

LiU-ITN-TEK-A--18/023--SE

Multiperspective visualization of genealogy data

Anna Georgelis

2018-06-14



LiU-ITN-TEK-A--18/023--SE

Multiperspective visualization of genealogy data

Examensarbete utfört i Medieteknik
vid Tekniska högskolan vid
Linköpings universitet

Anna Georgelis

Handledare Katerina Vrotsou
Examinator Camilla Forsell

Norrköping 2018-06-14

Upphovsrätt

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida <http://www.ep.liu.se/>

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: <http://www.ep.liu.se/>

LINKÖPING UNIVERSITY

MASTER THESIS

Multiperspective Visualization of Genealogy Data

Author:
Anna GEORGELIS

Supervisor:
Katerina VROTSOU

June 19, 2018

Abstract

This thesis presents and discusses the implementation of a web application developed as a Master's degree project at Linköping University. The application is a tool offering a multiperspective visualization of genealogy data, that can be used by genealogists in order to analyze his or her collected family tree data, but also to find what data that may be wrong. Data stored in a GEDCom file is being processed and stored in a database. By using D3.js, the data is then visualized in three different types of representations: an ancestor tree, a sunburst chart and a lifeline representation, all interacting with each other. The work concludes that by using different types of visualizations to present the same data, it is possible to create a genealogy application where new kind of insights about the data can be gained.

Acknowledgements

First of all I want to thank my supervisor *Katerina Vrotsou* and my examiner *Camilla Forsell* for all their help, discussions and feedback during this thesis work. I also want to thank *Per Filipsson* and *Hjalmar Granberg* for contributing with interesting discussions, knowledge, ideas and feedback from a genealogical point of view.

To my family I want to say a big *thank you* for their continuous support through my studies. And last but not least, thanks to you *Bassam* for always being there for me!

Norrköping, June 2018
Anna Georgelis

Contents

Abstract	i
Acknowledgements	ii
Glossary	vii
1 Introduction	1
1.1 Aim	1
1.2 Problem Description	1
1.3 Research Questions	2
1.4 Limitations	2
1.5 Report Structure	2
2 Background	3
2.1 Genealogy	3
2.2 GEDCom	3
2.3 Related Work	5
2.3.1 Tree Drawing Algorithms	5
2.3.2 Time Focused Visualization In Genealogy	8
3 Method	15
3.1 Implementation	15
3.2 Data Parsing	16
3.3 Data Processing	17
3.4 Visualization	18
3.4.1 Ancestor Tree	18
3.4.2 Sunburst	20
3.4.3 Lifeline Representation	21
4 Results	22
4.1 Ancestor Tree	22
4.2 Sunburst	23
4.3 Lifelines	25
4.4 Colors and Interaction	27
4.5 Evaluation Meeting	28
5 Discussion	30
5.1 Ancestor Tree	30
5.2 Sunburst	32
5.3 Lifeline Representation	32
5.4 Colors and Interaction	34
5.5 Implementation	34

6	Conclusion & Future work	36
6.1	Research Questions	36
6.2	Future Work	37

List of Figures

2.1	A simplified GEDCom file.	4
2.2	An individual record from the sample GEDCom file.	5
2.3	A family record from the sample GEDCom file.	5
2.4	The same data set drawn in different formations of a binary tree. The numbers indicates the level of the nodes (where level 0 is the root). . .	6
2.5	Genalogy data visualized in different formations.	6
2.6	The same sample tree, created with the two algorithms presented by Wetherell and Shannon [1].	7
2.7	A sample tree drawn by different algorithms.	7
2.8	A lifeline visualization created with TimeNets [2].	8
2.9	A pedigree chart created with Genelines [3].	9
2.10	A full descendants chart created with Genelines [3].	10
2.11	A family group chart created with Genelines[3].	10
2.12	A direct line chart created with Genelines [3].	11
2.13	A fan chart created with Genelines [3].	11
2.14	An example of an ancestor tree created by Mukaliyev [4].	12
2.15	An example of a descendants tree created by Mukaliyev [4].	12
2.16	An example of a visualization created by Mukaliyev. Person 1 is the central person, and person 2,3 and 4 are the ancestors fulfilling the rule of having an only upgoing path of birth lines to the central person [4].	13
2.17	A lifeline where personal events has been added [4].	14
3.1	The three type of collections stored in the database.	16
3.2	An example of an hierarchical array, written with pseudocode.	18
3.3	Two versions of the same sample tree.	19
3.4	The drawing process of the nodes.	20
3.5	A comparison of the two sizing techniques. As can be seen the red node in the left image is much smaller than the same node colored in red in the right image.	21
4.1	The final result of the application.	22
4.2	An ancestor tree created with the application.	23
4.3	A pop-up window appears when hovering a person in the tree.	23
4.4	A sunburst chart created with the application.	24
4.5	The sunburst chart when hovering a person.	24
4.6	Multiple paths of ancestors can be highlighted.	24
4.7	If a node is missing data, it will be drawn in red.	24
4.8	A lifeline representation created with the application.	25
4.9	The lifelines sorted and colored based on their sex.	25
4.10	The lifeline representation when a line is being hovered.	26
4.11	The paths of ancestors highlighted in the sunburst chart, drawn in different colors.	26
4.12	The lines represents the average age of each century.	26

4.13	The default colors used in the application.	27
4.14	The colors used in special cases.	27
4.15	The three visualizations of the application interacting with each other.	27
4.16	The first version of the ancestor tree.	28
4.17	The different looks of the sunburst chart.	29
5.1	Two ancestor trees with their own characteristic shapes.	31
5.2	One of the ancestor has four parents according to the tree, due to an error in the database.	32
5.3	An error in the database found by the lifeline representation.	33
6.1	An example of how a descendants tree could look when hovering one of the ancestors.	38
6.2	A sample tree drawn along a timeline.	38

Glossary

Centralperson	The root person from who the visualizations are based on
Lifeline	A line that represents a person's life, where the length of the line corresponds to the life span of that person

Chapter 1

Introduction

Genealogy is a hobby whose popularity has grown rapidly in recent years. In America it is the second most popular hobby [5]. But what is it that makes genealogy that interesting? To find kinships with people you did not even know existed? Getting to know ancestors who lived hundreds of years ago? To learn about your name and how that name has developed throughout all generations? Or is it about getting insights about your family and how different historical events have affected your present life? There are a lot of possible reasons that may attract new groups of genealogy practitioners.

One major reason for the growing interest is the internet. With internet it is not only easier to collect information, keeping contact and sharing experiences with other genealogists, it is also easier to actually get a visual representation of the collected data. There is a lot of genealogy software helping you to create visualizations. Common for most existing genealogy software is the use of GEDCom (section 2.2), a file format in which genealogy data is stored.

1.1 Aim

Genealogy data is usually visualized with so called family trees. However, these trees are often difficult to navigate through, and do not present much more information than hierarchical relationships between people, and their birth and death dates. In a GEDCom file it is possible to store different types of data (for instance temporal or geographical data) that gets lost in the traditional visualizations. Tools presenting this kind of data would lead to a more vivid genealogy experience.

The aim with this thesis work is to examine how different visualization techniques could be used to enable intuitive navigation through a GEDCom based family tree, and how a multiperspective labeling of historical individuals can be used to gain new insights from genealogical visualizations.

1.2 Problem Description

At present, there are two problems mainly discussed regarding visualizing genealogy data. One is the difficulty of processing large data sets. A family tree containing a large number of people will be cluttered and hard to interpret and navigate through. The other problem is about visualizing temporal attributes. Currently, there are a lot of visualizations available showing hierarchical relations, but the number of examples including visualization of temporal data are sparse and usually limited to including the temporal information in the text labels.

After discussion with two experienced genealogists, it also emerged that a problem with genealogy today is the difficulty of getting an overview of where in one's family tree there is missing data, and what data may be wrong.

These problems lead to the research questions that this work will try to answer.

1.3 Research Questions

- When visualizing a large data set, how can a family tree be visualized to simplify the navigation in the tree?
- How can genealogy data be visualized from a temporal point of view?
- How can genealogy data be visualized to clearly indicate where there is missing data?

1.4 Limitations

The application developed in this work will only work as a tool to visualize and analyze the genealogy data stored in a GEDCom file. No functionality for adding new persons to the database, or to modify the already stored data, will therefore be implemented.

Due to time constraints, the application will focus on visualizing only the ancestors of a central person.

1.5 Report Structure

The structure of this report is as follows:

- Chapter 2 - Background: This chapter will initially present some history about genealogy. Then, the GEDCom format will be described, followed by related work regarding methods for creating a tidy drawing of a tree, and for visualizing genealogy data from a temporal point of view.
- Chapter 3 - Method: In this chapter the different methods used for creating the application will be presented.
- Chapter 4 - Results: The final application will be presented in this chapter.
- Chapter 5 - Discussion: In this chapter, the methods used, as well as the final result, will be discussed.
- Chapter 6 - Conclusion & Future work: A conclusion based on the aim and research questions of this work will be presented in this chapter. Also, possibilities for extensions and future work will be included here.

Chapter 2

Background

2.1 Genealogy

Genealogy, defined as the study of families and their origin [6], is a tool for proving kinship between people and is used for several purposes. Some people use genealogy to find living relatives, some use it for medical reasons (to find genetic diseases within the family), and some use it just as a hobby in order to create a family tree that ranges as far back as possible. The interest in genealogy has grown dramatically in recent years, making it the second most popular hobby in the United States [5]. However, the idea of genealogy has not always been as positive. As a result of the American Revolution, politics in the country became unstable, and the usually huge respect for ancestors decreased. For many, genealogy was seen as something elitist. About 100 years later, after the Civil War, the United States began to regain its stability and prosperity. This resulted in a nation attracting a wave of immigrants, many of whom were met with hostility. Nativism spread throughout the country. During this time, genealogy became a tool for ideologies rooted in concepts such as heredity and race. [7]

The publication of the book *Roots: The Saga of an American Family* [8] by Alex Hayes in 1976 is said to be the start of the great interest in genealogy. The book follows the life of Kunta Kinte, an African youth sold and abducted to North America for slavery, and the lives of his descendants. The book convinced people around the world that each family has its own important story to tell. Regardless of origin, race or prosperity, it became respectable to search for their ancestors. [7]

With internet, genealogy became more accessible. The internet facilitated the process of collecting data and it also became easier to both keep contact with other genealogists and to create family trees. This made the whole genealogy process easier, and is one of the main reasons why the interest in genealogy has grown that much in recent years. [5] Today there are several genealogy software where users can store their family information as well as visualize the data (commonly with a family tree). Many of these software have in common that they use the GEDCom file structure (section 2.2), a data representation for genealogy data, which enables exchanging data between them.

2.2 GEDCom

GEDCom, short for GENEalogical Data Communication, is a data representation for genealogy data, developed and presented by The Church of Jesus Christ of Latter-day Saints. [9] The GEDCom data format is used to exchange genealogical data between different genealogy software.

0 HEAD	Header
1 GEDC	
2 VERS 5.5	
1 CHAR ASCII	
1 SOUR MYHERITAGE	
0 @I1@ INDI	Individual Records
1 NAME John /Doe/	
1 BIRT	
2 DATE 01 JAN 1950	
2 PLAC Stockholm	
1 DEAT	
2 DATE 01 JAN 2010	
2 PLAC Stockholm	
1 FAMS @F1@	
0 @I2@ INDI	
1 NAME Jane /Doe/	Family Records
1 BIRT	
2 DATE 01 JAN 1950	
1 FAMS @F1@	
0 @I3@ INDI	
1 NAME Johnny /Doe/	
2 DATE 01 APR 1980	
1 FAMC @F1@	
0 @F1@ FAM	
1 HUSB @I1@	
1 WIFE @I2@	
1 CHIL @I3@	
0 TRLR	

FIGURE 2.1: A simplified GEDCom file.

A GEDCom file usually consists of three parts: the header, the individual records part and the family records part. Figure 2.1 shows a sample from a GEDCom file. The header includes basic information about the file, for example the name of the software source (SOUR), the GEDCom version (5.5) and the character encoding (ASCII).

Both the individual records part and the family records part contain a stream of related records. Every record is described by a series of tagged lines hierarchically organized, where every line consists of a level number, a tag and an optional value. A line with the level number 0 is always indicating that it is the first line of a new record. Every line can have a subordinate line, whereof the level number refers to their hierarchical relationship. A subordinate line always has a level number increased by 1. In figure 2.2, an example of an individual record from a GEDCom file is presented.

The first line of the individual records part means that this is the record describing a new individual (INDI) that, in the example of figure 2.2, has been given the identification number "I1". The second line has the level number 1, the tag NAME and the value John /Doe/. Thus,

the individual I1 has the name John Doe.

Lines 3-5 in the example describe the birth (BIRT) of John Doe. The lines "2 DATE 01 JAN 1950" and "2 PLAC Stockholm" both have the level 2 which means that they are subordinate lines of "1 BIRT". The tag PLAC describes a place and DATE is the date for an event. Thus, John Doe was born on January the 1st, 1950, in Stockholm. Accordingly, lines 6-8 describe the death (DEAT) of John Doe.

The last line, 1 FAMS @F1@, describes that John Doe belongs to the family with the given identification number "F1". An individual can be linked to a family either by the tag "FAMS" or the tag "FAMC", where FAMS means that the individual is one of the spouses or parents of the family and FAMC that the individual is a child of the family. Since a person can be both a parent and a child, it is possible to be linked to more than one family.

