stamps occurring before the beginning of the visualized time-extent shall receive negative values. The graphics animation engine should recognize these negative values and show appropriate positions (non interpolated as well as interpolated). The showing of states reflecting time-stamps occurring after the temporal extent can simply be prevented by a mechanism making the animation stop once the time extent has elapsed.

5.4.4 Integration of additional georeferenced products

As previously introduced, we intend to visualize the icebergs within elements of their geographical context. A map of Antarctica with parallels and meridians as well as bathymetric heights were proposed as static data. Ocean currents, wind currents and the evolution of sea-ice was proposed as additional time-series data.

To demonstrate the interoperability of our system, we hope to integrate WMS layers from another map server. However, the state of the zoom and pan functions of the present RIMapper does not trigger dynamic requests to external map servers. We hope that this feature will be implemented on time for the integration to be possible.

5.5 High level system structure

In this section, we will first present the conceptual structure of our system. The next step will be to study the structure of the existing RIMapperWMS and

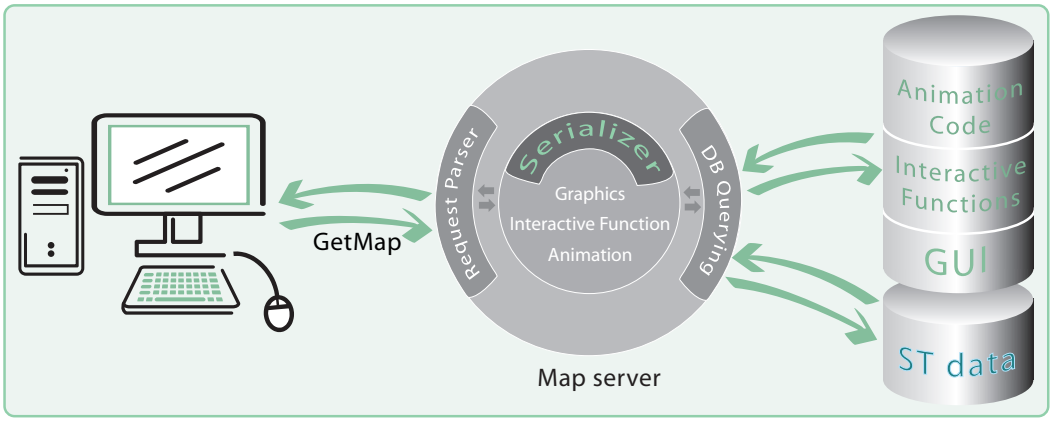


Figure 5.7: Three-tier architecture: Web-client-map server-database

assess its suitability for our purpose. In these steps, our system design will stop being generic and usable for implementation for any vector graphics output

format. Finally, we will present the ways in which RIMapperWMS needs to be transformed to deliver interactive animated maps from a database backend.

5.5.1 Conceptual representation of the system

Figure 5.7 offers a representation of the system and its components. It follows the common 'three-tire' architecture: database-processing capabilities- client. The Web client, in our case, shall be a simple Web browser capable of rendering the vector graphics animation format that we intend to implement as well as related interactive functions.

The map server is composed of three parts. The first part is triggered by the client's request. It parses the request and activates the appropriate mechanisms to render an interactive animated map product. The central component, that we call the serializer[7], once activated, retrieves the necessary information from the database and puts it together. For SVG interactive animated maps, the products resulting from this serializing are of two types: SVG graphics and animation on one side and interactive functions in ECMA-Script on the other. The final result, however, should be one single output (nested) file that can be rendered by the client. The third component is a querying engine triggered by the serializer which communicates with the database. It is responsible for translating the request into simple query language (SQL).

The database component is split into two parts. On one hand the actual data (for our prototype, iceberg data) and on the other, scripts related to the graphical user interface, the animations and the interactive functions. Although from a geographic point of view the latter group are not data, from a system design perspective, they are.

5.5.2 RIMapperWMS' present structure

Köbben [34] explains that the existing version of RIMapperWMS is made of three components:

- "A spatial database backend is used for storing both the configuration of the Web Map Services as well as the actual spatial and attribute data (...)
- A set of Java servlets and classes that respond to WMS compliant requests (...) by providing maps in SVG, with a built in GUI (...)"
- A "Web-client capable of rendering SVG to view and interact with the maps."

We represent this structure in more detail in figure 5.8. It is a three-tier architecture composed of Web client, middle-ware and database backend. To avoid

[7]Serializing is the term used in software development to describe the action of an engine that composes information from a database into an output of another type.

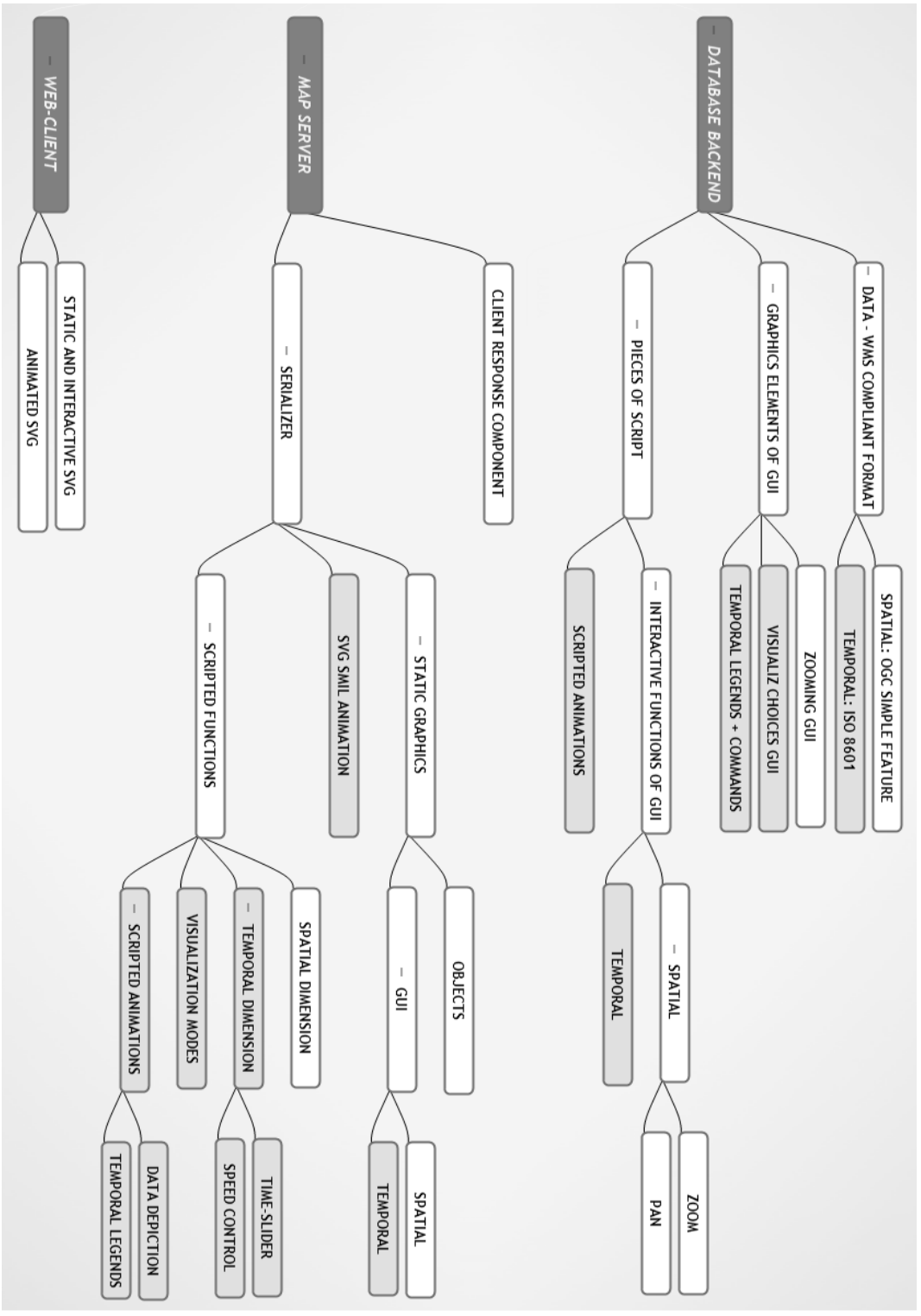


Figure 5.8: The structure of the present implementation of RIMapper (white boxes) and the proposed extensions (light grey boxes).

redundancy, the elements that we intend to add to the setup for interactive animated maps generation have also been included in the setup and appear in orange. We are now going to review the components of the present setup and their actions.

The data stored in the database backend follows OGC's simple feature recommendation. In addition, the graphics of the GUI as well as pieces of script for interactivity are also stored in the same database. The interactive functions presently available are zooming and panning, a layer switcher and mouseover attribute information retrieval. The middle-ware responds to client requests by activating the serializer. The serializer composes SVG interactive maps including a built-in GUI from the three types of 'data' in the database. Finally, the web-client is an SVG capable Web-browser

This basic structure seems to be well suited to be extended for our purpose. No changes to the structure seem necessary to store additional time components in the database, add temporal legends to the graphics, animate the graphical objects and add new interactive functions for animation control.

The behavior of the system concerning map server-client and map server-database communication are similar to what we described above. We therefore decide to build on the existing system by extending its capabilities for animations and their control.

5.5.3 Extending the structure: the birth of TimeMapperWMS

As announced, figure 5.8 contains the extensions that we need to make to the existing implementation of RIMapperWMS. Extensions need to be made to all components of the system. The reader is invited to follow these changes on the mentioned figure as we describe them.

Extensions to the database tier

On the database side, the database schema needs to be extended, in the way explained earlier to store a temporal information in the ISO 8601 extended format.

The graphics description of the GUI, as well for user choices as for interacting with the temporal dimension must be stored. A new component needs to be added to the scripts classes. Not all types of animations that we desire to render can be implemented fully using SVG's SMIL animation engine. In effect, the implementation of a cyclic temporal legend and of interpolated animated objects' tracks need scripting. In addition, the interactive functions library needs to be extended to include the behavior of: the time-slider and the visualization choices offered to the user[8].

[8] As described in the section on the user interface, this includes, among others, speed/temporal scale control.

Extensions to the middle-ware

Two sets of extensions are necessary regarding the middle-ware's serializer. While the present implementation serializes the geographic and GUI data into vector maps with built-in GUIs, the new setup shall be extended to serialize animated maps from spatio-temporal data and include appropriate GUI graphics. The second extension concerning the serializer comes from the necessity to generate a fairly large series of new interactive functions. Among these functions, the simplest are related to the user's choices and the most complex are related to the control of the animations' speed and temporal moment (time-slider).

Extended capacities of Web client

While for static vector maps, the Web client only needed to be capable of rendering static SVG, clients to be used for the new implementation need to be capable of rendering SVG SMIL animations[9]. Unfortunately, not all browsers have this capacity without a plug-in...

5.6 The actual GetCapabilities and GetMaps: requests and responses

GetCapabilities response

The first step of a user planning to use a WMS engine is to post a GetCapabilities request. Our system should respond to this request with a GetCapabilities XML document. This document should provide appropriate information on the data and on the service that the system can provide. In addition to the common parameters such as the Bounding Box, the Coordinate Reference System and the Layers, since our service provides different time-states of the same objects, it shall have a `TIME` parameter. The modalities of this parameter were reviewed in section 3.2.2.

Our service generates SVG format animations with built-in graphical user interfaces. As explained in Köbben's [34] article on RIMapperWMS, a user may or may not want to retrieve the GUI, depending if he wants to overlay the map product with other WMS products. The user should be informed of the possibility of getting the additional GUI or not. For this purpose, a parameter of type 'boolean' should be included. The name of this parameter is `getGUI`. Additional information on the visualization modalities and on the types of GUIs proposed should be provided within `<documentation>` parameters.

