FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Controllable Animation for Information Visualisation

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Simon Pointner
Matrikelnummer 01612401

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Ass. Hsiang-Yun Wu, PhD
Mitwirkung: Projektass. Dipl.-Ing. Dr.techn. Haichao Miao, BSc

Wien, 20. Oktober 2020

Simon Pointner	Hsiang-Yun Wu

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Controllable Animation for Information Visualization

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Simon Pointner
Registration Number 01612401

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Ass. Hsiang-Yun Wu, PhD
Assistance: Projektass. Dipl.-Ing. Dr.techn. Haichao Miao, BSc

Vienna, 20th October, 2020

Simon Pointner	Hsiang-Yun Wu

Erklärung zur Verfassung der Arbeit

Simon Pointner

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 20. Oktober 2020

Simon Pointner

Danksagung

Ein großer Dank geht an meine Hauptbetreuerin Hsiang Yun Wu. Sie hat viele der animierten Übergänge inspiriert und mich mit guter Literatur und verwandten Arbeiten versorgt. Auch hat sie mir mit der Organisation meines Arbeitsprozesses geholfen und mich mit vielen Ideen für den Implementierungsteil meiner Arbeit versorgt. Sie war immer kreativ darin, neue Möglichkeiten zu finden um die Implementierung zu verbessern. Weiters habe ich sehr von ihrer Erfahrung in der Erstellung von Nutzerstudien profitiert, womit diese besser geeignet war um die Hypothese der Nutzerstudie zu testen. Aber am dankbarsten bin ich dafür, dass sie sich fast wöchentlich eine Stunde Zeit genommen hat um mit mir über Fortschritte, Probleme und Vorschläge zu diskutieren, wobei sie stets eine Menge Geduld zeigte.

Ich möchte außerdem auch meinen Zweitbetreuer Haichao Miao danken, welcher zu manchen unserer Treffen auch anwesend war und nützliche Tipps und Ideen gab. Im speziellen bei der Lösung mancher technischen Details im Implementierungsteil der Arbeit. Wenn wir gerade über die Implementierung sprechen, möchte ich bei der Gelegenheit auch der Netzgemeinde rund um WebGL danken, welche frei verfügbare Tutorials und Informationen zu WebGL zur Verfügung stellen.

Außerdem Danke an meine Kollegen an der Universität und speziell meinem Bruder, welche mir stets meine Fragen über den Ablauf der Bachelorarbeit beantwortet haben und mich immer wieder motiviert haben.

Acknowledgements

Major thanks go to my supervisor Hsiang Yun Wu. She supported me with a lot by inspirations for animated transition approaches and she provided me with very useful related work. Also, she helped me a lot with organisation of my work process and she gave a lot of input on how to solve problems in the implementation part of the thesis. She was always creative on how to improve the implementation further and what additions might be useful. Further, I highly profit by her experience in designing a good user study which is suitable to test the hypothesis. But I am probably most thankful about the fact that Ms. Wu spent almost an hour each week together with me talking about the progress, problems and suggestions, where she showed great patience with me.

I also want to thank my secondary supervisor Haichao Miao who attended some of our meetings and also gave very useful inputs in brainstorming phases. Especially on how to solve some technical details in the implementation part. Talking about the implementation part, I also want to thank the online community around WebGL, which provides freely accessible tutorials and information on WebGL.

Thanks to my colleagues on the university and especially my brother, who answered my questions about the procedure of doing the bachelor thesis as well as for their motivational support.

Kurzfassung

Die Unterschiede zwischen verschiedenen Visualisierungen zu erkennen und zu identifizieren ist eine kognitive anspruchsvolle Aufgabe. Unterschiedliche Visualisierungen können zu verschiedenen Interpretationen von Daten führen, deshalb ist es wichtig, den Zusammenhang zwischen den einzelnen Visualisierungen zu verstehen und wie diese möglicherweise unterschiedliche Einsichten in die Daten geben können. Ein Ansatz, um diese Korrelation besser zu verstehen und zu vermitteln ist die Verwendung von animierten Übergängen zwischen den verschiedenen Visualisierungen. Dies ermöglicht es dem Betrachter die Veränderungen genau zu verfolgen. Der Fokus dieser Forschung liegt auf animierten Übergängen zwischen klassischen Visualisierungen wie in etwa den Balken-, Kreis-, Ring- und radialen Säulendiagrammen und deren Implementierung als kontrollierbare Animation. Eine kontrollierbare Animation erlaubt dem Nutzer, die Animation über eine Suchleiste wie bei einem Videospieler zu steuern. In dieser Arbeit werden zwei neue animierte Übergänge vorgestellt, eine davon zwischen Balken- und Kreisdiagrammen und die andere für hierarchische Balkendiagramme. Beide Animation nutzen andere Diagramme als Zwischenstufe um einen nachvollziehbaren Übergang zu erreichen. Somit soll die Effektivität und die grafische Wahrnehmung von animierten Übergängen weiter verbessert werden. Eine qualitative Nutzerstudie erbrachte jedoch keine signifikanten verbesserungen abgesehen von einer kleinen Verbesserung in der Effektivität der Animation bei älteren Personen.