A family record example can be seen in figure 2.3. In a family record there are information about who belongs to the family. A mother (WIFE), a father (HUSB) and children (CHIL) are connected to the family by their identification numbers. Every family record is also given an individual identification number. Except for the information presented in the figure, a family record can for instance also contain information about marriage between the parents.

Every GEDCom file ends with the line 0 TRLR. In the GEDCom Standard [9], all tags that can be used in a GEDCom file are defined.

```

0 @I1@ INDI
1 NAME John /Doe/
1 BIRT
2 DATE 01 JAN 1950
2 PLAC Stockholm
1 DEAT
2 DATE 01 JAN 2010
2 PLAC Stockholm
1 FAMS @F1@

```

FIGURE 2.2: An individual record from the sample GEDCom file.

```

0 @F1@ FAM
1 HUSB @I1@
1 WIFE @I2@
1 CHIL @I3@

```

FIGURE 2.3: A family record from the sample GEDCom file.

2.3 Related Work

This chapter will start with an overview of how a binary tree can be used for visualizing genealogy data. Tree drawing algorithms will then be presented, followed by a review of time focused visualization approaches used in genealogy.

2.3.1 Tree Drawing Algorithms

The probably most common way to visualize genealogy data is with a family tree. Trees are in general a typical data structure for presenting hierarchical data. A tree is built of nodes, and lines connecting the nodes. The nodes are representing the objects in the data set, and the lines demonstrate the hierarchical relationships between the objects. [10] In many cases, a specific node is the root of the tree. The tree will in those cases be called a rooted tree. When drawing a rooted tree some aesthetic rules are often adopted. To place nodes at the same level along the same horizontal line is an example of an aesthetic rule. [11] A common form of a rooted tree is the binary tree. In a binary tree, each node has at most two child nodes. Figure 2.4a shows a classic formation of a binary tree. In this figure the aesthetic rule of placing nodes at the same level along the same horizontal line is being adopted. This kind of binary tree is common when visualizing genealogy data. Figure 2.5a shows an example of how a binary tree can be used to create an ancestor tree.

A binary tree can be drawn in other formations than the tree presented in figure 2.4a. One example is the H-tree, figure 2.4b. Compared to the “classical” layout of a binary tree the H-tree is using more of the drawing surface, which Tuttle, Nonato and Silva [12] consider being an advantage when visualizing genealogy data. Figure 2.5b shows how Tuttle, Nonato and Silva use the H-tree to create a pedigree chart. The central person (the root node) is placed in the middle. The parents of the central person are then placed on each side of the central person, either in the vertical or the horizontal direction. The next generation of ancestors (the grandparents of the central person) are placed in the same manner, at each side of their respective children. The horizontal or the vertical direction of the placement is alternating between the generations. This process continues until all persons in the data set are drawn. The nodes will then build a fractal pattern, similar in shape to the letter H.

When visualizing hierarchical data, radial layouts can also be used. Instead of drawing the nodes along horizontal lines, they are being drawn along concentric circles with the root placed in the middle. Nodes at the same level are placed on the same concentric circle. This can be seen in figure 2.4c. Radial layouts have also been used when visualizing genealogy data. An example is shown in figure 2.5c.

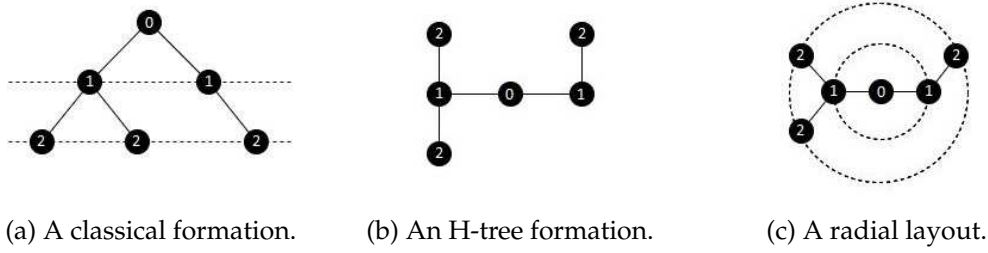


FIGURE 2.4: The same data set drawn in different formations of a binary tree. The numbers indicates the level of the nodes (where level 0 is the root).

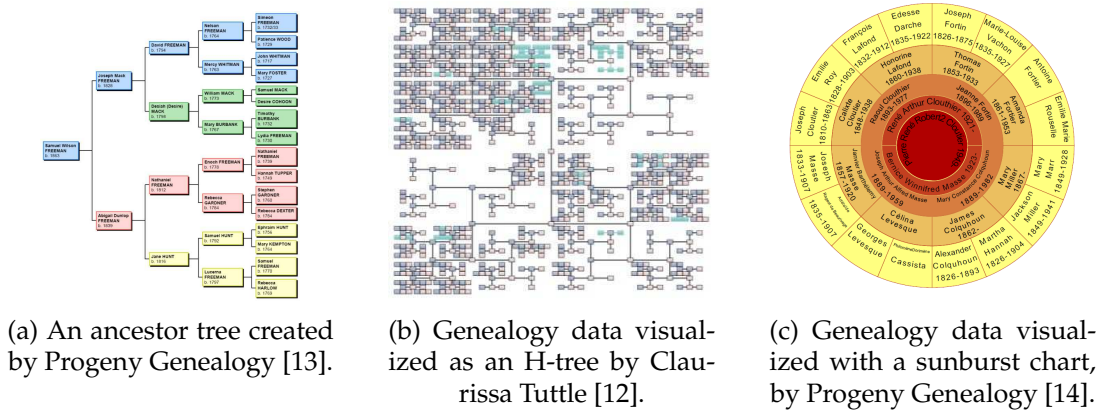


FIGURE 2.5: Genealogy data visualized in different formations.

Wetherell and Shannon discuss the difficulties of creating a tidy drawing of a tree, and present two algorithms in an attempt to solve these problems [1]. These algorithms are adjusted for binary trees, but can be modified in order to present an arbitrary tree. According to Wetherell and Shannon, a tidy tree needs to fulfill both physical and aesthetic requirements. Since the drawing surface is usually bounded in one or two dimensions the tree cannot use unlimited of space, and should therefore not be too big. Because of this, they implemented a physical limit in their algorithms that tries to draw the tree with a width as small as possible. The aesthetic requirements that a tidy tree needs to fulfill according to Wetherell and Shannon are as follows:

- Nodes of a tree at the same height should lie along a straight line, and the straight lines defining the levels should be parallel
- In a binary tree, each left son should be positioned left of its father and each right son right of its father
- A parent should be centered over its children.

Figure 2.6 shows two drawings of the same sample tree, one for each of the resulting algorithms. The left tree, figure 2.6a, fulfills all of the aesthetic requirements, but it does not have the smallest width possible. Because of this, Wetherell and Shannon modified the algorithm, resulting in the right tree, figure 2.6b. The difference between the algorithms is that in the modified algorithm, subtrees are being folded beneath their ancestors where possible. This will result in a narrower tree. On the contrary, it will no longer fulfill the last aesthetic rule. As can be seen in figure 2.6b,

the parent of node A is not placed in the center of its children, which will give a less tidy tree according to aesthetics.



(a) A sample tree fulfilling the aesthetic rules defined by Wetherell and Shannon.

(b) A narrower version of the same sample tree, created with the modified algorithm.

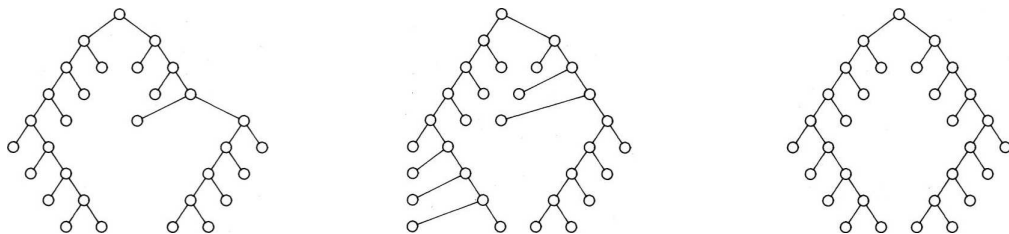
FIGURE 2.6: The same sample tree, created with the two algorithms presented by Wetherell and Shannon [1].

The problems occurring when drawing a tree with the algorithms presented by Wetherell and Shannon are discussed by Reingold and Tilford [15]. They conclude that the problems are a consequence of that, when drawing a tree with these algorithms, the shape of a subtree is affected by the position of the nodes outside that subtree. This will result in that a symmetric tree may be drawn asymmetrically, that is, a tree and its reflection will not always produce mirror image drawings. That a symmetric tree is being drawn asymmetrically is not a desirable behaviour, and to prevent that this will happen, Reingold and Tilford formulate a fourth aesthetic rule that should be fulfilled when drawing a tidy tree:

- A tree and its mirror image should produce drawings that are reflections of one another; moreover, a subtree should be drawn the same way regardless of where it occurs in the tree.

As an improvement to the Wetherell and Shannon algorithms, they present a new algorithm where the fourth aesthetic rule, as well as the three former rules, are fulfilled. In figure 2.7, a sample tree drawn by the three different algorithms can be seen.

Even though the algorithm presented by Reingold and Tilford produces tidy and pleasing drawings, the trees are not drawn with the smallest width possible. However, Reingold and Tilford consider the fourth aesthetic rule being of bigger importance than minimum width, since the aesthetic rule will aid in human perception.



(a) Drawn by the first algorithm presented by Wetherell and Shannon [15].

(b) Drawn by the modified algorithm presented by Wetherell and Shannon [15].

(c) Drawn by the algorithm presented by Reingold and Tilford [15].

FIGURE 2.7: A sample tree drawn by different algorithms.

2.3.2 Time Focused Visualization In Genealogy

TimeNets, proposed by Nam Wook Kim et al [2], is a visualization tool where hierarchical conditions are combined with a timeline, for presenting a family tree with a temporal context. In addition, TimeNets includes a technique for processing large data sets. In this technique, every individual in the data set is assigned a degree-of-interest. Based on this degree-of-interest the individual is either included or excluded from the visualization.

The visualizations in TimeNets are based on a horizontal timeline representing time continuing from left to right, figure 2.8. Each individual is represented with a lifeline. These lines are placed along the timeline so that the left end is placed at the time of the particular person's birth and the right end is placed at the time of the particular person's death. To visualize different kinds of relationships between individuals the vertical axis is used. A marriage between two persons is represented by their lifelines converging. Likewise, a divorce is represented by two lifelines diverging.

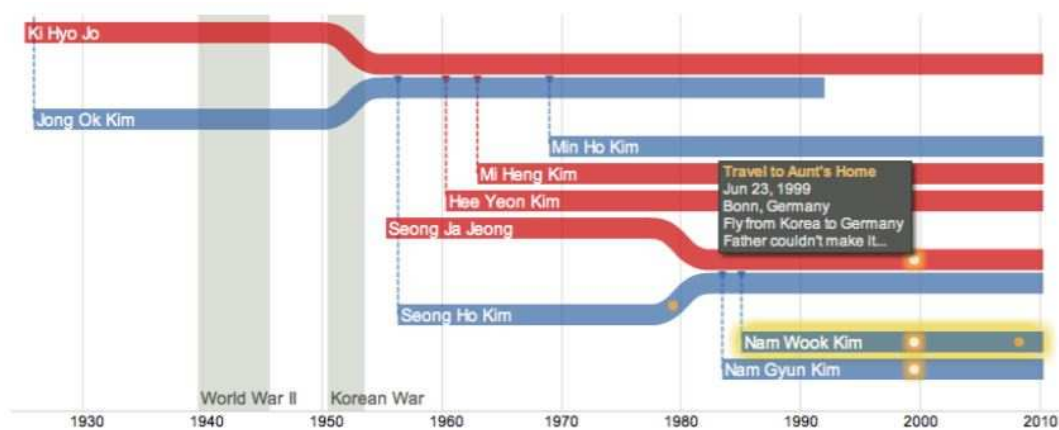


FIGURE 2.8: A lifeline visualization created with TimeNets [2].

To indicate a parent-child-relationship, TimeNets uses a drop line that connects the parents' lifelines with the child's lifeline. These lines are dotted and transparent in order to prevent the visualization from getting too cluttered. In those cases where a parent has more than one child, the children will be ordered vertically based on their birth dates. The youngest child will be placed the closest to their parents' lines, and the oldest the most far away. By sorting the lines like this, the number of intersecting lines will be minimized, which as well results in a less cluttered visualization.

As a default, the lifelines will be colored based on sex, blue for male and red for female. However, this can be changed to make the colors represent other data, e.g. an individual's geographical settlement. The colors will then represent different geographical locations, and the line will shift colors depending on when the represented person moved to another location. Another example of alternative color coding is to detect an individual's various diseases, when they occurred and how long they kept.

As mentioned in section 1.2, one of the main problems when visualizing genealogy data is the difficulty of processing large data sets. When the visualization includes a large amount of people it will easily become cluttered and hard to interpret. To improve this, TimeNets is using a *degree-of-interest* (from now on called DOI) technique. A DOI will be calculated for each individual, and compared to a threshold value. If the DOI of an individual is larger than the threshold, he or she will be included in the visualization, otherwise not. Based on the central person, the DOI of

all individuals in the data set is being calculated. The central person gets the highest DOI. The DOI will then decrease for each individual in relation to the distance to the central person. For people related to the central person, the DOI decreases linearly, while for marital relationships, it decreases more slowly.

Unlike a classical, static layout of a family tree, the TimeNets approach offers a tool for analyzing the data and gaining new kind of insights. By presenting the temporal data in a clear way, the user will “get to know” the family and their history better. Another strength of the visualizations created by TimeNets is the ability to easily present divorces and remarriages. In today’s society this is very common, hence, tools offering support for clearly presenting divorces and remarriages are highly relevant.