[9]The browsers with such capabilities are Opera, Safari and Internet Explorer with a plug-in.

GetMap request for distributions and motion dynamics

Using SVG and client-side interactivity, we hope that two types of GetMap requests will be enough for all visualization functions. A first type of request can be used for distribution as well as for motion dynamics visualizations. A second type of request should be used to visualize brushed distributions.

The difference between an interpolated animation and a step-wise animation of the same data are very similar in SVG – thanks to the power of SMIL. The request shall contain the following parameters:

- `VERSION=1.1.0`
- `REQUEST=GetMap`
- `LAYERS=icebergs`
- `CRS=3031`
- `BBOX=3000000,-3000000,3000000,3000000`
- `WIDTH=500`
- `HEIGHT=500`
- `TIME=1986-01-01/2006-12-31`
- `FORMAT=image/svg+xml`
- `getGUI=true`

This request should retrieve an SVG animation of the `iceberg` layer with start-time 01.01.1986 and end-time 31.12.2006 for the whole Antarctic region. The last parameter is vendor specific [34]. It specifies that an additional GUI should be retrieved.

We hope to be able to design interactive functions enabling all the visualization features to be rendered on the client-side. We predict that the least simple of these mechanisms to provide may be a way of setting the temporal scale.

GetMap request for brushed distributions

For brushed distributions, the `GetMap` request should be quite similar to the one above. The only parameter that needs to be different is the `TIME` specifier. Instead of a simple interval, an interval and an additional periodicity parameter should compose this element.[10] Here is an example:

```
TIME=1986-01-01/2006-12-31/P1Y
```

[10] WMS intervals and periodicity are explained in section 3.2.2

With all other parameters identical to the previous request, this should retrieve a brushed animation of the icebergs' instances (existing at that day) whose time-stamps are closest to the first of January of each year.

5.7 Summary

In the present chapter, we described how we envisage our system, its user interface, its functionalities, its way of storing spatio-temporal data, its overall structure and the communication between client and Map Server. In addition, we proposed steps for converting the temporal dimension of the data from the format recommended by OGC to a format that can be integrated in Web-based animatable vector graphics formats. In the next chapter, we will present how we went about the implementation of the system and its various components.

Chapter 6

TimeMapperWMS prototype implementation

6.1 Introduction

The approach followed to implement the system was four-fold. First, we had to follow OGC's recommendations for storing time-stamps in the database. With this in mind, we assured ourselves that there was a way to convert these stamps into a format that could be rendered by our animation engine. The second step was to produce the desired animated and interactive behavior without our objects being the reflection of the true data. The third step was to implement high-level programs that would generate the desired animations and interactive functions from the data in the database. Finally, the fourth step was to make our system respond to WMS GetCapabilities and GetMap requests.

In the present chapter, for better clarity, we deviate slightly from the sequence of these steps. As seen previously, one of the functions we intended to develop was a mechanism for the user to set the temporal scale. Our desire to make this speed-control accessible to the user without reloading the animations challenged us to develop a fairly complex mechanism. It is easier for one to understand how this mechanism works after one has seen how animations are built and how variables are stored on the client side.

6.2 Populating the database

For the data to be available, it first needed to be entered into the PostgreSQL database that the original RIMapperWMS makes use of. This was done following the indications provided in our design chapter. An additional attribute containing time in 'seconds since 1/1/1900' was added to the original dataset. To take a load of the system at run time, the data was stored in two different projections. The first is the lat/long positions of the original dataset and the second is a stereographic projection commonly used for Antarctica.

6.3 Client-side visualization functionalities

This section is about how the different client-side functionalities were implemented. First, we report on how the different types of object animations were built. Second, we explain the development of the different temporal legends. Third, we present the mechanisms found to control the temporal dimension of the animations. Finally, we explain how we offer the user control over the different visualization functionalities.

6.3.1 Building animation behavior

Before scripting the animations, the first step to visualize the icebergs was to symbolize them (Vis-IBspec:Discs). We used the SVG <circle> element for that. As projected in the previous chapter, several types of animations are developed to visualize the iceberg data: simple stepwise animation of the data, interpolated animations, interpolated and animated tracks and animation of the iceberg-area attribute. The reader shall not be surprised that most of our animations don't have any values for the 'begin' and 'dur' attributes. The reason is that these values will be set dynamically after the user sets the speed at which he wants to view the animations...

Stepwise and interpolated animations

Building simple stepwise animations (for MCB-Ds), interpolated animations (for IMBs) was achieved very simply using the SVG SMIL animate element. The script below shows a circle element with two of the latter.

```
<g id="IB_animation" style="fill:lightblue; stroke:red;
    stroke-width:4">
   <circle id="IB_A35B" cx="0" cy="600" r="25">
     <animate id="XanimIB_A35B_0" attributeName="cx"
        begin="0s" from="0" to="200" dur="2s"
        calcMode="discrete" repeatCount="none"
        fill="freeze" />
     <animate id="YanimIB_A35B_0" attributeName="cy"
        begin="0s" from="600" to="550" dur="2s"
        calcMode="discrete" repeatCount="none"
        fill="freeze" />
   </circle>
</g>
```

The first animates the x position of the object (specified by the attribute and value attributeName="cx" and the second animates the y. In brief, this circle is going to move from 0 to 200 on the x-axis and from 600 to 550 on the y-axis. This movement is achieved in a duration of 2 seconds. This code

thus only shows a single animation segment. To animate an object over a possibly very long trajectory from data in the database, the procedure is to add `XanimIB_A35B` and `YanimIB_A35B` animation elements for each segment.

The `calcMode` attribute specifies what type of animation should be displayed. Among the options offered by SMIL are `"discrete"` for a stepwise animation and `"linear"` for a linearly interpolated animation. In this case, we show `"discrete"` which implies that the movement will happen in leaps. If `"linear"` is chosen, the client-side application (browser) automatically calculates the intermediate positions where the object should be shown for the animation to appear smooth.

In chapter 3, we had predicted that we would use the keyTimes/value method. The reason we opted for these less compact separate animation clauses is that in the following type of animation, as we will now see, use had to be made of time-stamps individually. It made more sense to build sister-animations (over the same segments) using the same time values than to use different methods.

Animating interpolated tracks

The task of animating interpolated tracks revealed to be considerably more complicated. SVG does not have a built-in method for rendering lines of changing lengths. Hopefully, the SVG community provides many work-arounds for those features not (yet) implemented. The technique adopted here consists of using an SVG path element like the following and entering the proper parameters in a rather complex ECMA-Script function.[1]

```
<g id="IB_A35B_track" fill="none" stroke="lightblue"
      stroke-width="10">
    <path id="IB_A35B_track0" d="M0,600 l200,-50">
    <animate id="dashAnimA35B0"
        attributeName="stroke-dashoffset"
        dur="2s" begin="0s" fill="freeze" calcMode="linear"/>
    </path>
</g>
```

The attribute `d` is the actual path. `M0,600` is the coordinate of the starting point and `l200,-50` is a relative translation from that point. Again, we only pasted the code for one single segment. However, in this case, only one `<animate>` is necessary to display the growing segment both in x and y directions.

Regarding the projected functionality of informing the user on which trajectory positions are real data an which have been interpolated, preliminary testing convinced us that the implementation of this feature could be postponed to a later stage. In effect, firstly, the real positions are available in the MCB mode and secondly, in most cases, the angles between segments of iceberg trajectories

[1] The code of this function can be found in of Appendix D.2.

are big enough to be obvious to the user when interpolated tracks are switched on. The procedure we propose to implement this informative functionality is to use SVG's `set` animation element. We used this method in the implementation of the feature we present in the following paragraph.

Making objects appear and disappear

Objects such as icebergs appear and disappear. To render these existential changes, we made use of the `set` element and applied it upon the `visibility` attribute of the objects themselves. At the time an object "came to existence", it's this attribute is set to `"visible"` and at the time it disappeared, it is set to `hidden`.

Because of time constraints, animations to *emphasize* existential changes have not yet been implemented. This could be done in several ways. We could use a very similar method to the one just explained for their appearances and disappearances. We would use multiple `set` elements[2]. The frequency of the blinking or flashing could be fixed adequately. We foresee that there will be a tradeoff between the effectiveness of the blinking for an emphasis purpose and the portion of the object's trajectory over which this blinking will take place. Another way would be to insert new objects which represent the calving events. In this way, the blinking or flashing would emphasize the event at the position that it took place.

Brushed animations

For brushed animations, from a visualization point of view, stepwise animations would render acceptable results. In that sense, nothing remains to be done on the client-side to render brushed animations. However, animations built in this way would not reflect what happened in reality. We might prefer to again make use of SVG's `set` animation element and apply it on the x and y positions of the objects. This has not been implemented yet but shouldn't be difficult to achieve.

Animating attribute changes

Animating attribute change is done using elements almost identical to those for position changes. The parameter that differs is the attribute of the object that shall be animated. Instead of `attributeName="cx"` or `="cy"`, `attributeName="r"` is animated.[3]

[2] Or a keyTimes/values method if it exists with the set element

[3] The code of such animations can be found in Appendix D3.

6.3.2 Building temporal legends

The temporal legends were built following the same principle and methods as the animations. First their graphics are scripted and then their dynamic behavior.

The graphics of the time-bar legend are composed of three rectangular elements and two text elements. One rectangle represents the the time not yet elapsed (or future), another the past and the third serves as present vehicle-sign. The objects are animated in a very simple manner as we described in chapter 2 for Harrower's time-bar legend.[4]

Three different types of cyclic temporal legends were implemented. The first corresponds to the the small version of the pie cyclic temporal legend planned in the design chapter (the result can be seen in fig. 7.3. It was a challenge to develop because no SMIL technique exists to animate growing pie-portions. We had to use several rotating, appearing and disappearing half circles that graphically seem to be one single growing pie-portion.[5] Originally, we intended for the colors used to represent the past and the future to be the same as those used in the time-bar. We now believe that this would not be appropriate because the span of one cycle does not correspond to the span of the time-bar.

As planned, a small and a big cyclic legend mainly made of a rotating watch-type hand were also implemented. The first can be seen in figure 7.2 and the second in figure 7.1. We will evaluate the design of these three types of cyclic legends in the next chapter and recommend the further development of the big pie-portion cyclic legend that we describe in our design.

We have not yet implemented a digital clock. Although its principle is very simple, its realization is somewhat more demanding. It will involve three main mechanisms. The first consists of getting the time of the animation. The second is to get the system to update the time of the clock at a high frequency both when the animations are playing freely and when they are triggered by the time-slider. The third – and most challenging – is to generate real-world time strings in a multi-unit time format from display-time in seconds.

The first and second can be achieved by running the SVG function `getCurrentTime()` (present of the animation in display-time) at a rapid frequency and to update the value of the string. To go about the third, we propose a series of calculation steps:

1. Divide present of animation (in display-time) by the temporal scale.
2. Add start-time of temporal-extent.*At this stage, we have real-world time in seconds since 1/1/1900 (RWT1900).*
3. Use the following calculations to determine which year the present moment of animation represents.

[4] Code in appendix D4
[5] Code in appendix D5.

4. Divide RWT1900 by the number of seconds contained in a 4-year span (necessary because of bissextile years)
5. Multiply that result by 4 and store it as: 'years from 4 cycle'.
6. Take the remainder of the division and successively attempt to divide it by the number of seconds in a 3-year cycle, a 2-year cycle and in 1 year till an integer is output (only on 3rd round is 0 considered the answer). For this, we need to know when in the 4-year cycle the bissextile years are placed.
7. Once this integer is found, we add it to the result above ('years from 4 cycle') and further add 1900. *At this stage, we know which year we are in.*
8. The remainder of the previous calculation is the time in seconds of the moment within the year. Analog steps must be applied to determine which month and which day the animation is depicting.

6.3.3 Controlling time: time-slider and other functionalities

The time-slider

The time-slider was the most important challenge for our project. In our attempt to control time, the first step was to affect the temporal moment of animations at all. The second step was then to build a slider to offer the user a convenient way of controlling that moment.

After much looking around into complicated ways to control the moment of the animations, three simple lines of ECMA-Script did the trick:

```
SVGDocument = evt.target.ownerDocument;
SVGRoot = SVGDocument.documentElement;
SVGRoot.setCurrentTime(time);
```

The last line is actually the one that "acts". The two others have the function of storing the object over which it is meant to produce its action. It was however mastering these two first lines that was the most difficult, as, for that, we had to understand how the SVG XML DOM was built.

The code of the slider itself was borrowed from carto.net [52], a website specialized in SVG cartography, which provides many GUI items for mapping.[6] It revealed to be very simple to use.

```
TimeSlider = new slider("TimeSlider","TimeSlider",300,650,
   value1=beginTime,700,650,value2=endTime, 0, TSliderStyle,
   invisSliderWidth1,"sliderSymbol",controlTime,true);
```

To design our slider, we entered `beginTime` and `endTime` of the time extent to be visualized as values of the beginning and end of the slider. As the user drags

[6] This slider was first developed in JavaScript by Kevin Lindsey [40].

the thumb – here, the present vehicle-sign – a variable called `value` changes. The slider triggers a function that we called `controlTime`. This function in turn triggers the code presented above: `SVGRoot.setCurrentTime(time)`.

Play, pause and loop functions

In addition to the time-slider, play, pause and loop functions were implemented. The pause function makes use of the SVG `pauseAnimations()` mechanism and the play function of "un-pauses" the animations. The looping function was implemented by having the end event of the time-bar legend trigger a function which sets the present time of the animations to 0.

6.3.4 Providing visualization options to the user

Providing the user with the possibility to choose the different types of visualization modes and to turn on or off the additional visualization aids was done by using event-triggered functions. While in SVG, an event can be a 'mouse-over', a 'mouse-out' or a 'mouse-move' (such as used to drag our sliders), here, we only used 'click' events.

When the user, for example, chooses to visualize motion dynamics, he clicks on the appropriate tick-box. This triggers an ECMA-Script (ECMA) function that activates the desired visualization mode (by setting the appropriate parameters via the DOM) and deactivates the other visualization modes. To give an example, for changing from the distribution visualization mode to motion-dynamics, four main actions are triggered.

1. The tick-box changes appearance to indicate to the user that that visualization mode has been activated. Symmetrically, the tick-boxes of the other visualization modes are deactivated.
2. Also for the purpose of informing the user on what he should expect, the title of the visualization is set to indicate what visualization mode the application is in.
3. The value of the `calcMode` attribute is changed to `"linear"`, which, as seen above activates linearly interpolated animations.
4. Because we recommend interpolated tracks for supporting motion-dynamics visualization, the motion tracks animations are set to be visible as well. These can be deactivated by clicking the appropriate button.

The main mechanisms behind these actions have already been presented above. They act on the attributes of SVG elements. For several of the actions above, the 'visibility' of objects (and tracks) is set using the attribute `display`. It is set to `"none"` to make an element virtually disappear.[7]

[7]This attribute may have advantages over the 'visibility' attribute concerning the load put onto the computer's memory [80].

6.4 Serializing: generating interactive animated maps

As foreseen in the previous chapter, the serialization done by our map server is the process of converting data and graphical user interface elements into SVG and ECMA-Script files. Together, these code in these files compose the desired application. Serialization was implemented with the Java programming language. The work mainly consisted of converting the data into SVG animations, storing temporal values as ECMA-Script variables and building ECMA-Script functions that loop over all objects to set the time of their `begin` and `dur` attributes. A smaller part was to serialize the GUI and interactive functionalities.

6.4.1 Querying the database

To build the stepwise and interpolated animations as well as to store ECMA variables, the first step is to query the database using the spatial bounding-box, the temporal extent and the type of animation entered by the user. As planned, both spatial and temporal intersects were implemented and the output is a view of the data with those iceberg tuples which match the request.

To implement brushed animations, the main task that is to query the database in the a different way than it is done for “simple” animations. For each repetitive cycle of the temporal extent, the data values of the objects’ instances which are closest to the repetitive date shall be retrieved.

At present, the mechanism that queries the database has not been fully automated. The user is not yet free to specify the temporal extent he wishes to visualize. The work needed to achieve this is however known to be realizable.

6.4.2 Building animations from the data and storing time-stamp variables

To build the SMIL animations and store ECMA variables, we consider that the data view described above is ordered first by object (icebergs) and then by time instances (as shown in figure 6.1-A). The tuples of every object are indexed. For each segment that a given object travels, an x and a y `animate` elements are generated. These animate elements get names like `"XanimIB_A35B_0"`, `YanimIB_A35B_0`, `"XanimIB_A35B_1"`, etc... For one x and one y `animate` elements, the values of the `from` and `to` attributes are set in the following way: for the x-`animate`, the x-position of the first point of a segment is injected into the `from` attribute and the x-position of the second point is injected into the `to` attribute. The time-stamps of every position are stored in ECMA variables. These animation generation and variable storage are represented in figure 6.1-A. For the time-being, the reader need not attend to figure B; we will come back to it in the next section.

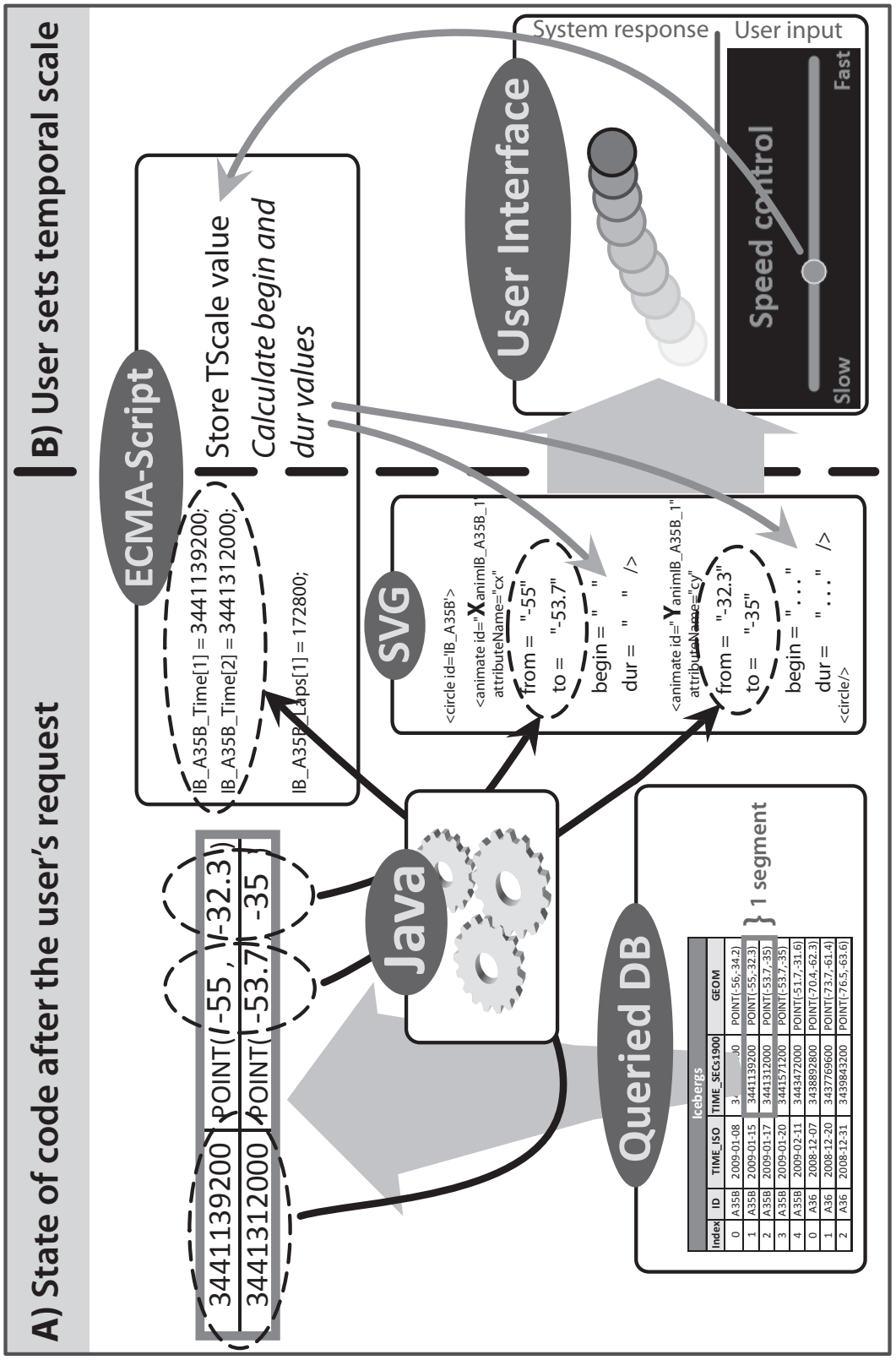


Figure 6.1: A) The Java serializer takes selected iceberg instances to build animations and store time-stamps in ECMA-Script variables — B) When the user sets the temporal scale, the 'begin' and 'dur' values are calculated from the stored time-stamps, and injected into the animations. These can then be rendered in the client.

An analog procedure would be followed to generate animations of the size of the iceberg symbols. Although the mechanism is ready for implementation, the serialization functions have not yet been scripted.

6.5 Setting the temporal scale: a functionality that binds all

To explain how the speeds of animations are set, we will start by explaining how the user inputs the temporal scale and what we did to optimize this functionality. Then, we present the mechanism that makes use of the temporal scale set by the user to render animations at the requested speed.

6.5.1 Setting the value of the temporal scale

The way we found to control the speeds of SMIL animations was to dynamically change the values of their `begin` and `dur` attributes. To display an animation at a "low speed", we multiply its real-world time stamps (RWTS) by a relatively big temporal scale and to display it "fast" by a relatively small temporal scale.[8]

A slider similar to the one described above was used. However, the original code did not intend for developers to enter non-integer values (such as we need for temporal scales). So to get the small values needed, we made the slider output a factor (very much resembling a spatial scale factor) that would further need to be "inverted" ('`1/x`' calculation).

Preliminary testing showed us that, for large temporal scale values (slow speeds), it was very difficult for a user to set the slider conveniently. The reason is that a relatively short slider had to output temporal scales going from 0 to 1000. So even to change the temporal scale from the slowest speed to ten times that, the user needed to drag the slider over only 1/100th of its total length.

To solve this problem, we analyzed the relation existing between the input value and the output temporal scale value. This was a linear relation: 0 goes to 0, 10 goes to 10 and 1000 goes to 1000. Figure 6.2 shows this linear relation and the curve that we propose to replace it with. We estimated that the user should move the slider approximately by 100 to get an output 10, by 500 to get an output 200 and of course 1000 to get an output of 1000. We matched these values with the polynomial equation $y = 5 \cdot 10^{-7}x^3 + 0.0005x^2 + 0.0423x$.