Despite its strengths, TimeNets is not an optimal tool in all situations. When analyzing only a few individuals closely related, as in the example presented in figure 2.8, it works great. In other situations though, where the aim may be to get an overview of the whole family, or comparing two individuals placed far from each other due to differences in birth years, it will not work as good.

Genelines [3] is another software that visualizes genealogy data along a timeline. It is possible to create five different kinds of visualizations presenting hierarchical relationships with Genelines: a pedigree chart, a full descendant chart, a family group chart, a direct line chart and a fan chart. The first four visualizations have in common that every individual, just like with TimeNets, is represented by a lifeline.

All visualizations have their own main task. The pedigree chart, figure 2.9, draws the central person together with its ancestors. Like TimeNets, parents are paired with their child using a dashed vertical line. In contrast, instead of placing the child beneath its parents, Genelines places the child between its parents. The colors and the placement of the lifelines indicate if the individual is a maternal or paternal ancestor. A maternal ancestor will be drawn in red beneath the central person, while a paternal ancestor will be drawn in blue above the central person. In those cases where there is no information about an individual’s birth or death date, Genelines will estimate that date. An estimated date is represented by the lifeline not being completely filled with color, either at the beginning or end of the line, depending on whether there is the birth or death date missing.

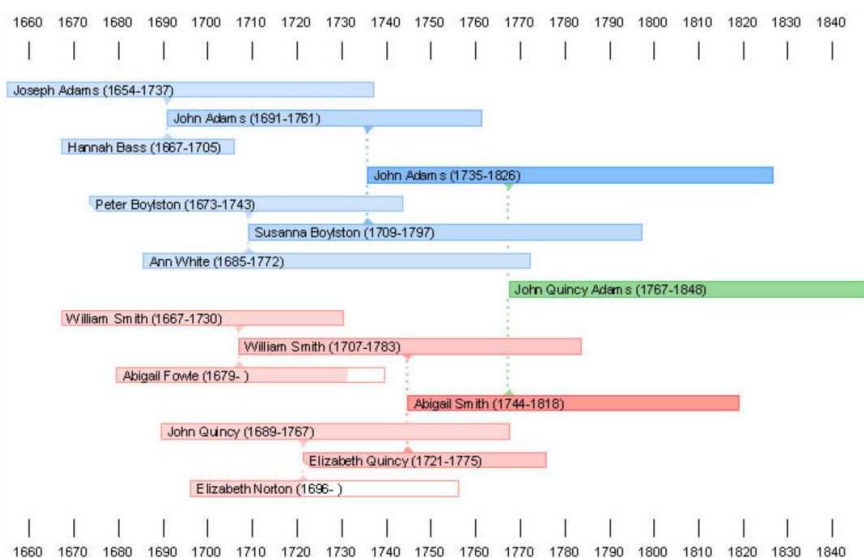


FIGURE 2.9: A pedigree chart created with Genelines [3].

The full descendant chart presents all descendants of the central person. The central person is drawn at the top together with his spouse, colored in green, figure 2.10. Every child to the central person is drawn, together with their respective descendants, in different colors.

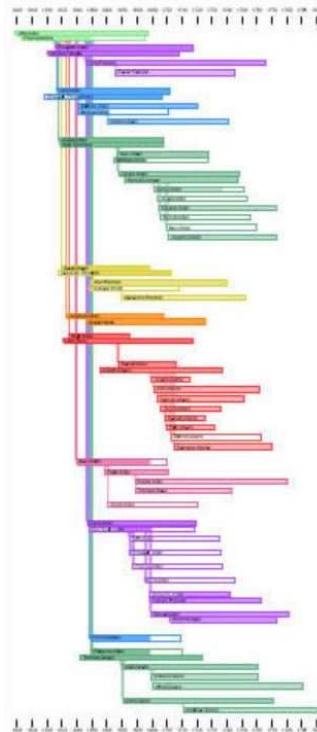


FIGURE 2.10: A full descendants chart created with Genelines [3].

The family group chart visualizes lifelines of the central person and his children, spouse and parents, figure 2.11. In this visualization the lifelines are colored blue for men and red for women. It is also possible to add historical events important to the family in the family group chart. They are visualized as vertical bands.

The direct line chart connects a person together with a chosen descendant or ancestor by showing only those people creating the path between these two, figure 2.12. As well as with the family group chart, it is possible to add historical event in the direct line chart.

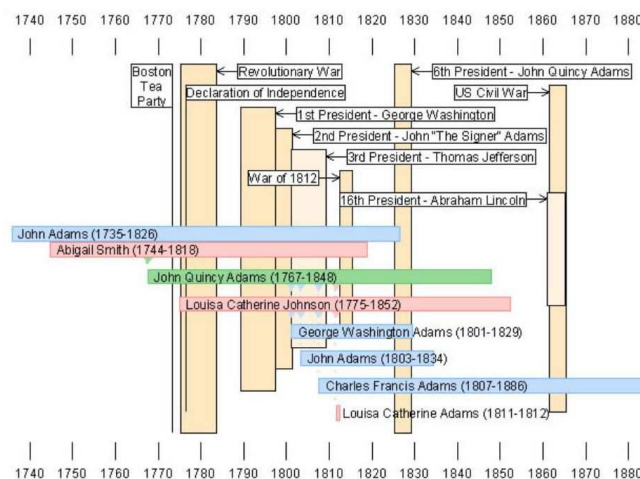


FIGURE 2.11: A family group chart created with Genelines[3].

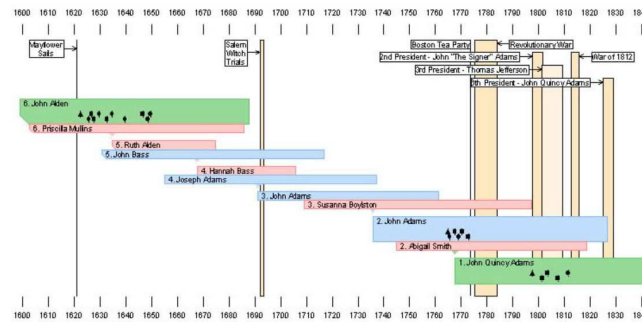


FIGURE 2.12: A direct line chart created with Genelines [3].

Finally, it is possible to create a fan chart. Similar to the other visualizations, the fan chart is based on a timeline. The central person is drawn in green in the middle, paternal ancestors are drawn on the left side and maternal ancestors are drawn on the right side of the central person.

One advantage with Genelines is that it is possible to create different visualizations of the same data. Comparing the charts, different types of insights can be gained from each chart. Hence, Genelines as a tool is broad, and covers several different needs. However, the different charts are neither linked together or visualized at the same time. Even more kind of insights could have been gained by the user if the charts were linked together and interacted with each other.

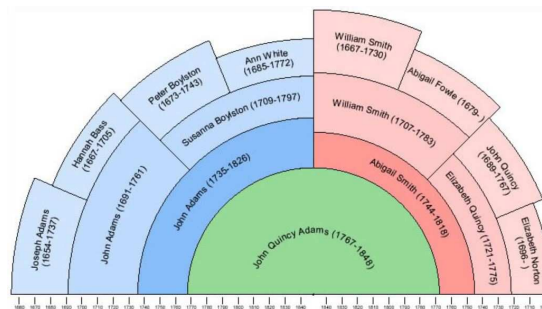


FIGURE 2.13: A fan chart created with Genelines [3].

Mukaliyev presents a new method for visualizing genealogy data [4]. The method is based on an attempt to combine methods for handling large data sets along with methods for visualizing data against a timeline.

Like TimeNets and Genelines, Mukaliyev as well uses lifelines placed along a horizontal axis to represent individuals. The proposed solution consists of two merged trees, an ancestor tree and a descendant tree. For the ancestor tree, the same technique as Genelines is used: each individual's lifeline is placed vertically between its parents' lifelines, figure 2.14. The descendant tree at the other hand uses the same technique as TimeNets where a child's lifeline is placed underneath both parents' lines, figure 2.15.

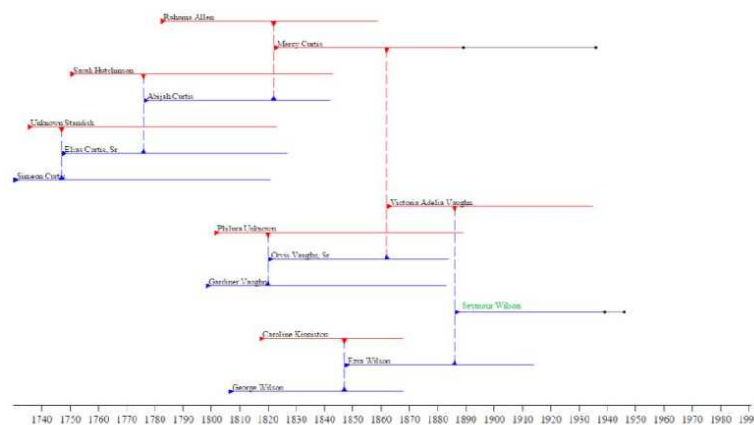


FIGURE 2.14: An example of an ancestor tree created by Mukaliyev [4].

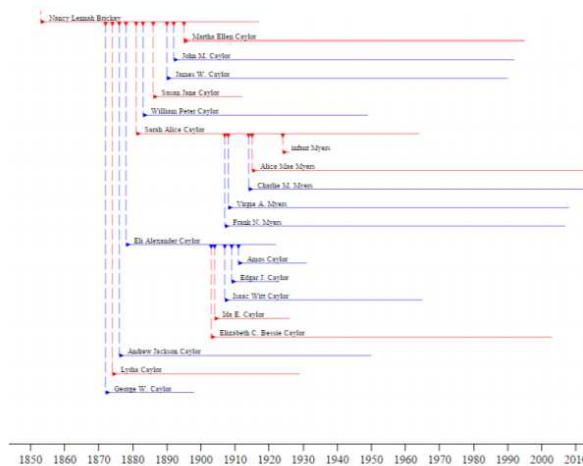


FIGURE 2.15: An example of a descendants tree created by Mukaliyev [4].

A visualization where every individual in the data set is included will be hard to understand, especially when it comes to large data sets. It would also be hard to avoid intersecting lines which makes the visualization even harder to interpret. To avoid this, Mukaliyev chooses to only draw a part of the data, and let the user have the possibility to affect which part is shown. Before any tree is drawn the user chooses which person in the data that should be the central person. The ancestor tree is drawn first, containing all ancestors of the central person. All ancestors have their own descendant tree, but in order to avoid a cluttered visualization all descendant trees cannot be drawn. Therefore, Mukaliyev introduced a rule saying that only the descendants belonging to an ancestor whose path of birth lines to the central person is only going upwards, will be drawn. This rule is further explained in figure 2.16. Person 1 is the central person, and person 2, 3 and 4 are the ancestors fulfilling the rule of having an only upgoing path of birth lines to the central person. Thus, a descendant tree will only be drawn for person 2, 3, and 4. This rule will result in that no birth line intersects with any lifeline.

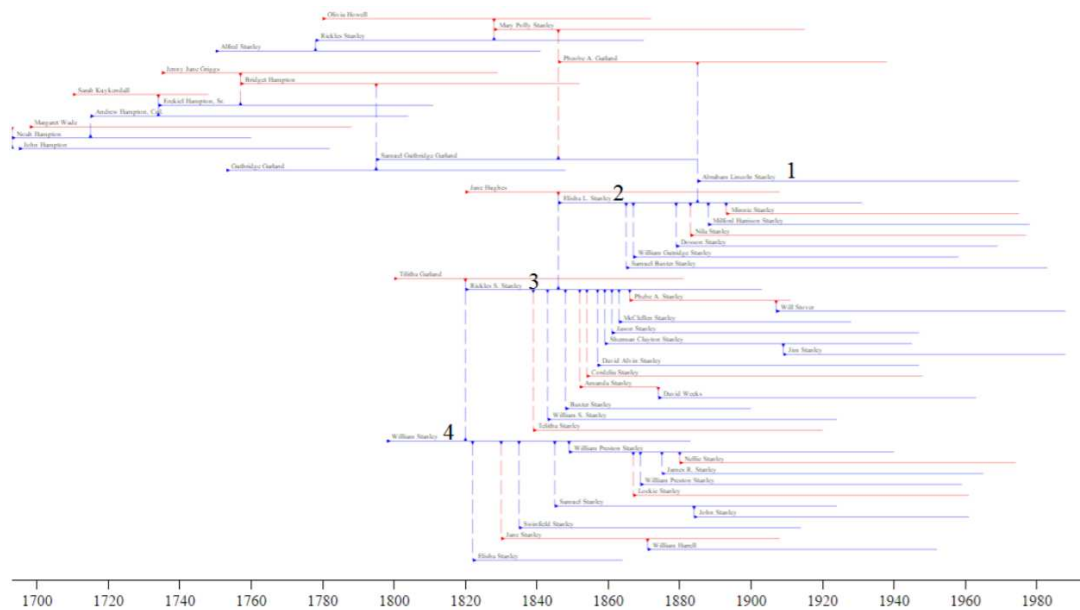


FIGURE 2.16: An example of a visualization created by Mukaliyev. Person 1 is the central person, and person 2,3 and 4 are the ancestors fulfilling the rule of having an only upgoing path of birth lines to the central person [4].

However, this does not mean that a descendant tree cannot be drawn for any of the other ancestors. By clicking on an optional ancestor, the tree will be redrawn, but with the greatest ancestor of the chosen ancestor as the lowermost person in the visualization. Thus, the ancestors fulfilling the rule will now be others than before. As a default, the greatest paternal ancestor will be the lowermost ancestor in the visualization.