This problem solved, we may now report on how the user is informed of the value of the temporal scale. As planned, this value, as well as the duration of the total animation (in display time) are offered to the user as dynamic text inside the animated map display. To do the latter, we need to multiply the real-world time-extent (stored as an ECMA variable in 'seconds since 1/1/1900') by

[8] E.g., to view two years of iceberg data in 2 minutes the TScale should be 1.9*10-6 and to view it in 30 seconds, the TScale shall be ca. 4.8*10-7

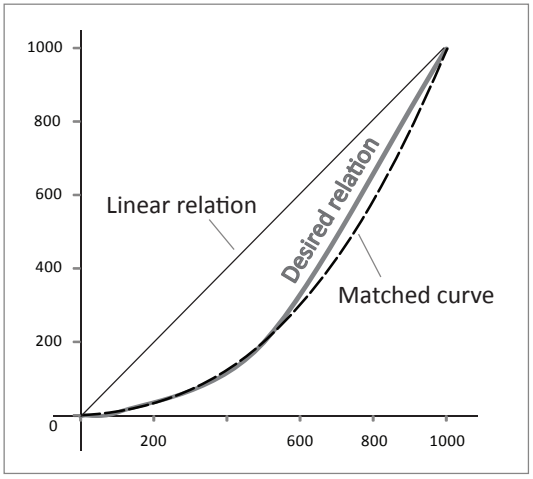


Figure 6.2: Transforming a linear relationship to a 3rd degree polynomial curve to provide the user with more flexibility

the temporal scale and set the value of an SVG `text` element to the informational text-string.

6.5.2 Setting the start-times and durations of the animations

Setting the start-times and durations of the animations (values of the `begin` and `dur` attributes) according to the temporal scale input by the user was achieved via DOM manipulation. This process is represented in the figure 6.1-B, already seen above.

To be accessible via the DOM, all the SVG `animate` elements – including the animations of the temporal legends – need to be retrieved into ECMA-Script variables. The RWTS stored by the serializer in ECMA variables are multiplied by the temporal scale and injected back into the `animate` elements as values of the `begin` attributes. The code below shows a simplified[9] example of this procedure for one single start-time (`begin` attribute):

```
IB_A35B_Time[0] = -432000;
IB_A35B_Animbegin[0] = IB_A35B_Time[0]*TScale;
XanimIB_A35B_[0] = SVGDocument
    .getElementById('XanimIB_A35B_0');
XanimIB_A35B_[0
    .setAttributeNS(null, "begin", IB_A35B_Animbegin[0]);
```

Similar procedures are used to inject appropriate values into the `dur` attributes of object animations, into `begin` and `dur` attributes of temporal legend animations and into the end value of the time-slider. The only difference for setting

[9]The true code includes loops.

the `dur` is that these durations need to be calculated from two object time-instances (by subtracting RWTS`[i]` from RWTS`[i1]`+). Setting the end-value of the time-slider is critical. In effect, a change of the temporal scale involves a change of the display time-extent and the beginning and end values of the slider should always correspond to this duration.

6.6 Integrating WMS map backgrounds and additional data

RIMapperWMS' developer managed to make the system zoom and pan external WMS layers. It was therefore possible to integrate valuable map backgrounds on Antarctica. The layers we got came from the Antarctic Cryosphere Access Portal (A-CAP) [69].[10] Among others, we included as base-map: the Antarctic continent and islands, the ice-shelf, the coastline, the Antarctic circle and a lat/long grid. Unfortunately, the portal did not contain an appropriate bathymetric layer. As additional layers, we added lines representing ocean current fronts, geographic names and a layer showing classified sea-ice concentrations. Unfortunately, the latter shows only one static state and was only included as an example of interesting data to correlate to iceberg dynamics.[11]

6.7 State of the implementation

The present stage of the implementation allows us to confirm that combining WMS Time Dimension and SVG interactive animation is viable. Both GetCapabilities and GetMap mechanisms work. At this stage, only a temporal subset of the data can be visualized but the implementation of a more flexible response mechanism is on its way. During the implementation phase, the client-side animation and interactivity were developed to run in the Opera browser. Currently, it is the only browser that offers a built-in SMIL rendering engine. However, we have good hope that the amount of work to make the implementation run in other Internet Explorer and Safari might be small. To make it run in Mozilla Firefox and Chrome using their internal SVG capabilities, a work-around called fake-SMIL could probably be used.

Regarding client-side visualization functionalities, most of the animation and interactivity features that we intended to develop are functional. We will present these more extensively in the next chapter. No obstacle stands before the implementation of the remaining functions. In some cases, the client-side animation mechanism is ready to be automated on the server-side. For other tasks, we still need to think about the best implementation approach.

[10]The number of layers available is impressive, the datasets offered are even more numerous and A-CAP projects to offer all their resources as WMS layers, which is promising for iceberg studies.

[11]A-CAP actually offers time-series sets of images of sea-ice concentration.

Chapter 7

Results, testing and evaluation

7.1 Introduction

The result of our implementation is a WMS prototype that we call TimeMapper, which is destined to visualize iceberg dynamics. The dataset used is the NIC's Antarctic Iceberg Dataset [49]. The graphical appearance of the system can be seen in figure 7.1. We consider this result to be composed of generic animated mapping functionalities on the one hand and visualization functionalities specialized for moving-object data on the other.

The main generic features that we implemented are: a mechanism for the user to choose the temporal extent of the animations, temporal legends, a time slider and a way for the user to control the speed of the animations. A less central but yet significant feature is the looping function we implemented. The more specific features developed are different types of animations for moving-object visualization and their attributes. The different object animations include the stepwise animations and the interpolated animations, including the objects' interpolated tracks.

In this chapter, we shall report on the testing and evaluation that we did of these two kinds of features. Our objectives are of two types: First, we wish to assess how well the generic features function. Second, we will try and determine the respective advantages and flaws of the different types of animations proposed. How well do the animations – those for destined to visualize the evolution of distributions (MCB-Ds) and those destined to visualize motion dynamics (IMBs) – fulfill their purposes? In addition to these objectives, we would have liked to explore the dataset more in depth and try to detect spatio-temporal patterns in the data. In effect, to make a full evaluation of the features proposed, we would need to assess the system's effectiveness for detecting such trends. This objective however surpasses the scope of this research and was not possible because of time constraints.

Since the implementation is still in an early phase, it has various runtime limitations. We will start by stating what these limitations are and how they can be partially sidestepped. After that, we will begin the evaluation *per se*. We should mention that in addition to our own testing, we had planned to have our system evaluated by one or several data exploration or iceberg specialists but because of time limitations again, this was unfortunately not possible. Finally, we shall step back from the prototype and evaluate how well the design and implementation of the system could (1) be used to visualize other moving-object datasets and (2) be extended to visualize other time-series datasets.

7.2 System testing: functionality and limitations

7.2.1 How well do the prototype's functionalities work?

A five-year subset of the data was used to dynamically generate GetMap responses for all the iceberg objects existing in that time span. We chose the years 1999 to 2004 as our time-extent because of the major calving events which took place during this period.

In addition to functionalities to manipulate the temporal dimension of the phenomenon, the original graphical user interface of RIMapperWMS is also part of the TimeMapper system. Both zooming and panning functionalities provoke new GetMap queries to the servers: spatial subsets of the icebergs are dynamically loaded along with additional WMS layers on Antarctica (which we get from A-CAP's map server).

The application loads quickly and the generic features implemented are all functional. However, the response speeds of the temporal scale setting function and of the time-slider are poor when many objects are visualized. The user needs to wait nearly one minute for the speeds of the animations to get set. Once the temporal scale has been set, all animations function well despite the large number of objects visualized. The time-slider works as well but in a very lagged way. Its responsiveness is so slow that it makes it unusable.

Because of these limitations, we decided to take a much smaller subset of the data to effectuate our tests: we took only three icebergs. This improved the responsiveness of both the temporal scale and the time-slider functionalities.

7.2.2 Solutions to improve responsiveness

The reason why the system takes a long time to integrate a temporal scale value entered by the user is that the values of the `begin` and `dur` attributes of hundreds of animation elements need to be set and that this setting passes through the XML DOM. Four solutions have already been identified to solve this problem. One of them involves making use of the much more compact way of writing animations constituted by the method we called the `keyTimes/values`

method.[1] The two others were recommended by an SVG and SMIL expert.[2] The fourth solution is not yet implementable. SMIL already has a built-in way of controlling the speeds of animations. This technique has been integrated in the upcoming version of SVG but is not yet rendered by any existing browser.

To improve the effectiveness of the slider for controlling large number of animations, it appears in this case also, that using the more compact form of animation that we called the `keyTimes/values` method offers a good chance of improving this responsiveness. Because of testing we did, our guess is that it will either improve it drastically or not at all. Even with the three object test-case, the number of animations is larger than 30. Our testing indicates that it may well be the number of animations, and not their complexity, that provokes lagged responses. In effect, we tried to interactively control complex animations such as combined rotations, re-scaling, color changes and even the morphed-paths animation that we saw in chapter (p.**??**). Their responsiveness to slider movements are all amazing. All these animations are characterized by a small number of animate elements but also by rather complex effects.

7.3 Evaluating the generic interactive functionalities

Time-slider

The time-slider works well with few objects and there is good hope of transforming the animations to make them more easily controllable. and in those conditions, the responsiveness of the animations to the user's mouse movements is amazing. Unfortunately, the advantage offered by the time-slider increases with the number of objects and the complexity of the behaviors observed. In effect, it is in such conditions that the user has trouble making sense of, and remembering what he sees. Therefore, evaluating the true value of the time-slider will only be possible if we manage to make the changing of the moment of animations less heavy for the computer's memory.

Looping

The looping function works well. It also provokes a delayed effect between the end and the new beginning when the number of objects is large. Because we could not reset the temporal extent and because looping over five years of movement data made no sense, we could not evaluate the usefulness of this functionality.

[1] This method would reduce the number of 'animate' elements by a factor of ca. 50.
[2] They consist of methods to neutralize known limitations of the DOM structure.

Temporal scale setting

Apart from the mentioned response time limitation with numerous objects and animations, the temporal scale setting function works very well. The work we put into making the relation between the slider position and the output temporal scale makes it much easier for the user to choose an accurate value for the speed of animations (which was formely impossible).

Temporal legends

One time-bar and three types of cyclic temporal legends have been implemented and we shall now attempt an evaluation of their effectiveness.

We said earlier that time-bar legends can potentially serve two purposes, the first of which is to help the user orient himself in time: what moment of the animation are we viewing (in respect to its beginning and its end)? The second is to give him an idea of the speed at which the real-world phenomenon is being depicted. With the present test-case of the prototype, the time-bar is useful for both of these purposes (the time-bar can be seen in figure 7.1. However, the absence of a digital clock implies that the user needs to keep in mind what are the start and the end times for the moments and the speeds to be meaningful. A digital clock is therefore required. The full advantages of the time-bar will come with the possibility for the explorer to choose the start-time and end-time of the animations. With smaller and task-specific time-spans (e.g., one year), the time-bar will be more useful.

Regarding cyclic temporal legends in general, our impression is that they are very useful to the user because their cycles provide him with a constant temporal reference frame. No matter what the speed or the temporal extent of the animation, one cycle corresponds to the same amount of time (in our case, one year). In addition, as stated in the literature, the angle of the rotating hands provides the user with the means to know what time of year is presently being visualized. This is particularly useful for the study of phenomena which, like icebergs, have a seasonal component.

The test cases we had did not fully allow us to compare the three cyclic legends developed. In general, our impression is that all three have different advantages. The ‘small hand’ is discrete and hardly takes any space in the display (see fig. 7.2a. Of the three cyclic legends, the ‘small pie’ is the one which obstructs the display the most (see fig. 7.3). For some mapped areas, there may be a convenient place to put the legend but for others, there may not.