If the user clicks on a lifeline that does not belong to an ancestor of the central person, the person whose lifeline was clicked will be the new central person. Both the ancestor and the descendant tree will then be redrawn based on the new central person. Makuliyev thus offers the user a great impact on what data is being drawn.

Even though only some parts of the data is drawn, the visualization can still become cluttered and contain a large amount of lines. Because of this Mukaliyev introduced techniques for zooming and panning. The zooming is only working in vertical mode, which means that the user can increase the vertical distance between the lifelines. The user can also drag the visualization around without zooming and thus affect which part of the visualization that is visible.

The colors of the lifelines depend on the gender, blue for male and red for female. Personal events, such as diseases, can be added to an individual. This is visualized by coloring the part of the lifeline corresponding to when the event took place with another color. At both ends of the event, a dot is drawn in the same color (see figure 2.17). This will help the user to keep track of when different events start and end in cases where different events overlap on time.

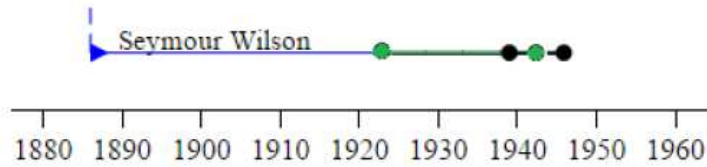


FIGURE 2.17: A lifeline where personal events has been added [4].

One of the strengths of Mukaliyev's approach is the rules deciding the placement of the lines, and which descendants that are being drawn. Thanks to these rules, the number of intersecting lines will be minimized, which is of big importance if the visualization should not become too cluttered. Another strength is that Mukaliyev offers the user a great impact on which data is being drawn. Since the user has the ability to change which data is being drawn, it does not matter that all data cannot be drawn at the same time.

Like the TimeNets approach, Mukaliyev's approach is not suitable in situations where the user wants to compare two individuals placed far from each other or wants to get an overview of the whole family. Visualizing a large data set, there is a risk that the resulting visualization will become cluttered due to the amount of lines.

Chapter 3

Method

This chapter will describe the approach and the different techniques used to create a tool for visualizing genealogy data.

Two experienced genealogists were involved during this thesis work. Before any of the implementation was done, continuous meetings were held in order to discuss some difficulties regarding genealogy today, as well as different techniques, representations, functionalities and layouts. After a first version of the application was implemented, there was also an evaluation meeting held where the result was discussed and the genealogists gave their feedback of what could be improved.

The result of this thesis has been a web application built using the MEAN-stack. MEAN stands for MongoDB [16] (a NoSQL database that stores its data in a JSON-like format), Express [17] (a flexible back end web application framework that runs on top of Node.js), AngularJS [18] (a front end web application framework) and Node.js [19] (an asynchronous event driven server framework) which are all JavaScript-based technologies. The MEAN-stack has been used due to its strengths in building fast, efficient and well structured applications that are easy to maintain. On top of the MEAN-stack D3.js [20] has been used, a JavaScript library for creating powerful visualizations.

3.1 Implementation

A database was created in the application for holding the GEDCom data. The database consists of three different collections: one collection containing all individuals, one containing all families, and one containing all events. In order to simplify the handling and structuring of the objects in the database, Mongoose [21] was used. Mongoose is an object modeling tool for MongoDB. With Mongoose a Schema is created for each collection, defining what information is stored in every object of the collection [22]. The Schemas created in the application are called Indi, Family and Event, and are defined according to figure 3.1. In the Event collection an object stores all events related to a specific person. In this application, the only event data saved is about birth, death and marriages, since this is the only temporal data needed for the implemented functionality (presented in section 4.3). The Events database can be expanded with information about other events as well as information about where a specific event took place, since this is data that can be stored in a GEDCom file.

```
var Indi = new Schema({
  id: {type: String},
  name: {type: String},
  sex: {type: String},
  fams: {type: String},
  famc: {type: String}
});
```

(a) A schema defining an object in the Indi collection.

```
var Family = new Schema({
  id: {type: String},
  mother: {type: String},
  father: {type: String},
  children: [String]
});
```

(b) A schema defining an object in the Family collection.

```
var Event = new Schema({
  id: {type: String},
  birth: {type: String},
  death: {type: String},
  marriage: {type: String},
  children: [{id: {type: String}, birth: {type: String}}]
});
```

(c) A schema defining an object in the Event collection.

FIGURE 3.1: The three type of collections stored in the database.

NodeJS and MongoDB are asynchronous. The difference between synchronous and asynchronous programming is that with synchronous programming, the code is executed in the same order in which it is written, from top to bottom. When a function is executed, the program waits for it to finish before moving on to the next part of the code. Asynchronous code on the other hand, does not have to wait for a function to run to completion, but several statements can be executed simultaneously. The flow therefore does not become sequential, where the code is executed from top to bottom. The advantage of writing asynchronous code is that the program gets faster and more efficient. Functions that are independent can be run in parallel which will save time. [23] However, in those cases when several asynchronous functions depend on the result from each other, it may cause difficulties. Asynchronous functions use callbacks to define what will happen when the function completes. Combining multiple asynchronous functions leads to nested callbacks, which are difficult to handle. The utility module `async` [24] was used in this project, in order to simplify the handling of the asynchronous callbacks. The method `async.waterfall` was used in order to handle the flow in the code. `Async.waterfall` works in such a way that an array of asynchronous functions are executed sequentially, and each function sends its result to the next function of the array [25].

Express is a web application framework for Node.js, which contributes to faster and better structured code. In this application it was used for setting up a local web server and to simplify the routing. When receiving an incoming request in the form of an URL from the client (in this case from AngularJS), Express redirects the requests to a certain part of the code. That code will then be executed, and a response is sent back to the client. [26]

3.2 Data Parsing

The data in the GEDCom file needed to be parsed in order to be saved in the database. This was done by creating a JavaScript script that would read the file line by line and process its data. Since GEDCom files often contain data about a large number of people they become very large, which can cause problems for the application trying to read it. The application only has a limited amount of memory allocated, and for each line read some of that memory is being used. After reading a large number of lines the application will run out of memory and therefore stop working. To avoid

this, the proposed application divides the sent in GEDCom file into smaller parts, reads one part, processes its data, and then throws it away before continuing to read the next part. This means that memory will only be allocated for one part at the time, hence, the application wont run out of memory.

The data was processed in order to be saved in the three different collections, defined in figure 3.1. As described in section 2.2, each individual and family have a unique identification number that is used in order to connect individuals to a specific family or event. Each person can be connected to multiple families, either as a parent (tag = FAMS) or as a child (tag = FAMC).

3.3 Data Processing

In order for the data stored in the database to be visualized, it first needs to be processed. All persons in the data set will not be drawn simultaneously, and therefore the application needs to know which data to draw. At present, due to time constraints, the application is focused on visualizing only the ancestors of the central person (in section 6.2 there is a discussion about how the application could be extended to visualize descendants as well). Based on the central person, an array is created containing all the ancestors of that person. The visualizations created by the application are based on the data stored in that array. It is possible for the user to change the central person (this is described in section 4.1). A new array will then be created before redrawing the visualizations.

The array is created using a recursive function. The function works as described below:

1. Add the central person to the array
2. Examine if the person at the current position of the array has a mother and/or a father stored in the database
3. If yes, add them to the array
4. If this is the last position of the array, return and save the array. Otherwise, continue to the next position of the array, then repeat step 2-4.

By repeating these steps, all ancestors will be added to the array. Hence, the application will know which data to visualize.

To be able to create the array, the collection called Family in the database (figure 3.1b) is used. This is because the necessary hierarchical relationships are stored in this collection. After created, the array was restructured in order to represent the correct hierarchy. In figure 3.2 a sample array can be seen to better understand the structure of the hierarchical array.

```

var individuals = [
  {id: central person, children: [
    {id: mother, children: [
      {id: grandmother, children: [
        {id: great-grandmother, children: [...]},
        {id: great-grandfather, children: [...]}
      ]},
      {id: grandfather, children: [
        {id: great-grandmother, children: [...]},
        {id: great-grandfather, children: [...]}
      ]}
    ]},
    {id: father, children: [
      {id: grandmother, children: [
        {id: great-grandmother, children: [...]},
        {id: great-grandfather, children: [...]}
      ]},
      {id: grandfather, children: [
        {id: great-grandmother, children: [...]},
        {id: great-grandfather, children: [...]}
      ]}
    ]}
  ]}
];

```

FIGURE 3.2: An example of an hierarchical array, written with pseudocode.

3.4 Visualization

Just like Genelines, the application in this work consists of different types of visualizations. A difference though, is that in this application all visualizations are visible at the same time. The visualizations are also linked with each other, allowing interaction between them through brushing and selections. The representations created are an ancestor tree, a sunburst chart, and a bar chart like representation where each individual is presented as a lifeline ordered and aligned according to various temporal events.

The ancestor tree, or more accurately the tree as a data structure in general, has as earlier discussed difficulties in drawing large data sets in order to be neat and understandable. Despite this, an ancestor tree is still used in this work. This is because genealogy data represented as a tree is widely recognized, which will also help the user to understand the other, not as common, visualizations.

To create the visualizations, D3.js is used. D3.js makes use of SVG (Scalar Vector Graphics), a format in which vector-based graphics are defined. With SVG, different kind of objects can be created, like lines, graphs and figures of different complexity. An advantage with SVG-objects is that regardless of resolution and size, they maintain the same high quality. It is thus possible to zoom in on an object without any quality loss. [27] SVG-objects can both be styled, which is done using CSS, and transformed. They also support animation. Different event handlers can be applied to an SVG-object. [28] Examples of event handlers used in this application are "onmouseover" and "onclick" handling what will happen when the user hovers over an object and when the user clicks on an object.

3.4.1 Ancestor Tree

The ancestor tree is a binary tree with the root placed in the bottom. One of the aims with the ancestor tree is that the user should get a clear overview of the whole tree, thus all ancestors should be visible at the same time. The user should not have to pan

in order to see all ancestors. For all ancestors to fit in the same view the tree can not be too big, and some rules for the node positioning need to be adopted. Wetherell and Shannon [1] present three aesthetic rules and one physical limit that should be adopted when drawing a tidy tree, see section 2.3. They implemented two algorithms following these rules differently. The resulting trees from their algorithms can be seen in figure 2.6. The first tree fulfills the aesthetic rules, but not the physical limit since it has a larger width than the smallest possible. The second tree, on the other hand, has the smallest width possible, but does not fulfill the third aesthetic rule. In this work the assumption that the first tree is a better choice when visualizing genealogy data is taken. This is because it is of importance that the hierarchical relationships are clear, and that the user has no problem of understanding them. A tree that is more aesthetically appealing will ease this. With the additional aesthetic rule, presented by Reingold and Tilford, the tree will be more aesthetically appealing, which will ease the understanding of the tree even further.

The aesthetic rules described by Wetherell and Shannon will be applied in this work, but with a modification to the second rule. The second rule says that every left child should always be positioned to the left of its parent, and a right child to the right of its parent. According to the modified rule this will only be applied in those cases when the parent node has two children. If the parent node has only one child, they will instead be placed straight above each other. This is because it will result in a tree with a width smaller than otherwise. This is clarified in figure 3.3. The fourth aesthetic rule will also be applied in this work. The aesthetic rules adopted when drawing the ancestor tree are thus:

- Nodes of a tree at the same height should lie along a straight line, and the straight lines defining the levels should be parallel.
- In a binary tree, each left child should be positioned left of its parent and each right child right of its parent, except in those cases where the parent has only one child. The child should then be positioned straight above its parent.
- A parent should be centered beneath its children.
- A tree and its mirror image should produce drawings that are reflections of one another

The physical limit saying that the tree should be drawn with the smallest width possible will also be applied, as long as it does not break any of the aesthetic rules.



(a) A sample tree drawn according to the rules defined by Wetherell and Shannon.

(b) A narrower version of the same sample tree, due to the modified rule.

FIGURE 3.3: Two versions of the same sample tree.

In figure 3.4 the drawing process of the nodes is described. They are drawn according to the order defined in figure 3.4a. The algorithm deciding which node will

be drawn next is recursive, and is always investigating if the node just drawn has any child node. If it does, the child node will be drawn next, instead of the previous drawn node's eventual remaining child. This is recursively repeated which will result in the nodes being drawn in the described order.

Each time a new node is drawn, the position of each node will be examined, and updated when needed. This is to ensure that the aesthetic rules are always fulfilled. Their vertical position is fixed and depends on the level of the node. The horizontal position is the one that may change during the drawing procedure. Figure 3.4b is showing an example of how the nodes change position each time a new node is added.

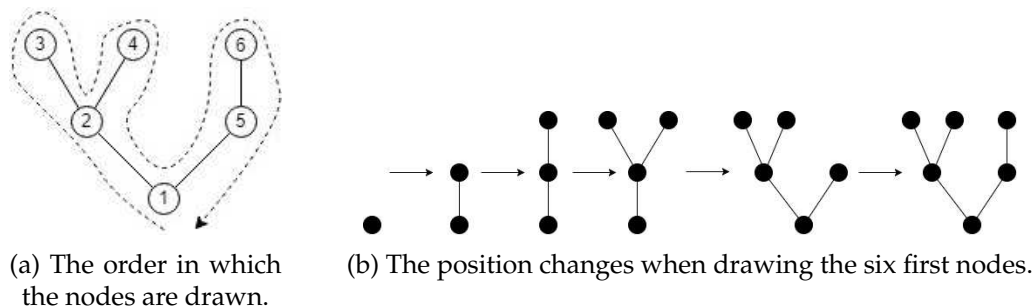
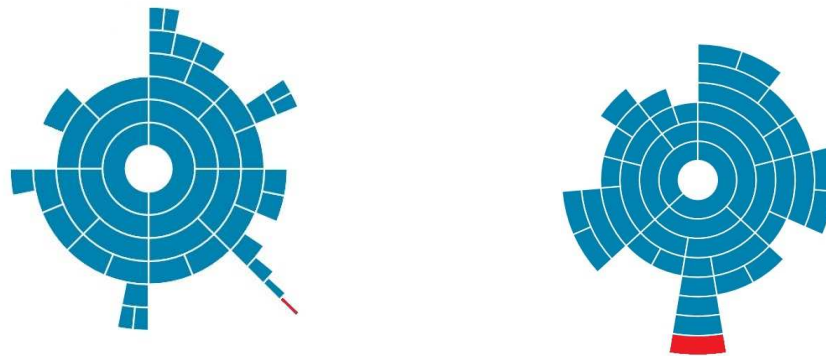


FIGURE 3.4: The drawing process of the nodes.

3.4.2 Sunburst

The fan chart drawn by Genelines, figure 2.13, is drawn along a horizontal timeline. An advantage of that is that it is easy to compare the difference in birth years between people of the same generation. On the contrary, the fan chart could be a bit misleading as it is easy to interpret the width of a node as the total age of the person represented by that node. This is not a correct interpretation, since the width of the node depends on the birth year of its child node, and not by its death date. The width will therefore not represent the entire life span of the person. With this in mind, a sunburst diagram has been chosen for this application which is not drawn along a timeline. The placement of the nodes will only represent the hierarchical relations. Nodes belonging to the same generation will be drawn along the same concentric circle. The order in which the nodes are drawn is the same as described in section 3.4.1.

Nodes along the same concentric circle will not necessarily have the same size. The size of a node is dependent on the total amount of nodes creating the branch that the concerned node is a part of. When comparing two nodes at the same level, where one of them belongs to a branch containing for example five nodes and the other belongs to a branch containing fifteen nodes, the last one will be the larger one. To size the nodes like this is advantageous in those cases where one or more of the branches are very long. In each branch, the size of the nodes decreases with every generation. Imagining a very long branch, the node at the end will become very small, and in worst cases hard for the user to see, clarified in figure 3.5. By determining the size of a node based on the size of the branch, those cases will be minimized. Another reason for using this technique, is that it will be more apparent where in the data set there is more data collected. The user will then get an indication of in which part of the family he or she should collect more data.



(a) A sunburst chart where all nodes at the same concentric circle have the same size.

(b) A sunburst chart where the size of a node depends on the number of nodes creating the branch.

FIGURE 3.5: A comparison of the two sizing techniques. As can be seen the red node in the left image is much smaller than the same node colored in red in the right image.

3.4.3 Lifeline Representation

A bar chart like representation is used to visualize the individuals as lifelines. Nam Wook Kim et al. [2], Daniyar Mukaliyev [4], and Genelines [3] all use lifelines in their works to visualize genealogy data. Unlike those representations though, no hierarchical relationships are presented in this lifeline representation. The lifelines are sorted vertically according to a user selected variable, such as birth year, sex etc. A vertical axis is included in the representation which represents the time of an arbitrary event (such as getting married, or having your first child), selected by the user from a drop-down menu, see figure 4.8. All lifelines are then horizontally aligned along this axis with respect to this chosen event. As the lifeline is representing the life span of a person, each year of that person's life corresponds to a certain point of the lifeline. If the vertical axis is set to represent the time of getting married, the lifeline will be horizontally positioned in order to intersect with the axis at the point corresponding to the age of which the represented person got married. Each lifeline is positioned as described, hence, it is possible to compare the age of the visualized persons at different life happenings.

The lifeline representation is designed like this because of the assumption that additional insights and correlations can be found by comparing the lives of the ancestors, instead of presenting their hierarchical relationships. Also, the hierarchical relationships are already visualized in the ancestor tree and the sunburst chart.

Chapter 4

Results

This work has resulted in the implementation of a web application for visualizing different aspects of genealogy data stored in a GEDCom file. In figure 4.1, the final result is presented, consisting of three different representations: an ancestor tree, a sunburst chart and a lifeline representation, all interacting with each other.

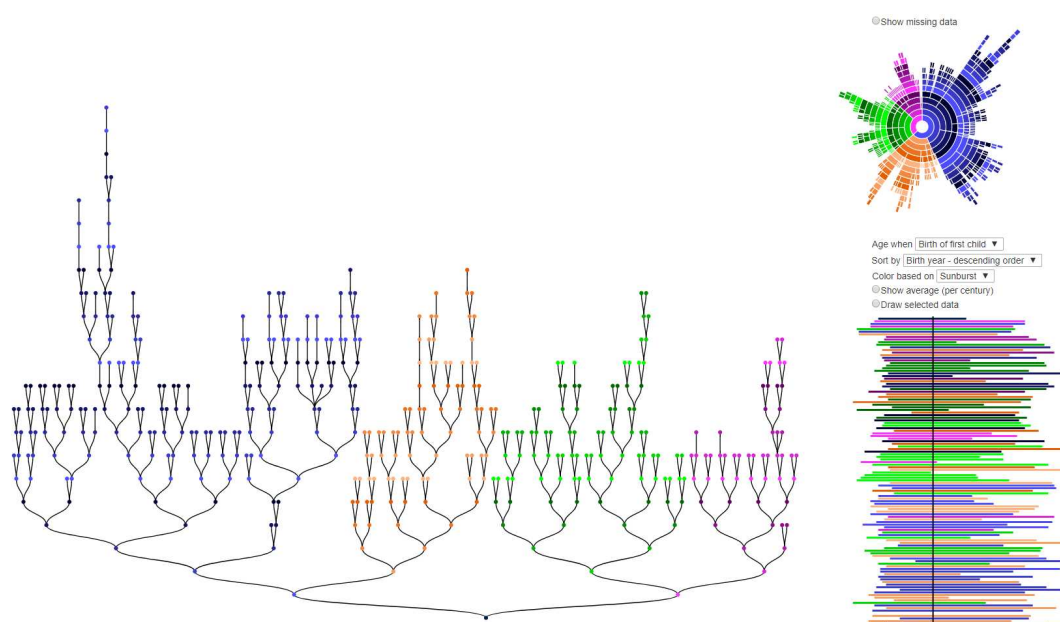


FIGURE 4.1: The final result of the application.

4.1 Ancestor Tree

The ancestor tree can be seen in figure 4.2. Each person is drawn as a small dot, linked together with curved lines representing the hierarchical relationships. At each branching, the male ancestor is placed to the left of the parent node, and the female ancestor is placed to the right. In those cases where the parent node has only one child node, the child node will be placed straight above its parent node according to the second aesthetic rule (section 3.4.1), thus, the user cannot tell if the ancestor is male or female. To make sure that the user can always determine if a person is male or female, information about a person's gender is presented as a symbol in a pop-up window that appears when hovering over that person. The pop-up window also contains information about the person's name and birth year, as well as a number written in parenthesis describing the number of generations between the hovered person and

the central person, figure 4.3 (since the data in the application is real, no names are presented in the figures of this report. Instead the individual identification numbers are used in the figures). When a person is hovered, the path between that person and the central person is highlighted, while the rest of the tree becomes transparent in order for the user to be able to focus on the hovered person. Multiple colors are used to draw the nodes. The colors are further discussed in section 4.4.

By double clicking on one of the nodes in the tree, the corresponding ancestor will become the new central person. The ancestor tree, as well as the sunburst chart and the lifeline representation will then be redrawn, based on the new central person.

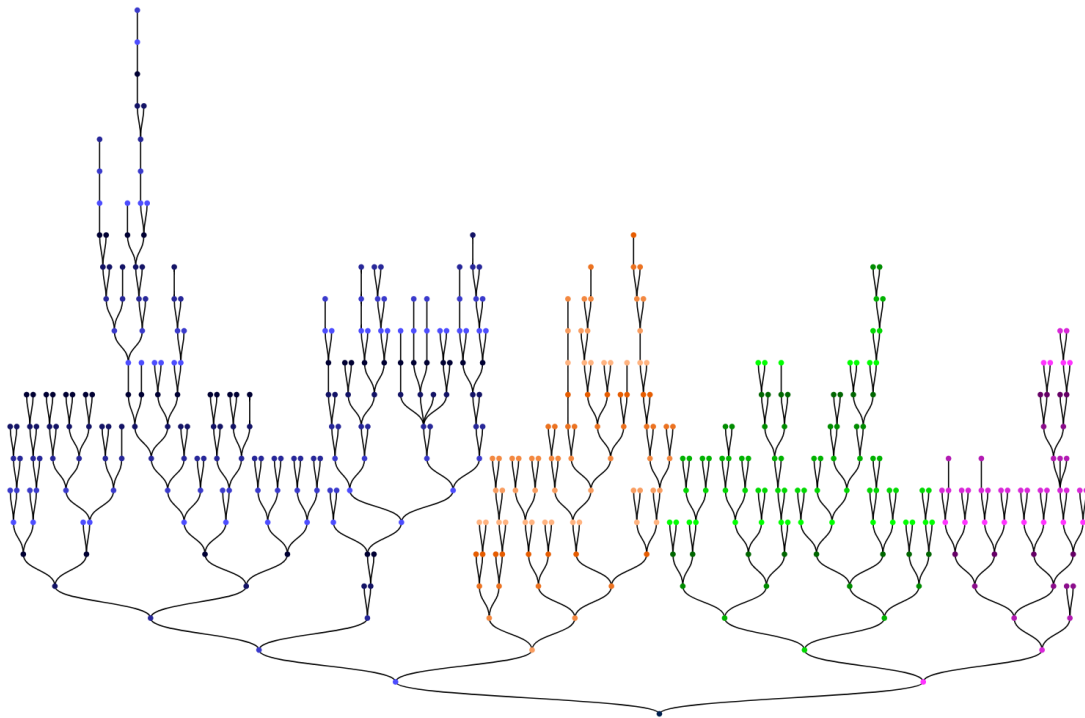


FIGURE 4.2: An ancestor tree created with the application.

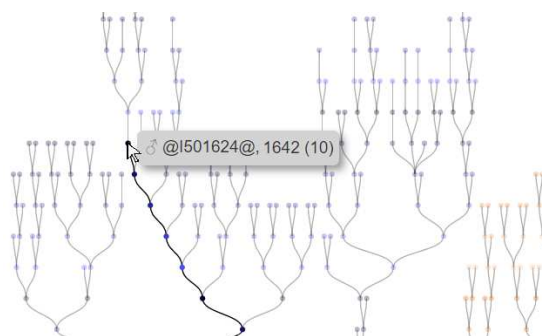


FIGURE 4.3: A pop-up window appears when hovering a person in the tree.

4.2 Sunburst

The sunburst chart, similar to the ancestor tree, presents all the ancestors and their hierarchical relationships, but in a radial layout, figure 4.4. Each person is drawn

as an arc, a part of a circle. The persons being a part of the same circle, belong to the same generation. The root, which represents the central person, is placed in the middle, and for every generation a new circle is built around the root. Just like in the tree representation, the male ancestors are placed to the left and the female ancestor to the right of each other. The colors used in the sunburst are the same used in the ancestor tree (described in section 4.4).

Like with the ancestor tree, the user can get additional information about a person by hovering it. An identical pop-up window will then appear containing information about the person's gender, name and birth year, as well as the number of generations to the central person, figure 4.5. When the user hovers a person, the path of ancestors connecting that person to the central person will be highlighted. This is done by making the rest of the sunburst transparent.

The user can also choose to click on a specific person. The path of ancestor will then remain highlighted. The user can highlight one or more paths like this, figure 4.6. By selecting the checkbox "Draw selected data" in the lifeline representation (section 4.3), the highlighted ancestors will be the only ancestors drawn as lifelines. This will be further clarified in the next section.

In the sunburst chart it is also possible to present which ancestors, according to the current settings in the lifeline representation, are missing data that are needed for them to be drawn as lifelines. This is done by the user clicking on the radio button "Show missing data". The persons missing data will then be drawn in red instead of their default color, figure 4.7.

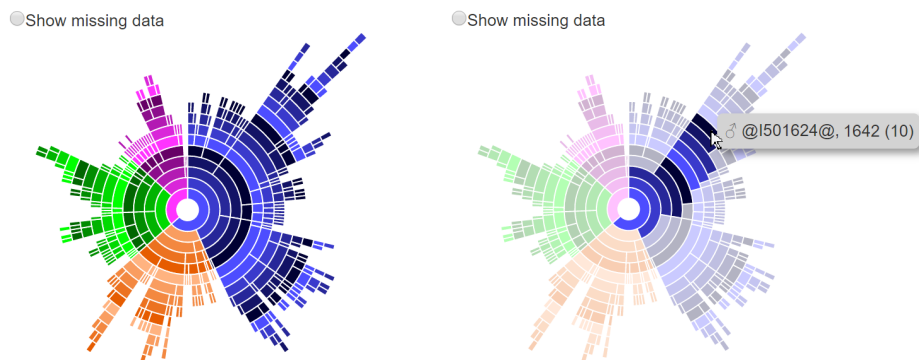


FIGURE 4.4: A sunburst chart created with the application.

FIGURE 4.5: The sunburst chart when hovering a person.