As explained earlier, the advantage of the ‘small pie’ is that it provides an additional indicator for an event-related beginning of cycle. However, one could argue that simple line at the place of the cycle considered to be the start could just as well serve this purpose. The question that remains is whether the growing pie-portion improves to user’s perception of which stage of the cycle he is visualizing. The ‘big hand’ has the advantage of totally freeing the center of

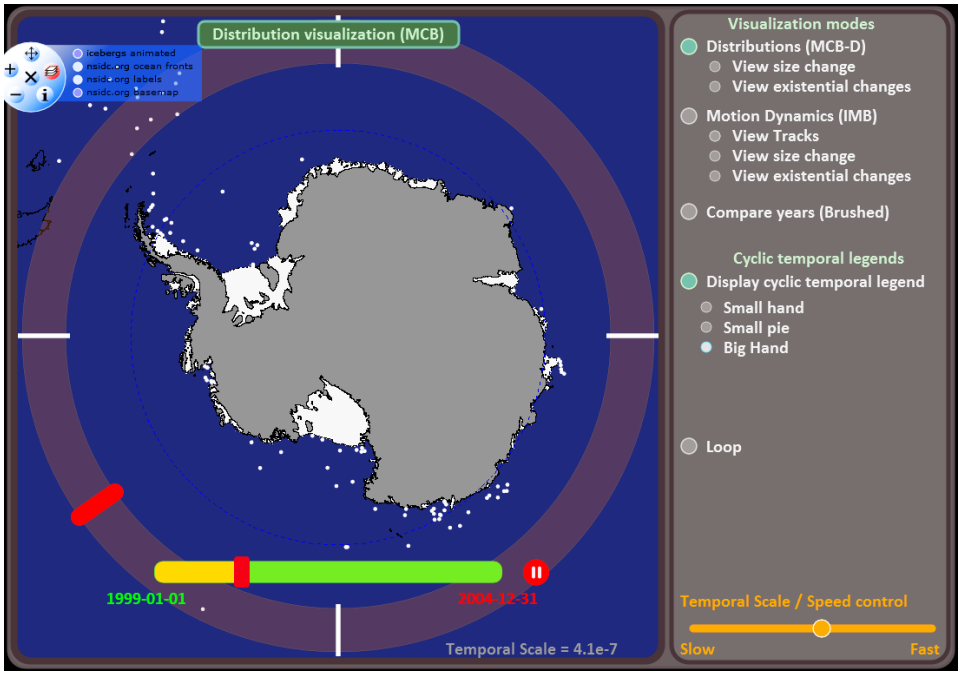


Figure 7.1: The TimeMapperWMS prototype in 'distribution' visualization mode (here with the 'big hand' type of cyclic temporal legend

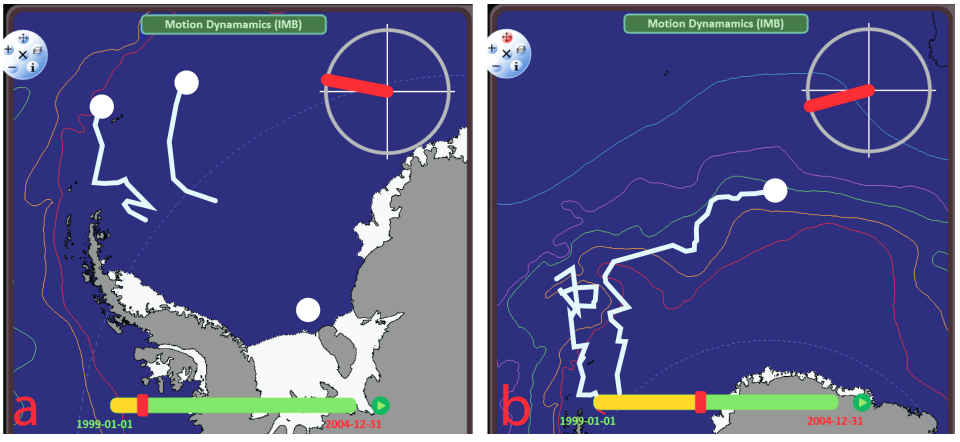


Figure 7.2: Two stages of a 'motion dynamics' visualization showing two icebergs (and a third one which is grounded): In a), the two icebergs are traveling at similar speeds and in the same direction. In b), the two icebergs have taken different trajectories.

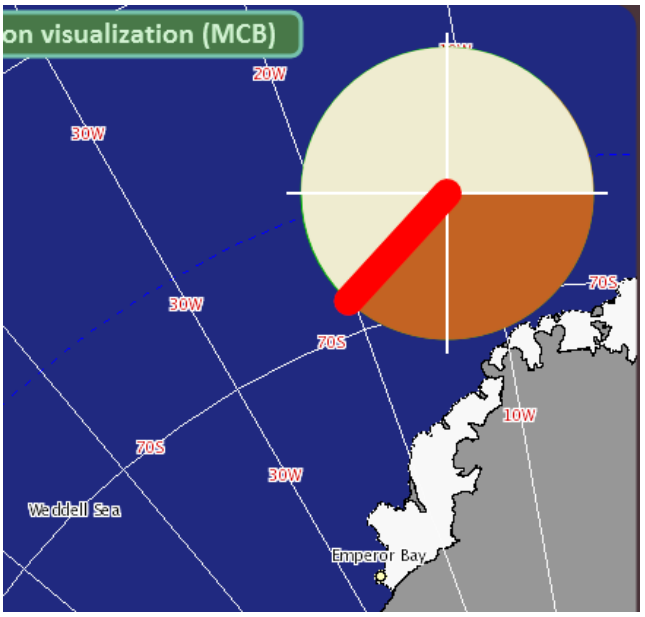


Figure 7.3: The 'small pie' cyclic temporal legend

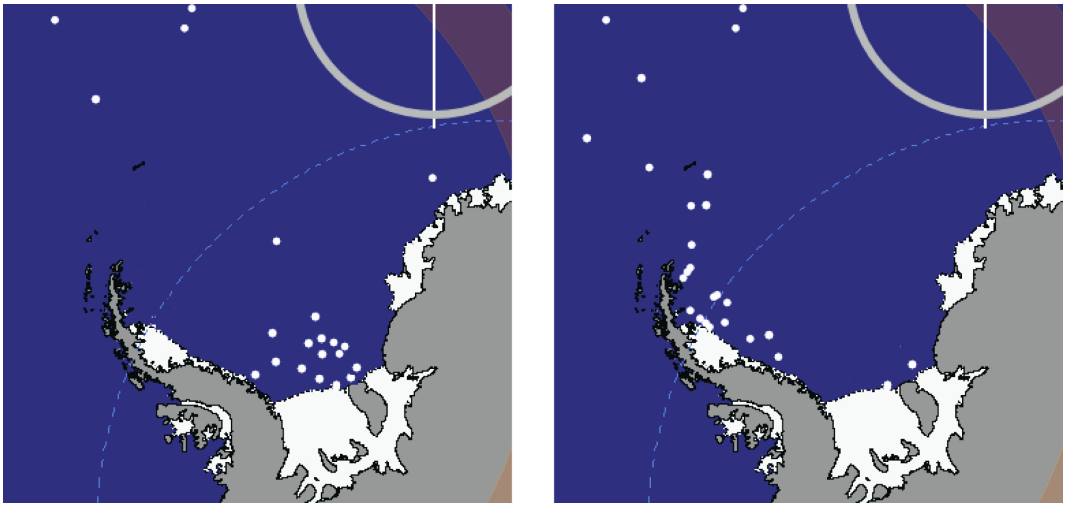


Figure 7.4: Moving cluster pattern: a. Cluster of icebergs soon after a major calving event, b. Six months later, the icebergs have traveled North West by 500 to 1500 km.

the display. But our impression after little testing seems that the hand rotating around may be distracting.

7.4 Evaluating animation types for moving-object visualization

Limited testing enables us to confirm that the two types of animations implemented indeed favor different visualization tasks. The user's attention is automatically brought upon different types of behaviors.

With the 'distribution mode', our attention mainly went to groups of objects. Major spatio-temporal trends in the objects' distribution were easily detected. For instance, as suggested in figure 7.4[3], after a major calving event from the Ronne Ice-shelf in 2001, a large cluster of icebergs traveled North-West along the Antarctic Peninsula between. Other icebergs were more dispersed in space.

In contrast, the distribution mode is not effective to visualize individual motion dynamics because the viewer needs to remember where the last position visualized was.[4] In addition, the fact that position changes happen in sudden jumps keeps the viewer from having a unified perception of the movement he is witnessing.

In the 'motion-dynamics' mode, the user's attention is attracted by individual motion behaviors. With our subset test-case composed of three icebergs, the directions and speeds of movements are easy to visualize. Similarities and differences between individual behaviors can also be detected. Figure 7.2[5] shows two stages of such a visualization. In the first two icebergs are moving in similar directions and speeds (at the same time and in the same region). The interpolated tracks are particularly helpful to compare such trajectories. A third iceberg is grounded close to its time of calving. It will remain grounded for several years but the data still contains minute in its position.

The two frames of the mentioned figure also show an additional WMS layer with (static) ocean current fronts. The red line is the limit of the Antarctic circumpolar current. In the first visualization, it seems probable that the motion of the two icebergs are both influenced by the same current. In the second frame, the two icebergs have taken different trajectories. The one on the left seems to have got caught in a sort of circular current. The one on the right follows the current for a period of approximately 6 months before disappearing. Animation is more useful in the exploration of the first phase than it is for the second because it enables the viewer to observe the relative movement between the two objects.

[3]These results have been simulated because, at the time of taking these screen dumps, the server was not functioning with large numbers of icebergs.

[4]Solutions to this have however been found, by using progressive fading of the previous positions. We however don't think this would work well with many objects.

[5]In the figure, we have digitized the tracks and made their stroke wider for better visibility and we also increased the size of the iceberg symbols.

One might be tempted to compare these animations with possibilities offered by static mapping. However, as we have seen earlier, this question has however already been answered by other authors: special techniques *are* necessary for the visualization of time-series georeferenced data.

The icebergs sometimes appear to move at irregular speeds. While part of this behavior may be explained by real-world iceberg speed variations, it is certainly also due to limitations of the dataset. When we presented the Antarctic Iceberg Dataset, we predicted that the low temporal precision of the recordings would be a limitation. The fairly low temporal resolution is also a limitation. As an example to demonstrate this, we can analyze the case of an iceberg moving slowly over a short distance. Between the two recorded positions, the object may have followed a long and complicated trajectory. We conclude after others that, in cases where the temporal resolution is not high enough to record a sufficient proportion of the changes taking place, interpolation can make a viewer draw erroneous conclusions.

Similarly to our above conclusion that motion is not well rendered by stepwise animations, we assert that studying distributions with interpolated animations is not as effective as with non interpolated animations. In effect, motion that can easily be followed by the eye unnecessarily loads the user's cognition. With the distribution mode, although the user can often see from where to where individual objects move, these individual events can easily be ignored to assess the change over the positions of the entire population (or a subset).

7.5 Using the system for other datasets and extending its capabilities

Except for its responsiveness limitations, the present version of the system could already be used to visualize other moving-object datasets. New functionalities to animate some of the visual variables, such as hue and value could very quickly be implemented.

The interactive functionalities of the system are built in a way that makes them perfectly usable for controlling other types of animations. Preliminary testing has shown that re-scaling, rotations and complicated shape transformation animations can already be controlled by the present features.

Chapter 8

Conclusion and recommendations

8.1 Combining distributed services and vector animation

The main objective of this research was to look into possibilities, both from theoretical and practical perspectives, of combining animated and interactive vector graphics with distributed geo services. This double objective has been fully attained. We pointed out the two frameworks that appeared to be most promising for this combination: OGC's WMS and the W3C's Scalable Vector Graphics. We showed on a theoretical level that they could be combined and demonstrated it on a practical level by developing an animated mapping WMS prototype. This web application, that we call TimeMapperWMS was built for visualizing moving object data in general and iceberg dynamics in particular.

The fundamental research task to achieve this combination was to determine how the time dimension of the data had to be manipulated. We determined how the temporal attribute should be stored, queried and converted to be further integrated in the vector graphics animation. The storage must be done according to the ISO 8601 extended format for the system to comply with OGC's standard. The data should be queried by applying a temporal intersect to the data in addition to the spatial intersect. Once a view of the data is retrieved which matches the user's request, the temporal attribute further needs to be converted. We identified a series of steps to effectuate this conversion. The last step applied is to multiply real-world time-stamps by a temporal scale.