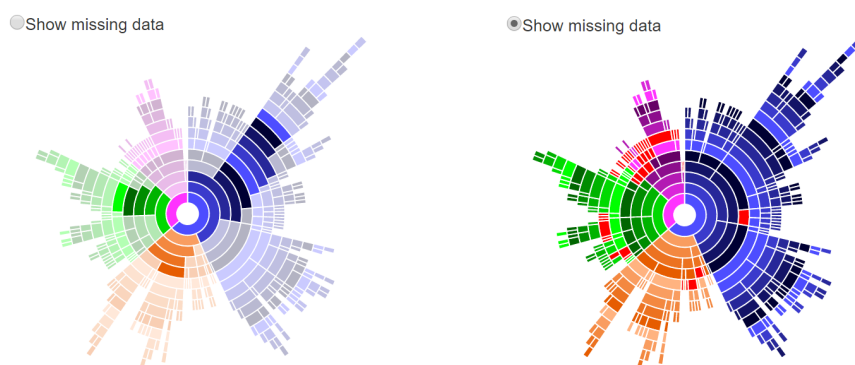


FIGURE 4.6: Multiple paths of ancestors can be highlighted.

FIGURE 4.7: If a node is missing data, it will be drawn in red.

4.3 Lifelines

In figure 4.8, the lifeline representation can be seen, where each lifeline represents a person. In the example shown in the figure, the vertical axis is set to represent the time when the visualized ancestors got their first child. The user can change what event the vertical axis is presenting, by choosing another option in the drop-down menu. Currently, the options available are “Marriage” and “Birth of first child”. By alternating these options, the lines will be realigned around the vertical axis with respect to the chosen event.

The user can also define how the lines will be sorted vertically. By default, they are sorted by their birth date, with the youngest person placed at the top. It is also possible to sort the lines by their birth dates, but with the oldest person placed at the top. The user can also choose to sort the lines based on gender. This is in order to more clearly present possible differences between men and women when it comes to the age of, for instance, having their first child. When the lines are sorted based on gender, the men are placed at the top, and the women at the bottom.

By default, the lifelines are colored with the same colors as the ancestor tree and the sunburst chart. However, the user can change the coloring in order to present the gender instead, figure 4.9. The lifeline of a male ancestor will then be colored blue, and the line of a female ancestor will be colored red. This as well clarifies the possible differences between men and women.

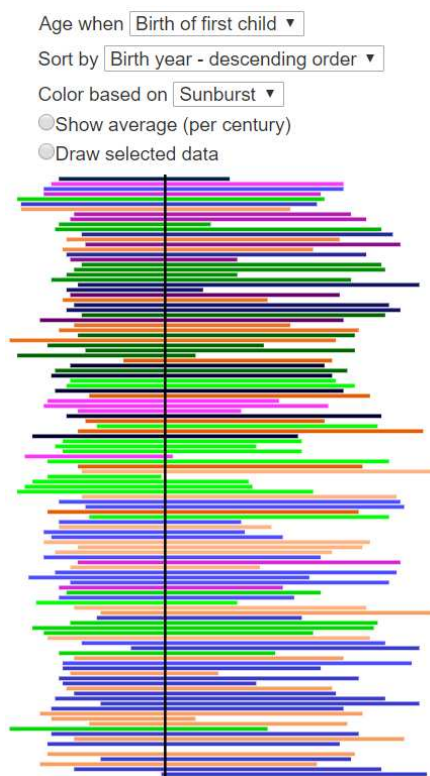


FIGURE 4.8: A lifeline representation created with the application.

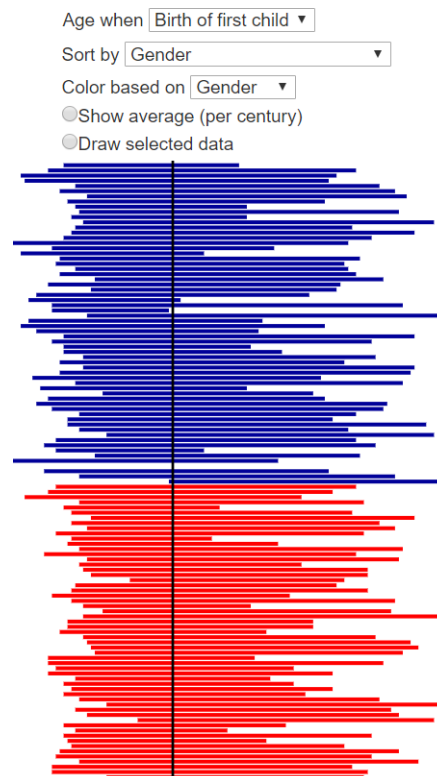


FIGURE 4.9: The lifelines sorted and colored based on their sex.

For more information about who a particular lifeline is representing, the user can hover over that line, figure 4.10. The rest of the lines will then be transparent, while the hovered line retains its color. There will also appear a pop-up window, as with

the ancestor tree and the sunburst chart, presenting the name and birth year of the person, as well as the age of the person at the time of the represented event.



FIGURE 4.10: The lifeline representation when a line is being hovered.

In those cases when the data set visualized in the application is too big, all ancestors will not fit at the same time in the lifeline representation, since the drawing surface will be too small. To slightly reduce the data set, the persons who do not have all the information needed stored in the database (e.g. birth or death date, or a date for the event represented by the vertical axis) will not be drawn. Moreover, functionality for scrolling in the lifeline representation has been implemented. Thus, even though all lifelines can not be seen at the same time, the user has the ability to scroll through the lines in order to see all of them.

In order to allow the user to have a greater impact on which ancestors that are visualized in the lifeline representation, functionality for selecting ancestors in the sunburst has been implemented. This is done by, as described in section 4.2, highlighting one or more paths of ancestors in the sunburst chart. The lines will then be drawn in different colors than the default colors. The paths of chosen ancestors will be alternating between two different shades of blue, figure 4.11. This is to distinguish which lines belongs to the same path in order to compare them against each other. The sunburst chart will still be colored with the default colors.

It is also possible to set the lifeline representation to present the average age of a particular event. Each line in the visualization will then represent a specific century, where the length of each line corresponds to the average life length of that specific century. The average age of the particular event is then calculated for each century, figure 4.12. Here as well, the lines are alternating between two different shades of blue in order for the user to be able to distinguish them from each other. By hovering a line, a pop-up window will appear, containing information about the average age for that specific century, as well as information about which century the line is representing.

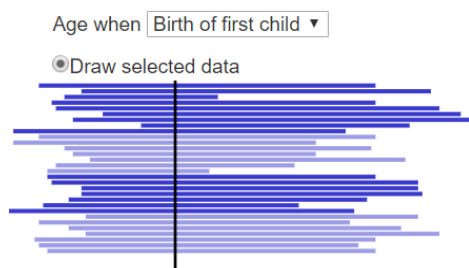


FIGURE 4.11: The paths of ancestors highlighted in the sunburst chart, drawn in different colors.

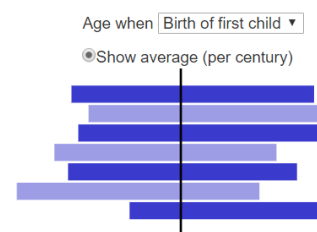


FIGURE 4.12: The lines represents the average age of each century.

4.4 Colors and Interaction

The colors used as default in order to draw all ancestors in the different representations can be seen in figure 4.13. Four base color are used, all alternating between five different shades. As can be seen in figure 4.2 and figure 4.4, the four different colors clearly divide the graphs in four branches, one for each grandparent. For each color, the different shades alternate between the generations. All persons belonging to the same generations, will thus be drawn with the same shade.

In those cases where the lines are drawn in other colors than the default, the colors presented in figure 4.14 are used. The colors are selected in order to be easy to distinguish from each other. This is to make sure that different groupings, like the one in figure 4.11, will be clearly understood.

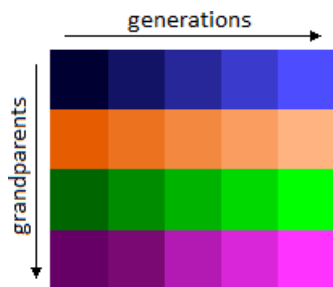


FIGURE 4.13: The default colors used in the application.



FIGURE 4.14: The colors used in special cases.

In figure 4.15 an example is shown where the different representations are interacting with each other. When a person is hovered by the user in any of the representations, that person will as earlier described be highlighted. This will not only happen in the current representation though, the person hovered will be highlighted in the rest of the representations as well.

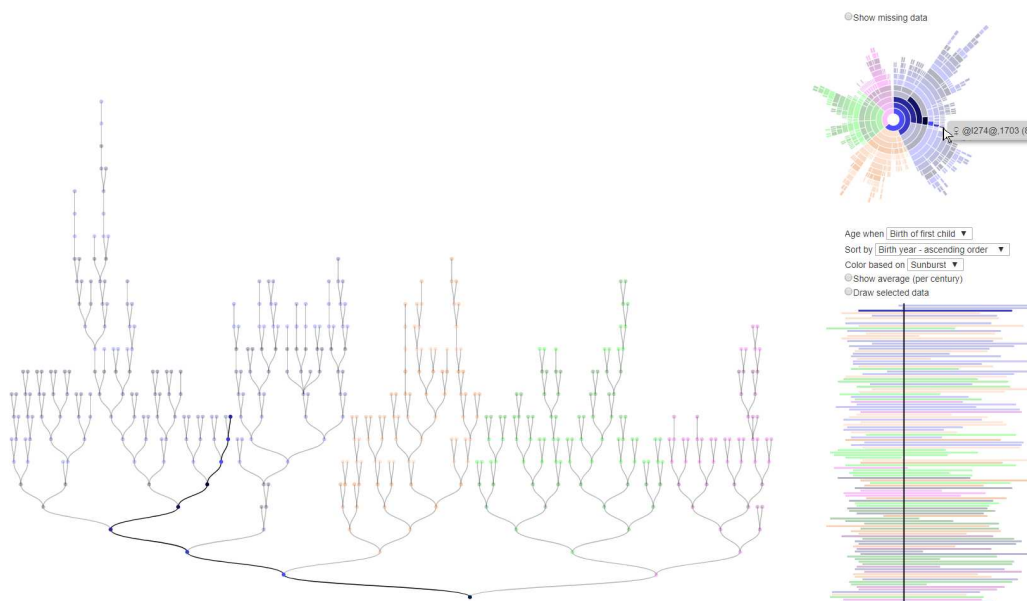


FIGURE 4.15: The three visualizations of the application interacting with each other.

4.5 Evaluation Meeting

After implementing all three representations, there was a meeting held with the two earlier mentioned (section 1.2) genealogists. The purpose with the meeting was to evaluate the different representations. The feedback from the genealogists led to some changes concerning the result.

In figure 4.16, the first version of the ancestor tree can be seen. It did not contain any colors, but the persons were drawn as black dots, and the information appearing when hovering a person did just contain the name and birth year of that person. The changes discussed were that the nodes in the ancestor tree should be colored with the same colors as the sunburst chart and the lifeline representation since that will create a better connection between all representations. It was also suggested that when hovering a person, there should be a number presenting the number of generations between the hovered person and the central person. This is because in a large tree, it can be difficult to keep track of the hierarchical relationships between people.

The looks of the sunburst chart were also changed due to the evaluation meeting. The different stages can be seen in figure 4.17. At first, there was only one color used, gradually changing to a darker shade for every generation, figure 4.17a. During the meeting it was discussed that the color should instead alternate between five shades, repeated for every fifth generation. This resulted in the sunburst chart presented in figure 4.17b. It was also discussed that instead of using only one color, four different colors should be used, one for each grandparent, which would lead to a clearer distinction of the different parts of the family. This resulted in the final sunburst chart, figure 4.17c.

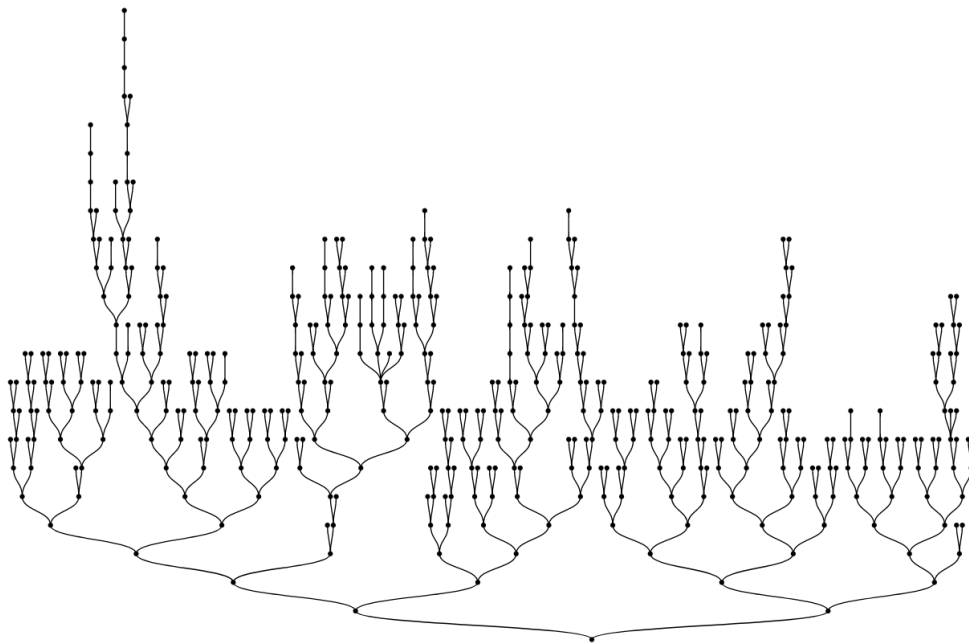


FIGURE 4.16: The first version of the ancestor tree.