To integrate these mechanisms within the WMS framework, we also specified the ways in which the GetCapabilities and GetMap requests/responses needed to be implemented.

8.2 Designing a visualization environment for iceberg dynamics

The review we did on animated mapping literature helped us to identify the most important features to offer in an animated mapping exploratory environment. Further, reviews on moving object and iceberg literature were used to point out which of these features were central for the development of our prototype.

Traditional temporal user controls in animated mapping are similar to VCR or DVD-player functions: pause/play and stop are the basic. Looping has often been proposed as well. A set of five more visualization features more specific to temporal animated mapping were also identified: (1) The user should be given the possibility to specify the temporal extent of the data that he wants to visualize. (2) Temporal legends should be offered to help the user orient himself within the temporal dimension. (3) Interactive control over the moment of the animation should be made possible via a time-slider – similar to sliders in computer-based media players. (4) The user should be provided with a way to control the speeds or temporal scale of the animations. (5) The user could benefit from a functionality destined to quickly generate and organize small-multiple maps (which are complementarity with animations).

A framework for moving object data visualization was adopted from our literature review. This framework was one of the tools we used to analyze an iceberg visual exploration use-case. The aim of this analysis was to understand the visualization requirements of specialists from various fields interested in icebergs. This led us to identify three main visualization tasks for which animated mapping could be helpful. Different types of animations were identified as potentially the best ways of going about these tasks.

The first tasks is exploring the evolution of the distributions of the whole population (or a subset) of icebergs. For this purpose, we designed and implemented simple stepwise animations of the data. The second task is to study the motion dynamics of the objects. These dynamics comprise the movements of objects in space but also their relative movement (between objects). For this, we proposed two features: linearly interpolated animations of the objects' positions and, interpolated tracks. The third task is to compare object distributions at repetitive moments. For our case-study, we intended to compare the distributions of icebergs over several years but only taking one (and the same) date of each year. To realize this objective, we proposed to make use of so-called brushed animations of the data.

Apart from these main visualization tasks, we also proposed to use animation to study the dynamics of attributes of our moving objects as well as appearance and disappearance phenomena (existential changes). The attribute that we were interested to visualize for icebergs was their size. We logically proposed to animate the visual variable size of the symbols representing the icebergs. Existential changes are automatically depicted by simple animations of the data

(the symbol appears at the time representing the first time-stamp present in the data for that object. However, these existential changes might happen to discretely to be noticed by the user. To solve this problem and emphasize these events, we proposed to make use of the dynamic visualization variable 'frequency'.

8.3 Implementation of the TimeMapper prototype

To implement the TimeMapperWMS prototype and thus practically combine the WMS framework and Scalable Vector Graphics animations, we extended RIMapper, a WMS package developed at ITC which already outputs static SVG but no animations. The data, including its temporal attribute, was loaded into a database backend following a WMS-compliant schema. In addition to storing real-world time-stamps in the recommended multiple time-unit strings, we stored an additional temporal attribute in a unique time unit (seconds).

Triggered by the user's GetMap request, the map server queries the database and does part of the time conversion steps. It then generates animations, interactive functions and a graphical user interface that are all sent as a package to the end-user.

In addition the system was extended to integrate WMS layers from external map servers. When the user zooms or pans the map, new data is loaded both for the objects and for the external map layers. In a similar way, we project to offer the user the possibility to quickly change the temporal extent of the animations. The system will then have to reload new data based on the new temporal request.

8.4 Results and system evaluation

The visualization prototype developed offers a set of functionalities integrated in an easy to use graphical user interface. It has two main visualization modes: it runs stepwise animations to visualize object distributions and interpolated animations to visualize object movement dynamics. For the latter, additional interpolated track animations were also developed.

To support the visualizations, a time-bar and three types of cyclic temporal legends were developed as well as most of the interactive controls planned. Play, pause and loop functions are proposed as well as the more advanced features such as the time-slider and an animation speed control slider.

Although some of the interactive functionalities still have a fairly poor responsiveness when used with numerous objects, they are all functional. There is good hope for the responsiveness impediments to be fairly easy to solve.

Working with a subset of the data, we evaluated the system's functionalities. They enabled us to easily detect patterns in the evolution of distributions as

well as in the motion dynamics of objects. In particular, for distributions, moving clusters were detected as well as dispersion patterns. For motion dynamics, similarities in speeds and directions between objects as well as divergence of these same objects were detected. In addition, a static raster overlay of ocean current fronts enabled us to observe the behavior of icebergs in their approach of these currents as well as their behaviors within them.

8.5 Recommendations

The fundamental research problem has been solved. The recommendations that we have therefore are related to the development of better visualization features.

A few functionalities that we had hoped to develop have not been implemented yet. This includes: the third type of object animation planned, i.e., brushed animations; animations capable of emphasizing existential changes in the phenomenon depicted; the essential digital clock temporal legend and the functionality which would output small-multiple maps from the animations. In addition, the mechanism designed to animate the size of objects to reflect attribute changes stored in the database is ready for implementation. We naturally recommend the implementation of these functionalities.

For exploring iceberg trajectories, it would be helpful to be able to identify icebergs. Moreover, such a functionality would help researchers share visualization cases. [1] For some visualization tasks, it may also be useful to differentiate iceberg objects with different colors.

Further testing of the system is necessary. The preliminary exploration we did of subsets of the Antarctic Iceberg Dataset showed promising results. The system should be tried with more data subsets which hopefully can become bigger if we manage to diminish the runtime limitations of the system. In particular, in addition to distribution dynamics and movement, we propose to explore calving events more in depth.

In our testing phase, we were only very partially able to evaluate the design of the temporal legends that we developed. Somewhat related to temporal legends, in chapter 2, we presented a hypothesis regarding the effectiveness of temporal sliders. The idea advanced is that, for vector animations, a well responsive temporal slider can be considered to be a link to another tactile interface embodied by the user's movements with the mouse. We argue that, with some experience, the user can have a fairly precise idea of the movement he is provoking with his hand movements on the time-slider and therefore on the temporal dimension. We further argue that if this is the case, such a cognitive

[1] A functionality for displaying information on objects when the user points at it with the curser has already been implemented in RIMapper and it would not be difficult to use it again for icebergs.

mechanism can reduce the split-attention effect between the map display and the temporal legends. This hypothesis would need some empirical testing.

Particularly for exploring motion dynamics that can be found in the iceberg dataset, we would recommend to contact the NIC again and check whether they possess a version of the data with higher temporal *precision*. We would also propose to try the system with other moving object datasets, possibly with higher temporal *resolution*.

We had projected to visualize iceberg dynamics with other related phenomena such as bathymetry, ocean currents and sea-ice concentration. We recommend to attempt such visualizations, if possible using time-series data of the two latter phenomena.

Finally, to expend the scope of TimeMapper, we propose to look into possibilities of generating data-driven animations of other types of phenomena and other types of data.

Appendix A

Scenario to introduce the three basic needs in animated mapping

We defend that the three basic needs in animated mapping are: time to get acquainted, temporal legends and interactivity. To introduce these needs, we are going to start by presenting a scenario showing the cognitive process triggered by a simple example of a static map reading task. An analyst is given the job of studying the relations between the (static) distributions of land-use, water-bodies and urban areas in a particular region.

If a think-aloud experiment was done during the beginning of this task, we might hear the analyst formulate sentences such as the following:

> OK . . . , here is this particular water-body and here is this city. Hem . . . , so the region in question is . . . grossly . . . this area. OK, we have this city here and this one here; this symbol represents the main-road network, this color is forests and this is . . . fields . . . OK, so now for my task . . .

Examining this fragment from the point of view of the Cognitive Load Theory, we could say that before going about his task, the analyst explores or gets acquainted with the content of the static map and that he stores elements of information in his working memory one by one.

Now, what lessons can we derive from this example regarding animated mapping, that is, for a visualization task where the information displayed does not remain static but evolves with display-time? In other terms, what might be the needs of an analyst in the preliminary phase of getting acquainted with an animated map?

First, the animated map reader needs to get acquainted with the *what* and the *where* dimensions of the phenomenon. Second, in an analogous way, he needs to get acquainted with the temporal dimension. Kraak and others call

this process building an "appropriate temporal schema [39]". To assist users in this task, animated mapping researchers recommend the use of temporal legends. We will present temporal legends more in detail further. Third the analyst will need to get familiar with the different types of events, processes and behaviors that are observable in the animation. To do so, he will need to see various sequences repeatedly and have time to make sense and to store information about these dynamic processes into his working memory. Once all these preliminary steps have been completed, the analyst can be expected to focus his attention, explore and try and understand more elaborate dynamics.

These steps point out at two basic needs in the preliminary phase of exploring a spatio-temporal dataset using the technique of map animation: a) the need for time to get acquainted and to make sense of what one is seeing and b) the need to understand the temporal characteristics of the representation of the phenomenon under study. To satisfy these two needs, two basic requirements for the design of animated maps have been pointed out in literature:

1. The need for temporal legends for the user to build an appropriate temporal schema. Kraak and others (Kraak, Edsall et al. 1997) state that "for users to understand a temporal animation, (...they must ...) apply an appropriate temporal schema that allows them to interpret meaning inherent in the sequence and pacing of the animation. They conclude that the "animated maps should be accompanied by a legend that prompts an appropriate schema," i.e., one or several temporal legends.

2. The need for interactivity for the user to have time or give himself time to get acquainted with the different aspects of the animated maps. Specifically for exploratory purposes, Andrienko and others [6] advance that ""pure" animation is insufficient for supporting exploratory analysis of time-referred data. An analyst should have at her/his disposal powerful and convenient facilities to control display time within the presentation."

Harrower and Fabrikant [30], as well as Slocum and others [72], report that both in literature recommendations and in animated mapping practices, the needs for interactivity and for temporal legends have been acknowledged and taken into consideration. Therefore, in the forthcoming sections, we will never make distinctions between animated maps with or without interactive controls and with or without temporal legends as we assume that in all cases where they are helpful, they are provided.