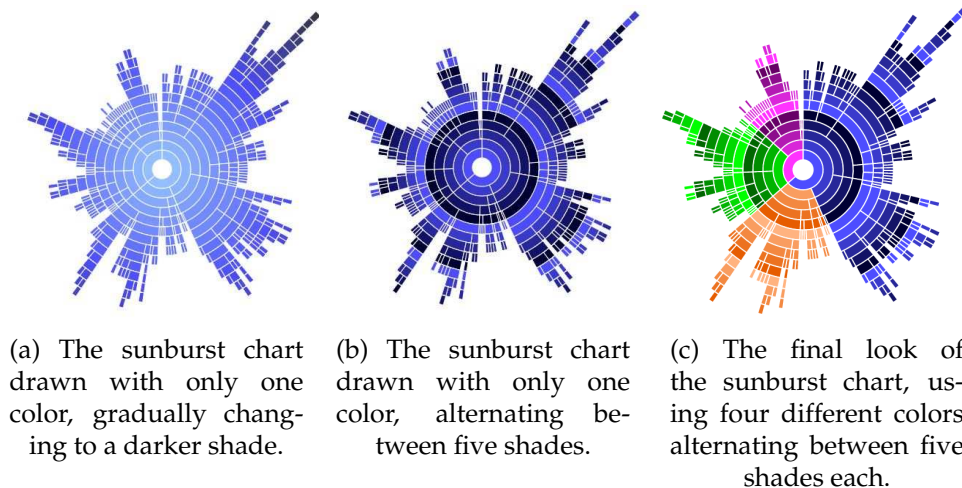


FIGURE 4.17: The different looks of the sunburst chart.

The appearance of the lifelines has not changed due to the meeting, other than the color changes described above. However, there was a suggestion that it should be visible which lifelines do not have all the temporal data necessary stored in the database. Instead of making that visible in the lifeline representation, this suggestion has been applied to the sunburst chart, where as presented in figure 4.7, it is possible for the user to see the persons with missing data drawn in red. The reason for presenting this in the sunburst chart instead of the lifeline representations is because in the sunburst chart, the nodes do not need any temporal data in order to be drawn in their correct positions. In the lifeline representation however, the position of each line is calculated based on its temporal data. If there is temporal data missing, the correct position of the lifeline cannot be calculated, and the lifeline will not be drawn. It will therefore be more clear to present which nodes are missing data in the sunburst chart.

There was also a request that the user should be able to select a group of people that would be drawn as lifelines instead of always showing all the individuals. This was then implemented in the sunburst chart as well, as mentioned in section 4.2, making it possible for the user to highlight paths of ancestors, and then visualize them as lifelines.

Chapter 5

Discussion

In this chapter, the different decisions taken and the methods used in this work are discussed. The final result as well as suggestions on what may be improved will also be discussed.

5.1 Ancestor Tree

The aim with the ancestor tree was to offer the user a representation of the genealogy data that would be easy to recognize and understand. As earlier mentioned, the classical layout of a tree (section 2.3.1) is the most used method for visualizing genealogy data. This makes that layout a suitable choice for creating a recognizable visualization. Compared to the H-tree for example, it will be much easier for the user to understand the connection between two ancestors in the tree with the chosen layout. The path of ancestors connecting the two people will include several direction changes in the H-tree, and even by highlighting the path, it would still be easy to get confused along the path.

From the evaluation meeting with the two genealogists it appeared that the ancestor tree visualizes the data in a clear and comprehensive manner. The tree differs from other family tree representations in such a way that all ancestors are visualized at the same time. Hence, the user does not have to drag the tree around in order to see all ancestors. This will contribute to the tree being easy to understand.

Another advantage of the resulted ancestor tree according to the genealogists is that it will work as a "fingerprint" of the visualized family. Each ancestor tree will have its own characteristic shape, and each time using the application, the user will immediately recognize the tree. By using this application regularly, the user will learn where in the tree a certain ancestor is placed, which will be helpful in order to not get lost in the tree. The "fingerprint" idea will also make the genealogy experience more fun. The user will get a visual "portrait" of his family, to both present to his family and to compare with the portraits of other genealogists. In figure 5.1 two different trees are drawn, both with their own characteristic "fingerprint".

As mentioned in section 3.4.1, four aesthetic rules were applied to the drawing process of the tree. These rules contribute in different ways to a good structure of the tree, where the hierarchical relationships are presented in a clear way, and from which the user can draw different kind of conclusions. For instance, the first rule saying that all nodes belonging to the same level should be placed along the same horizontal line will contribute with not only a better structure of the tree but will also work as an indicator of where in the data set the user should collect more data, which one of the main research questions of this work has been about (section 1.3). As can be seen in figure 4.2, some of the blue branches are much higher than for example the purple branches. This means that the blue branch contains more generations of ancestors,

hence, more data is collected in that part of the family. Based on this, the user can take the decision to prioritize collecting data in other parts of the family.

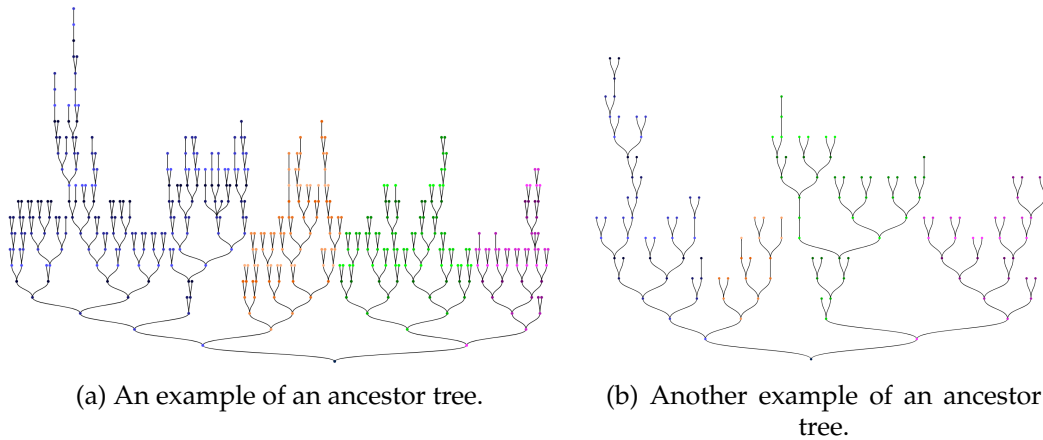


FIGURE 5.1: Two ancestor trees with their own characteristic shapes.

The ancestor tree can also be used to detect incorrect data in the database. One example is shown in figure 5.2. The tree is showing that one of the ancestors has four parents, which cannot be correct. Because of the tree presenting all the ancestors at the same time, this kind of incorrect data is easy to detect. In a tree where the user would have to drag the tree around in order to view all ancestors, it would be easier to miss that mistake.

The tree shown in figure 4.2 consists of 439 ancestors, where the greatest ancestor was born in 1350. The fact that they all fit in the drawing surface at once is a great result. Still, suppose that the number of ancestors to be drawn was double the amount of the ancestors in the figure. In this case they would probably not fit at the same time, and functionality to handle these situations should be implemented. The degree-of-interest technique presented by Nam Wook Kim et al. [2], where each person is assigned a degree of interest in order for the application to prioritize which persons to draw, could be implemented in this application in order to solve this problem. The DOI functionality could then be extended to be able to be affected by the user. When calculating the DOI for each person, Nam Wook Kim et al. have implemented that the central gets the highest DOI, and for each person in the data set the value decreases in relation to their distance to the central person. For people related to the central person, the DOI decreases linearly, while for marital relationship, it decreases more slowly. For the user to be able to affect the DOI functionality, it could instead be implemented that the user can decide how the DOI should decrease for each ancestor. If the user is interested in analyzing only the paternal ancestors, he or she can set the application to prioritize them. The DOI would then be decreasing more slowly for the paternal ancestors, than for the maternal ancestor. If the user instead is interested in viewing as many generations as possible, the DOI would be calculated to fit that request instead. The user would then have a great impact on what data that is being visualized.

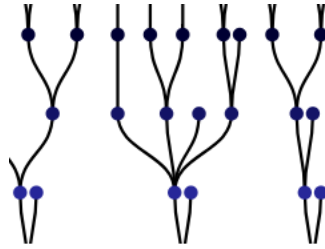


FIGURE 5.2: One of the ancestor has four parents according to the tree, due to an error in the database.

5.2 Sunburst

Compared to the ancestor tree, the sunburst visualizes the same data more compact – it does not need the same amount of space in order to draw the same number of ancestors. Because of this, the sunburst works as a great complement to the ancestor tree. In cases when all ancestors cannot be visualized at once, as described in section 5.1, they can still be included in the sunburst chart since it uses less space.

As earlier mentioned, due to time constraints, no descendants of the central person are visualized in the application. If the descendants were visualized as well, there would be even more data excluded from the tree representation due to the lack of space. In that case, two sunburst charts could be drawn, one for the ancestors and one for the descendants of the central person. The sunburst representation would then have an even more important role as a complement to the tree.

Another great use of the sunburst chart is to present which of the ancestors that do not have all data needed stored in the database, in order to be drawn as a lifeline. By coloring them in red, the user will get a clear indication of which ancestor needs more data to be added to the GEDCom file.

The sizing method for deciding the size of each node in the sunburst chart was discussed during the evaluation meeting. With the method used in the application, it may occur that a node that is placed further away from the root compared to another node will be drawn larger in size. One of the genealogists mentioned that this could be misleading since the user could get the impression that the node further away would be of bigger importance than the other node. There was a suggestion that all nodes in the same generation should instead be equal in size. The advantages and disadvantages of the suggested method, and the method used in the application were discussed. After trying both methods, it was decided that the method used in the final result is more suitable in this application, based on the motivation in section 3.4.2.

5.3 Lifeline Representation

Nam Wook Kim et al. [2], Daniyar Mukaliyev [4], and Genelines [3] all use lifelines in their works to present genealogy data. In those representations, the hierarchical relationships between the people are the main focus, compared to the lifeline representation in this work where the comparison between the lines is the main focus. For instance, comparing two people with four generations of difference with the TimeNets representation for example, will be difficult since their lines will be placed far apart due to their birth dates. In the presented lifeline representation, the comparison between people will be easy regardless which generations they belong to. During the

evaluation meeting, the genealogists gave the feedback that the lifeline representation presents a new approach of visualizing genealogy data. Compared to the most usual genealogy visualizations, new type of insights can be gained with this representation. The lifeline representation is therefore considered having an important role in the application, as the aim of this work is partly to create a visualization where the user will get new kind of insight from their collected data.

Thanks to the different sorting settings, different correlations can be found. For instance, by vertically sorting the lifelines based on the persons' birth year, and horizontally aligning them based on their age of marriage, potential patterns can be seen telling the user for example how the age when getting married has changed throughout the generations. Potential differences between male and female ancestors regarding their age when getting married or having their first child, can be clarified by sorting the lines based on sex instead.

An example of another correlation that was found during this work due to the lifeline representation is that a big part of the family visualized at the time, got their first child the year after they got married. This was found just by alternating which event the axis represented. To be able to easily find correlations like this will make the genealogy process more fun and feel more vivid. However, the only events the user can currently chose to visualize is "Marriage" and "Birth of first child". In order to enable that more and different types of insights and correlations can be found with the lifeline representation, more type of events stored in the GEDCom file should be added to the application.

The lifeline representation also works as a tool to find errors in the database. In figure 5.3, one example can be seen. According to the visualization, the hovered person was just one year old when getting her first child. This will tell the user to adjust the data stored in the GEDCom file.

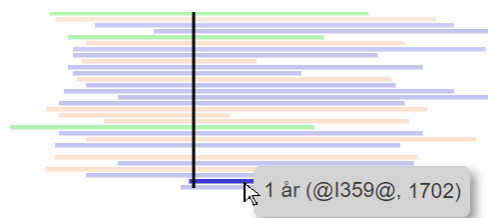


FIGURE 5.3: An error in the database found by the lifeline representation.

In the final result, as described in section 4.2, by highlighting one or more paths of ancestors, the user can select which persons are drawn as lifelines. A suggestion of an improvement in order to make the application more flexible, is to allow the user to select separate individuals as well, and not only a complete path of ancestors. In some case the user may need to compare two persons, placed in completely different parts of the sunburst. A lifeline should then not be drawn for each person belonging to the paths connecting the selected persons to the central person, since that will just complicate the comparison between the two specific persons. One improvement of the application would therefore be to allow the user to select separate individuals as well.

5.4 Colors and Interaction

In order to create a unified application, the three different visualizations are linked with each other. For example, when a person is hovered over in one of the visualizations, the same person will be highlighted in the other visualizations as well. This will also help the user to understand the different visualizations. For instance, if the user understands exactly how the ancestor tree is structured and how the hierarchical relationships are presented, but not fully understands the sunburst chart, by hovering a person in the sunburst chart, and seeing the placement of that person in the ancestor tree, the user will more easily understand the sunburst chart.

When hovering a person there will also appear an information box presenting some personal information. By presenting personal information like this, instead of writing it directly in the nodes, the nodes can be drawn much smaller. In the ancestor tree for instance, each person is drawn as a really small dot, which results in that more persons can fit in the drawing surface. Using the hovering technique in order to present personal information, will therefore help the visualization to become tidier and to present more data in the same surface.

Four different base colors were used in the visualizations in order to create four distinct branches. This is because the user will easier understand which part of the family a certain ancestor belongs to. By using the four colors like this, they will also work as an indicator telling the user in which parts of the family he or she needs to collect more data. In both figure 4.2 and figure 4.4 it is clear that the blue part of each visualization consists of more ancestors than for instance the purple part.