Appendix B

Explanations on Köbben's SVG scripted animations

As can be seen in the code below, the mechanism that makes the program dynamic is the `currVal + 1` expression of the `updateAnimation()` function. The '1' value can be changed to bigger or smaller values to make the animation slower or faster. Scripted animations can thus be built using the extended ISO 8601 format for time more easily than SMIL animations. However, as we said, SMIL animations remain more powerful and simpler to use.

```
function toggleAnimation() {
  if (running) {
    running = false;
    window.clearInterval(interval);
      //alert("stop");
  } else {
    running = true;
    interval = window.setInterval("updateAnimation()",500);
      //alert("start");
  }
}

function updateAnimation() {
  currVal = mySlider.getValue();
  currVal = currVal + 1;
  if (currVal > 24) {
    currVal = 0;
  }
  mySlider.setValue(currVal, false);
  showVal("change", "slider1", currVal);
```

Appendix C

Example of periodicity in an MCB visualization

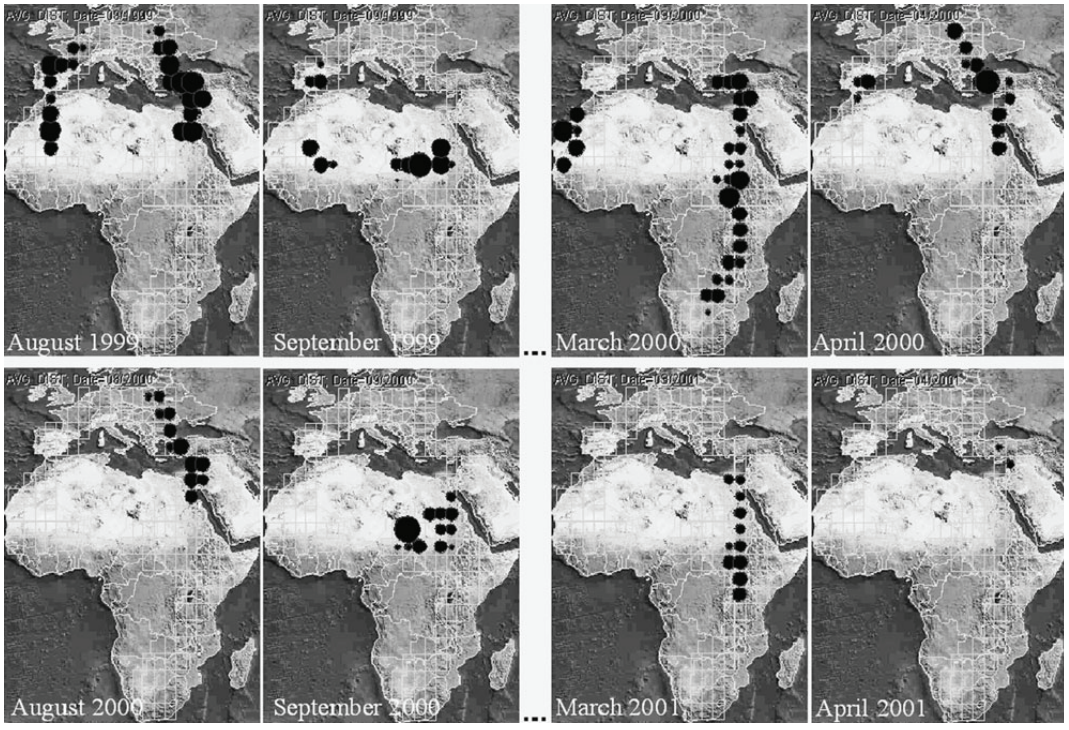


Figure C.1: MCB-based "Visualization of aggregated movement speeds of white storks during two migration seasons: 1999/2000 (top) and 2000/2001 (bottom)" (5)

This figure shows the results of an MCB-based visualization of bird migrations. It represents the distribution of the intensities of *aggregated movement speeds* of flocks of birds. The reader can see here an example of an MCB of another type than distribution.

Appendix D

SVG and JavaScript code

D.1 Introduction

The bits of code below are parts of the code used to develop our platform that may interest the reader. He shall not be surprised that most `<animate>` elements do not have any `begin` and dur+ values. The reason is that these are dynamically entered client-side – after the animations are loaded – when the user sets the value of the temporal scale...

D.2 Interpolated iceberg tracks

```
function IcebergTracks(){
for(var i=0; i<4; i++) {
setAnimations(i);
}
  }

// workaround for firefox bug (browser cannot execute function
getElementById if the element animate is unknown)
function myGetElementById(id)
{
  var element = document.getElementById(id);
  if(element==null && document.evaluate!=null)
  {
    element = document.evaluate(
    '//*[@xml:id="' + id + '" or @id="' + id + '"]',document,
    function(ns)
    {
        switch(ns) {
            case "xml": return "http://www.w3.org/XML/1998/
                namespace";
```

```
            case null: return null;
          }
    }, XPathResult.ANY_TYPE, null).iterateNext();
  }
  return element;
}

function setAnimations(number)
{
// IB_A35B_track instead of busTrack
// dashAnimA35B instead of dashAnim
// A35BAnim instead of busTrackAnim
IB_A35B_track=document.getElementById('IB_A35B_track'+number);
var A35BAnim=myGetElementById('dashAnimA35B'+number);
trackLength=IB_A35B_track.getTotalLength().toString();

IB_A35B_track.setAttributeNS(null,'stroke-dasharray',trackLength+',
        '+trackLength);
IB_A35B_track.setAttributeNS(null,'stroke-dashoffset',trackLength);
A35BAnim.setAttributeNS(null,'from',trackLength);
A35BAnim.setAttributeNS(null,'values',trackLength+';0');
 }

function setAnimIceberg(number){
    var IB35Bsegm=document.getElementById('IB35Bsegm'+number);
    var IB35BtrackAnim=myGetElementById('IB35BtrackAnim'+number);
    trackLength=IB35Bsegm.getTotalLength().toString();

    IB35Bsegm.setAttributeNS(null,'stroke-dasharray',trackLength+',
        '+trackLength);
    IB35Bsegm.setAttributeNS(null,'stroke-dasharray',trackLength);
    IB35BtrackAnim.setAttributeNS(null,'from',trackLength);
    IB35BtrackAnim.setAttributeNS(null,'values',trackLength+';0');
  }
```

D.3 Animating attribute changes

```
<g id="SMIL_animations" style='fill:lightblue; stroke:red;
    stroke-width:4'>
   <circle id='IB_A35B' cx='0' cy='600' r='25'>
     <animate id="SizAnimIB_A35B_0" attributeName="r"
        begin=""from="25" to="35" dur="" calcMode="linear"
        repeatCount="none" fill="freeze" />
     <animate id="SizAnimIB_A35B_1" attributeName="r"
        begin=""from="35" to="15" dur="" calcMode="linear"
```

```
        repeatCount="none" fill="freeze" />
     <animate id="SizAnimIB_A35B_2" attributeName="r"
        begin=""from="15" to="20" dur="" calcMode="linear"
        repeatCount="none" fill="freeze" />
     <animate id="SizAnimIB_A35B_3" attributeName="r"
        begin=""from="20" to="35" dur="" calcMode="linear"
        repeatCount="none" fill="freeze" />
   </circle>
</g>
```

D.4 Linear temporal legend

The code below is the implementation of the linear temporal legend.

```
    <g id="LinearTLeg">
     <rect id="future" x="299" y="637.5" rx="10" ry="10"
        width="401" height="25" style='fill:lime;
        stroke:darkgreen; stroke-width:2'/>
     <rect id="past" x="300" y="638" width="0" height="24.2"
        rx="10" ry="10" style="fill:orange; stroke:none">
        <animate id="animTBarPast" attributeType="XML"
            attributeName="width" from="0" to="400" dur=""
            fill="freeze" repeatCount="none" />
     </rect>
     <!-- Here below is actually the present symbol -->
     <rect y="632" width="18" height="36" rx="4" ry="4"
        fill="darkyellow"> <animate id="animTBarPresent"
        attributeName="x" from="293" to="692" dur=""
        fill="freeze" repeatCount="none"/>
     </rect>
    </g>
```

D.5 Pie-portion cyclic temporal legend

The code below is the implementation of the pie-portion cyclic temporal legend.

```
<g id="CyclicSMILfull" transform = "translate(100, 180)">
    <g transform="rotate(180)">
        <circle r="100" fill="#C9C299" stroke="green"/>
        <path id="past2" d="M -100 0 A 100 100 0 1 1 -100 0
            A 100 100 0 1 1 100 0" visibility="hidden"
            fill="#8A4117" stroke="#8A4117"  >
        <set attributeName="visibility" from="hidden"
        to="visible" begin="10s" fill="freeze"/>
```

```
    </path>
    <path id="past" d="M 100 0 A 100 100 0 1 1 -100 0 "
        fill="#8A4117" stroke="#8A4117">
        <animateTransform id="animCyclicLeg2"
            attributeName="transform" type="rotate"
            repeatCount="none" dur="10" by="180" />
    </path>
    <path id="past" d="M 100 0 A 100 100 0 1 1 -100 0 "
        fill="#C9C299" stroke="none"/>
    <path id="OnTop" d="M 100 0 A 100 100 0 1 0 -100 0 "
        visibility="hidden" fill="#8A4117"
        stroke="#8A4117" >
    <set attributeName="visibility" from="hidden"
        to="visible" begin="10s" fill="freeze"/>
    <animateTransform id="animCyclicLeg3"
        attributeName="transform"
        type="rotate" repeatCount="none" begin="10"
        dur="10" by="180" fill="freeze"/>
    </path>
    <g id="Hand" transform="rotate(-90)">
        <line stroke-width="20" y2="-100"
        stroke-linecap="round" stroke="red" />
        <animateTransform id="animCyclicLeg"
        attributeName="transform" type="rotate"
        repeatCount="none" dur="20" by="360" />
    </g>
    </g>
</g>
```

Bibliography

[1] Behaviour. In *Oxford English Dictionary*. Oxford University Press, Online version, second edition, 1989.

[2] W. Acevedo and P. Masuoka. Time-series animation techniques for visualizing urban growth. *Computers & Geosciences*, 23(4):423–435, 1997.

[3] Adobe Inc. SWF file format specification, Version 9. Technical report, Adobe Systems Incorporated, 2008.

[4] N. Andrienko and G. Andrienko. *Exploratory analysis of spatial and temporal data : a systematic approach*. Springer, Berlin etc., 2006.

[5] N. Andrienko and G. Andrienko. Designing visual analytics methods for massive collections of movement data. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 42(2):117–138, 2007.

[6] N. Andrienko, G. Andrienko, and P. Gatalsky. Supporting visual exploration of object movement. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI 2000)*, pages 217–220, Palermo, Italy, May 23-26 2000. ACM Press.

[7] N. Andrienko, G. Andrienko, and P. Gatalsky. Exploratory spatio - temporal visualization : an analytical review. *Journal of Visual Languages and Computing*, (14):503–541, 2003.

[8] J. Arlow and I. Neustadt. *UML 2 and the unified process : practical object - oriented analysis and design*. Object technology series. Addison Wesley, Upper Saddle River, USA, 2005.

[9] J. Ballantyne and D. G. Long. A multidecadal study of the number of antarctic icebergs using scatterometer data. In *Proceedings of the International Geoscience and Remote Sensing Symposium*, pages 3029–3031, Toronto, Canada, 2002.

[10] F.-J. Behr and H. Li. SUAS mapserver, An open source, SVG-oriented framework for extended Web Map Services. Nuremberg, Germany, http://www.svgopen.org/2008/papers/50-SUAS_MapServer__an_Open_Source_SVGoriented_Framework_for_extended_Web_Map_Services, 2008.

[11] J. Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. The University of Wisconsin Press, Madison, 1983.

[12] C. A. Blok. Monitoring change: characteristics of dynamic geo-spatial phenomena for visual exploration. In Christian Freksa, Reinhard Moratz, and Thomas Barkowsky, editors, *Spatial Cognition II*, volume 1849 of *Lecture Notes in Artificial Interlligence*, pages 16–30. Springer, Berlin, Heidelberg, 2000.

[13] C. A. Blok. *Dynamic visualization variables in animation to support monitoring of spatial phenomena*. ITC Dissertation 119. Universiteit Utrecht, ITC, Utrecht, Enschede, 2005. Doctorate thesis.

[14] D. DiBiase, A. M. MacEachren, J. B. Krygier, and C. Reeves. Animation and the role of map design in scientific visualization. *Cartography and Geographic Information Science*, 19(4):201–214, 1992.

[15] M. Dodge, M. McDerby, and M. Turner. *Geographic Visualization: Concepts, Tools and Applications*, chapter The Power of Geographical Visualizations, pages 1–10. John Wiley and Sons, 2008.

[16] S. Dodge, R. Weibel, and A.-K. Lautenschutz. Towards a taxonomy of movement patterns. *Information Visualization*, 7(3-4):240–252, 2008.

[17] D. Dorling and S. Openshaw. Using computer animation to visualize space-time patterns. *Environment and Planning B: Planning and Design*, 19(2):215–227, 1992.

[18] R. I. Dunfey, B. M. Gittings, and J. K. Batcheller. Towards an open architecture for vector gis. *Computers & Geosciences*, 32(10):1720–1732, 2006.

[19] R. Edsall and D. J. Peuquet. A graphical user interface for the integration of time into gis. Seatle, 1997. ACSM/ASPRS Annual Convention.

[20] S. Fabrikant. Towards an understanding of geovisualization with dynamic displays: Issues and prospects. In T Barkowsky, C Freksa, M Hegarty, and R. K. Lowe, editors, *Reasoning with Mental and External Diagrams: Computational Modeling and Spatial Assistance*, pages 6–11, Standford University, USA, 2005.

[21] S. Fabrikant and K. Goldsberry. Thematic relevance and perceptual salience of dynamic geovisualization displays. 2005.

[22] Geology.com. Map of antarctica and southern ocean. http://geology.com/world/antarctica-satellite-image.shtml.

[23] GeoServer. What is geoserver. http://geoserver.org/display/GEOS/What+is+Geoserver, Last accessed 14.01.2009.

[24] F. M. Goodchild. Geographic information science and systems for environmental management. In *Annual Review of Environment and Resources*, volume 28, pages 493–519. Annual Reviews, 2003.

[25] Google. Google earth 4.3, July 2008.

[26] A. L. Griffin, A. M. MacEachren, F. Hardisty, E. Steiner, and B. Li. A comparison of animated maps with static small-multiple maps for visually identifying space-time clusters. *Annals of the Association of American Geographers*, 96(4):740–753, 2006.

[27] M. Harrower. Animated and web-based maps. http://www.geography.wisc.edu/ harrower/Geog575/finalProjects03.html, 2003.

[28] M. Harrower. A look at the history and future of animated maps. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 39(3):33–42, 2004.

[29] M. Harrower. The cognitive limits of animated maps. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 42(4):269–277, 2007.

[30] M. Harrower and S. Fabrikant. The role of map animation for geographic visualization. In M. Dodge, M. McDerby, and M. Turner, editors, *Geographic Visualization: Concepts, Tools and Applications*, pages 44–66. John Wiley and Sons, West Sussex, UK, 2008.

[31] International Organization for Standardization (ISO). Iso 19128:2005, geographic information, Web Map Server interface. Standard, 2005.

[32] D. Jansen, M. Schodlok, and W. Rack. Basal melting of a-38b: A physical model constrained by satellite observations. *Remote Sensing of Environment*, 111(3):195–203, 2007.

[33] S. Kalyuga, P. Chandler, and J. Sweller. Managing split-attention and redundancy in multimedia instruction. *Applied Cognitive Psychology*, (13):351–371, 1999.

[34] B. Köbben. RIMapperWMS: a web map service providing svg maps with a built-in client. In S. Fabrikant and M. Wachowicz, editors, *The European Information Society, Leading the Way with Geo-information*, number XVIII in Lecture Notes in Geoinformation and Cartography, pages 217–230. Springer-Verlag, Berlin, 2007.

[35] B. Köbben. SVG and geo web services for visualization of time series data of flood risk. Nuremberg, Germany, 2008. SVG Open.

[36] B. Köbben, R. Lemmens, and A. Wytzisk. *A short Introduction to Geowebservices*. International Institute for Geo-Information Science and Earth Observation (ITC), Enschede, The Netherlands, 2008.

[37] A. Koussoulakou and M. J. Kraak. Spatio-temporal maps and cartographic communication. *The Cartographic Journal*, 29:101–108, 1992.

[38] M. J. Kraak. The space-time cube revisited from a geovisualization perspective. In *Proceedings of the 21st International Cartographic Conference*, pages 1988–1995, Durban, South Africa [CD-ROM], 2003.

[39] M. J. Kraak, R. Edsall, and A. M. MacEachren. Cartographic animation and legends for temporal maps : exporation and or interaction. In *Proceedings of the 18th ICA International cartographic conference*, volume 1, Stockholm, Sweden, 1997. International Cartographic Association (ICA).

[40] Kevin L. KevLinDev. http://www.kevlindev.com/gui/widgets/slider.

[41] Eric LaMar. *WMS Proposed Animation Service Extension (project, not standard)*. Open Geospatial Consortium (OGC), http://portal.opengeospatial.org/files/index.php?artifact_id=14698&passcode=1txw59j11ahnqh4ategh, version 0.9 edition, 2006.

[42] A. M. MacEachren. Time as a cartographic variable. In H. M. Hearnshaw and D. J. editor Unwin, editors, *Visualization in Geographical Information Systems*, pages 115–130. Wiley & Sons, London, 1994.

[43] A. M. MacEachren. *How Maps Work, Representation, visualization, and design*. The Guilford Press, New York, 1995.

[44] A. M. MacEachren, F. P. Boscoe, D. Haug, and L. W. Pickle. Geographic visualization: Designing manipulable maps for exploring temporally varying georeferenced statistics. In *Proceedings IEEE Information Visualization Symposium*, pages 87–94, North Carolina, USA, 1998.

[45] Map Server. Documentation. http://mapserver.org/documentation, Last accessed 14.01.2009.

[46] T. Midtbo, K. C. Clarke, and S. Fabrikant. Human interaction with animated maps: The portrayal of the passage of time. In *Proceedings, 11th Scandinavian Research Conference on Geographical Information Science*, As, Norway, 2007.

[47] M. Monmonnier. Strategies for the visualization of geographic time-series data. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 27(1):23–36, 1990.

[48] NASA. Available animations. http://www.nasa.gov, Last accessed 11.01.2009 2008.

[49] National Ice Center. Antarctic Icebergs. http://www.natice.noaa.gov/products/iceberg.

[50] National Ice Center. Iceberg d-15. http://www.natice.noaa.gov/pub/iceberg_images/jpeg/d15_352.jpg, Date of image: 17.12.2008 Last accessed on 14.01.2009.

[51] Natural Resources Canada. The calving of icebergs a-43 and a-44. http://www.ccrs.nrcan.gc.ca/radar/spaceborne/radarsat1/action/int/ant/video/calv00_a.wmv, May 2000. Ronne Ice Shelf, Antarctica.

[52] A. Neumann and A. Winter. carto.net. http://www.carto.net.

[53] A. Neumann and A. Winter. Time for SVG, towards high quality interactive web-maps. Beijing, China, http://www.carto.net/papers/svg/articles/paper_icc_congress_china_2001.pdf, 2001. 20th International Cartographic Congress.

[54] A. Neumann and A. Winter. SVG - scalable vector graphics, a future cornerstone of the WWW-infrastructure. 2002.

[55] A. Neumann and A. Winter. Example for path-morphing. http://www.carto.net/papers/svg/samples/path_morphing.shtml, Last accessed, 01.12.2009.

[56] A. Neumann, A. Winter, and Tirol Atlas. Webmapping with Scalable Vector Graphics (SVG): Delivering the promise of high quality and interactive web maps. In Peterson Michael, editor, *Maps and the Internet*, pages 197–220. Elsevier Science, Oxford, 2003.

[57] H. Nguyen Thi Phuong. Event-based clustering: a visual analytic tool for iceberg movement. MSc, International Institute for Geo-Information Science and Earth Observation, 2009.

[58] P. J. Ogao and M. J. Kraak. Defining visualization operations for temporal cartographic animation design. *International Journal of Applied Earth Observation and Geoinformation*, 4(1):23–31, 2002.

[59] OGC. *OpenGIS Web Map Server Cookbook*. Open Geospatial Consortium, Kris Kolodziej (editor), 1.0.2 edition, 2004.

[60] Open Geospatial Consortium (OGC). *OpenGIS Web Map Service (WMS) Implementation Specification 1.1.1*. http://www.opengeospatial.org/standards/wms, 2003.

[61] Open Geospatial Consortium (OGC). *OpenGIS Web Map Service (WMS) Implementation Specification 1.3.0*. http://www.opengeospatial.org/standards/wms, 2006.

[62] SVG Open. Abstracts and proceedings. Nuremberg, Germany, http://www.svgopen.org/2008/index.php?section=abstracts_and_ proceedings#paper_100, August 2008.

[63] Open Geospatial Consortium. OGC Website, 2008. last accessed: November 2008.

[64] Z.-R. Peng and M. H. Tsou. *Internet GIS: distributed geographic information services for the internet and wireless networks*. Wiley & Sons, Hoboken, 2003.

[65] Z.-R. Peng and Ch. Zhang. The roles of geography markup language (gml), scalable vector graphics (svg), and web feature service (wfs) specifications in the development of internet geographic information systems (gis). *Journal of Geographical Systems*, 6(2):95–116, 2004.

[66] M. P. Peterson. *Interactive and animated cartography*. Prentice Hall series in Geographic Information Science. Prentice Hall, Englewood Cliffs, USA, 1995.

[67] D. J. Peuquet. 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(3):441–461, 1994.

[68] R. A. Rensink. Internal Vs external information in visual perception. In *Proceedings of the Second International Symposium on Smart Graphics*, pages 63–70, Hawthorne, USA, 2002.

[69] T. Scambos, J. Maurer, R. Bauer, J. Bohlander, T. Haran, and K. Leitzell. A-CAP: The Antarctic Cryosphere Access Portal, OGC services. Boulder, Colorado USA: National Snow and Ice Data Center, http://nsidc.org/agdc/acap, Last accessed 12.02.2009.

[70] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pages 336–343, Washington, 1996. IEEE Computer Society Press.

[71] D. J. Simons and Ch. Chabris. Visual Cognition Lab, University of Illinois. http://viscog.beckman.uiuc.edu/djs_lab/demos.html, Last accessed 28.01.2009 1999.

[72] T. A. Slocum, R. B. McMaster, F. C. Kessler, and H. H. Howard. *Thematic cartography and geovisualization*. Prentice Hall series in Geographic Information Science. Prentice Hall, Upper Saddle River, USA, 3rd edition, 2009.

[73] Space, science and engineering center. Identify factors influencing iceberg motion. http://www.ssec.wisc.edu/sose/hi/tracking-icebergs5.html, Last accessed 14.01.2009.

[74] H. Stephen and D. G. Long. Study of iceberg B10A using scatterometer data. In *Proceedings of the International Geoscience and Remote Sensing Symposium*, volume 3, pages 1340–1342, Honolulu, 2000.

[75] Y. Tang and D. W. Wong. Exploring and visualizing sea ice chart data using java-based gis tools. *Computers & Geosciences*, 32:846–858, 2006.

[76] B. Veitch and C. Daley. Iceberg evolution modeling, A background study. Report, Memorial University of Newfoundland, St. John, Canada, May 2000.

[77] W3C. Scalable vector graphics (svg) 1.1 specification, animation. Technical report, World Wide Web Consortium, http://www.w3.org/TR/SVG/animate, 2008.

[78] W3C. Scalable vector graphics (svg) 1.1 specification, animation. Technical report, World Wide Web Consortium, http://www.w3.org/TR/SVG/animate, 2008.

[79] W3C. SMIL Animation. Technical report, World Wide Web Consortium, http://www.w3.org/TR/smil-animation, 2008.

[80] W3C. SVG 1.1 Specification, 11.5 Controlling visibility. Technical report, World Wide Web Consortium, http://www.w3.org/TR/SVG/painting.html#VisibilityControl, 2008.

[81] A. Watt, Ch. Lilley, D. J. Ayers, R. George, Ch. Wenz, T. Hauser, K. Lindsey, and N. Gustavsson. *SVG Unleashed : how to create and manipulate Scalable Vector Graphics SVG programming*, chapter Chapter 1: SVG Overview, pages 7–50. Sams, Indianapolis, 2003.

[82] A. Watt, Ch. Lilley, D. J. Ayers, R. George, Ch. Wenz, T. Hauser, K. Lindsey, and N. Gustavsson. *SVG Unleashed : how to create and manipulate Scalable Vector Graphics SVG programming*, chapter Chapter 11: SVG Animation Elements, pages 425–481. Sams, Indianapolis, 2003.