5.5 Implementation

During this work, several difficulties were encountered regarding parsing the data from the GEDCom file to MongoDB. Code that worked for some GEDCom files did not work for others, and when trying to load large files the application crashed. As mentioned in section 3.1, this problem was solved by dividing the GEDCom files into smaller parts when trying to read them, and process the data from one part at the time. However, much time was spent before understanding that the reason why the application did crash was the size of the GEDCom files. Therefore, a lot of unnecessary time was spent trying to solve the same problem. There are a lot of already implemented GEDCom-parsers that could have been used in this application in order to save time, but the decision was made to not use any of them. This is due to the fact that by implementing your own parser, you have full control over how the parsed data will be structured, and which data will be saved. A GEDCom file contains a lot of information not used in this application. By using an existing GEDCom-parser a lot of data would be unnecessary stored in the database, and the structure of the data would not be adjusted for this application, which would make the rest of the implementation more difficult. Despite the knowledge about the difficulties experienced, the same decision regarding writing your own parser or using an existing one would have been made today. The ability to customize the structure of the data based on the specific conditions of the application, is assumed more advantageous than the time saved by using an already existing parser. Also, the reason for the amount of time spent on the problem is the lack of experience regarding solving this kind of tasks. Someone more experienced may not have encountered the same difficulties, and therefore it is considered better to not use an already existing parser in this specific situation.

Other difficulties encountered during the implementation is about asynchronous programming. Because of lack of previous experience regarding asynchronous programming, the flow in which asynchronous code is executed was not fully understood. This affected the implementation time as well, since a lot of time was spent trying to understand how the code should be structured. With the help of the utility module `async`, it later became easier to understand and implement the asynchronous functions as well.

D3.js as a tool for visualizing the different representations in the application has worked very well. There have not been any difficulties when implementing the different types of visualizations, and with D3.js it has also been easy to handle what will happen when the user hovers over an object or clicks on it. The use of SVG (scalar vector graphics) leads to that the objects can be drawn in any resolution or size, without any quality loss, which is important in order to be able to control the looks of the graphs in a flexible manner. D3.js is therefore a powerful and adaptable tool for creating this kind of applications.

Chapter 6

Conclusion & Future work

With the aim to examine how a multiperspective visualization could be used in order to gain new insights from genealogy data stored in a GEDCom file, three research questions were stated, section 1.3. In this chapter those questions will be answered, followed by some conclusions. This chapter will also discuss some future work of this thesis.

6.1 Research Questions

- *When visualizing a large data set, how can a family tree be visualized to simplify the navigation in the tree?*

A tree visualizing a large data set, where the user has to pan the view in order to see all data presented, will be difficult to navigate through. It will be difficult for the user to get a clear overview of the data, which will also complicate the understanding of the connections between the persons in the tree. This work presents a family tree where all visualized persons, for a data set of considerable size, can be seen at once in order to solve this problem. By using a hovering technique it will be even easier for the user not to get lost in the tree when trying to understand the connection between two of the visualized persons. By highlighting the path between the central person and an optional person, their relationship will be clarified. Another method that will help the understanding of the tree is to let the colors alternate between different shades. Thanks to the different shades of the colors, the user will easier keep track of the number of generations between two people in the tree. To summarize, by visualizing the data in a comprehensive manner, where all persons can be seen at the same time, and by using different helping techniques (such as hovering, and using different colors), it is possible to draw a tree easy to navigate through, even when visualizing a large data set.

- *How can genealogy data be visualized from a temporal point of view?*

GEDCom files contain a lot of temporal data, usually not visualized other than with text labels. Depending on the aim with the visualization, temporal data can be visualized differently. Nam Wook Kim et al. [2], Daniyar Mukaliyev [4], and Genelines [3] all present their genealogy data along a timeline. They also have in common that they present a person as a lifeline. Their different approaches of visualizing temporal data work great in visualizations where the hierarchical relationships are in focus. In this work, a different approach of the use of lifelines is presented. As mentioned, the aim with this work is to examine how different visualization techniques could be used in order to gain new kind

of insights. The focus of the lifeline representation is therefore to compare the lives of the visualized ancestors, which, as described in section 3.4.3, will enable that different kind of insights and correlations can be found. In conclusion, genealogy data can be visualized in multiple ways in order to present temporal data, depending on the aim of the visualization.

- *How can genealogy data be visualized to clearly indicate where there is missing data?*

In this work, several techniques have been proposed to clearly indicate where more data needs to be collected, that is, in which part of the family that there is not as much data collected as in other parts. By dividing the family into four parts by using four different colors, the user will clearly see the difference in how many persons each part consists of. The shape of the ancestor tree and the sunburst chart will also give the user an indication of which parts contain more data. Since both the ancestor tree and the sunburst chart visualize all persons at the same time, it is easy to compare the length of the different branches. Also, by sizing the nodes in the sunburst chart as described in section 3.4.2, it will be more clear which parts contains more data. With the presented application it is also possible to see which persons that are missing some personal information in the database. By coloring them in red in the sunburst chart, the user will get an indication that their data needs to be adjusted.

One conclusion that can be drawn from this work is that, by using different kinds of visualizations that can interact with each other, thus creating a multiperspective visualization, it is possible to create an application from which new kinds of insight and correlations can be found. With one or more of the visualizations presenting all visualized data at the same time, the application can also be used as a tool for learning in which part of the family more data need to be collected. It is also concluded that the interaction between the user and the application, as well as the interaction between the different visualizations, is important in order to provide an extra understanding. In this thesis work, the use of colors has also shown to play an important role when creating a visualization easy to understand.

6.2 Future Work

The main change that should be done in future work of this application is to implement support for visualizing descendants, and not limit the application to only visualize ancestors. If all descendants were drawn in the ancestor tree in the same manner as the ancestors, the tree would probably become too large to fit in the drawing surface. A proposal of how they could be drawn instead is to by default only visualize the ancestors, just like with the current representation. Then when the user clicks on one of the ancestors, a descendant tree would be drawn, containing only the descendants of that ancestor. They would be drawn inside a box placed above the ancestor tree. The rest of the tree would then be drawn transparent in order to not draw any attention from the user. To make the descendant tree disappear the user would only have to click on the same ancestor node again. An example of how a descendant tree could look is seen in figure 6.1.

All descendants of the central person could also be visualized as a sunburst chart, just like the current ancestor chart. The central person would thus be placed in the middle of the sunburst, and the descendants placed along concentric circles. The lifeline representation would work pretty much the same as in the current application,

with the only difference that descendants would be drawn as well. More settings could possibly be implemented for the lifeline representation, for instance letting the user sort the lines based on if they present an ancestor or a descendant.

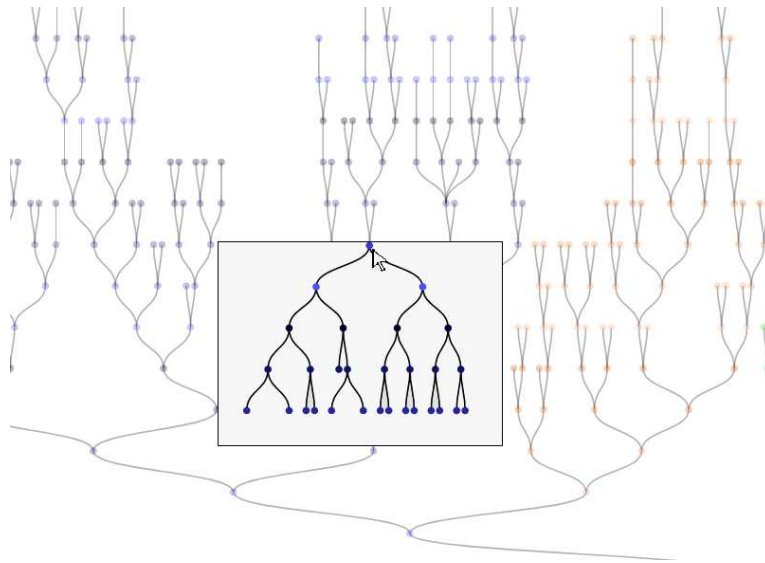


FIGURE 6.1: An example of how a descendants tree could look when hovering one of the ancestors.

Another functionality that could be implemented in future work of this application is to add a timeline along the ancestor tree. The timeline would in this case be placed vertically along the ancestor tree, and the height of a node would no longer depend on which level that node belongs to, but the birth year of the person represented by that node. A simplified example of how this could look can be seen in figure 6.2. By implementing a timeline like this, it would be possible for the user to compare the birth dates of people belonging to the same generation. Just because two persons belongs to the same generation, it does not mean that they were born around the same year. Sometimes the birth year can differ by decades. By being able to see this kind of differences, the user would get more types of insights of his or her family history. To make sure that the user would still have an ancestor tree easy to understand, a toggle functionality could be implemented, offering the user the possibility to switch between the usual ancestor tree, and the ancestor tree drawn along to a timeline.

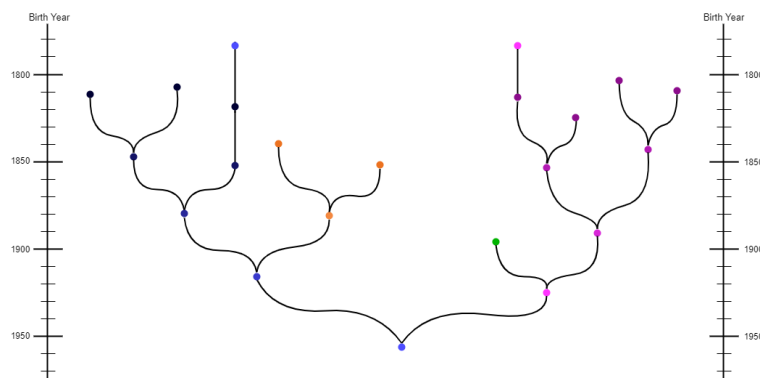


FIGURE 6.2: A sample tree drawn along a timeline.

Bibliography

- [1] Wetherell C, Shannon A. Tidy drawings of trees. *IEEE Transactions on Software Engineering SE-5(5):514-520*, 1979.
- [2] Wook Kim N, Card S K, Heer J. Tracing genealogical data with timenets. In *Proceedings of AVI 2010, Advanced Visual Interfaces*, 2011.
- [3] Progeny Genealogy. Genelines. URL <https://progenygenealogy.com/products/timeline-charts>. Accessed 2017-09-10.
- [4] Mukaliyev D. Visualizing large genealogies with timelines. *Joensuu, University of Eastern Finland*, 2015.
- [5] Dennis Szucs L, Hargreaves Luebking S. *The Source: A Guidebook to American Genealogy, 3rd edition*. Ancestry, 2006.
- [6] Nationalencyklopedin. Genealogi. URL <http://www.ne.se/uppslagsverk/encyklopedi/lång/genealogi>. Accessed 2018-01-10.
- [7] Shown Mills E. Genealogy in the 'information age': History's new frontier? *National Genealogical Society Quarterly* 91, 2003.
- [8] Haley A. *Roots: The Saga of an American Family*. Dell Publishing Company, 1977.
- [9] The Church of Jesus Christ of Latter-day Saints. The gedcom standard release 5.5. 1996.
- [10] Tamassia R. *Handbook of Graph Drawing and Visualization*. Chapman Hall/CRC, 2013.
- [11] Battista G D, Eades P, Tamassia R, Tollis I G. Algorithms for drawing graphs: An annotated bibliography. *Computational Geometry: Theory and Applications* 4(5):235-282, 1994.
- [12] Tuttle C, Nonato L G, Silva C. Pedvis: A structured, space-efficient technique for pedigree visualization. *IEEE Transactions on Visualization and Computer Graphics* 16(6):1063-1072, 2010.
- [13] Progeny Genealogy. Ancestor chart. URL <https://progenygenealogy.com/Products/Family-Tree-Charts/Sample-Charts/Ancestors>. Accessed 2018-01-26.
- [14] Progeny Genealogy. Ancestor fan. URL <http://progenygenealogy.com/Portals/0/images/charts/CCFTM-360-Fan-plain.PDF>. Accessed 2018-01-26.
- [15] Reingold E M, Tilford S J. Tidier drawings of trees. *IEEE Transactions on Software Engineering SE-7(2):223-228*, 1981.
- [16] MongoDB. URL <https://www.mongodb.com/>.

-
- [17] Express. URL <https://expressjs.com/>.
 - [18] AngularJS. URL <https://angularjs.org/>.
 - [19] Node.js. URL <https://nodejs.org/>.
 - [20] D3.js. URL <https://d3js.org/>.
 - [21] Mongoose. URL <http://mongoosejs.com/>.
 - [22] Mongoose. Mongoose documentation. URL <http://mongoosejs.com/docs/guide.html>. Accessed 2018-01-22.
 - [23] Pasquali S. *Mastering Node.js*. Packt Publishing, 2013.
 - [24] Async. Async documentation. URL <https://caolan.github.io/async/docs.html>. Accessed 2018-01-22.
 - [25] Async. Async waterfall. URL <https://caolan.github.io/async/docs.html#waterfall>. Accessed 2018-01-22.
 - [26] Holmes S. *Getting MEAN with Mongo, Express, Angular, and Node*. Manning Publications Co., 2016.
 - [27] W3Schools. SVG Intro. URL https://www.w3schools.com/graphics/svg_intro.asp. Accessed 2018-01-23.
 - [28] W3C. About SVG. URL <https://www.w3.org/Graphics/SVG/About.html>. Accessed 2018-01-23.