Abstract

Understanding and identifying the alternations between different visualisations are cognitively demanding tasks. Distinct visualisations can lead to a different interpretation of data, thus it is important to understand how visualisations correlate with each other and how the insight to the data gained might vary. An approach to achieve the correlation understanding is to introduce animated transitions between different visualisations that allows to precisely follow changes, pursuing the research in the field of animated transitions. In particular, the focus of this research is on animated transitions between commonly used visualisations like bar, doughnut, pie and radial column charts with the addition of implementing them controllable. A controllable animation allows the user to control the animation with a seek-bar like in a video player. This work proposes and implements two new animated transitions, one animation between bar and pie charts and another one for hierarchical bar charts, both utilising other charts as intermediate steps. Expectations are to further improve the effectiveness and graphical perception of animated transitions. Though, a quantitative user study yielded no significant improvements apart from a little effectiveness gain among elder persons.

Contents

CHAPTER 1

Introduction

Since more and more people use digital devices to read the news or do research online, it allows the providers to show them animated graphics instead of static visualisations in printed versions. Animated graphics are a good way for providers to engage more people to read their publications. But this raises the question if those animations are beneficial apart from the encouraging argument. Or might those animations even confuse readers? To answer this question research has already been done for animation in statistical data graphics, or for motion charts. Further, the growth of graphics hardware also allows implementing animation, which is not pre-rendered and allows to implement animation which can be controlled. Those opportunities lead to the question if controllable animated transitions will bring further advantages like a better insight into the underlying data and a better understanding of the visualisations and their relation.

1.1 Overview

Animations are known to spark interest and encourage users to further explore a visualisation [TMB02]. The focus of this thesis is on animated transitions, more precisely about controllable animated transitions. An animated transition is an animation that transforms one visualisation into another smoothly, allowing the user to better understand how the different visualisations represent the same data. Bar charts and pie charts are arguably the most commonly used visualisations and therefore animated transitions for those chart types will be further investigated in this thesis. While bar charts are very effective for value comparison and absolute value estimation, pie charts are an effective tool for conveying information about relative proportions [CM86]. Animated transitions promise to combine the strengths of both chart types. Two new approaches on bar-chart to pie-chart animated transitions will be introduced, of which the arguably better one was implemented. Parts of this implementation were also used to create another animated transition, a hierarchical pie chart. Further, the effectiveness of being able to control

the animation were evaluated by comparing a controllable implementation to a non controllable one in a user study.

The implementation and evaluation of those animations will be described in detail, followed by a discussion about the results and whether or not controllability in animation and the animated transitions might be advantageous.

1.2 Contribution

Extending previous work about animated transitions this thesis contribution can be describes as follows:

- Introduction of two new animated transitions between bar-charts and pie-charts as schematic drawings.
- Implementation of two animated transitions in WebGL.
- Design, implementation and execution of a quantitative user study about controllability in animated transitions
- Evaluation of the user study and drawing of a conclusion.

1.3 Structure of the thesis

The thesis starts off with a chapter about the introduced approaches and terminology which is used in the following chapters. After the description of the proposed animations, chapter 4 explains how the animated transitions were implemented in detail. In chapter 5 the results of the implementation are illustrated, containing images of the animated transition as well as a link to a YouTube video which shows the animations in action. After chapter 6, the evaluation explains, how the user study was designed, which hypothesis was tested and states the results of the evaluation. In the last chapter, the results of the thesis get concluded and further improvements are described. But before all of that, we will take a look at what has already been done in the field of animated transitions.

CHAPTER 2

Related work

While the topic of animation has already been broadly researched, less research was done on animated transitions between different chart types. The main reason for this might be because such animations need to be data-driven, and tools for creating data-driven animations are sparse[TLLS20]. Though, a recent publication[GZL+20] introduced a high-level domain-specific language for creating data-driven chart animations. While the main purpose of this language is to build animations within a chart type, the authors state that animated transitions, like those proposed by [HR07], can be built with their system. This is achieved by an animation the authors call "magic move", which smoothly morphs one SVG visualisation into another one. Though, this morphing effect is very powerful, the variety of multi-staged transitions that can be achieved by it is still limited. As already mentioned, a very relevant contribution to the topic of animated transitions has been done by Heer and Robertson [HR07], where they introduced and implemented several animated transitions between basic visualisations. They also performed a controlled user study to test the performance of their animated transitions. Further Fisher[Fis10] proposed two animated transitions between bar and pie charts. In those approaches, Fisher describes commonly made mistakes in animated transitions and how to embed important animation principles. Consecutively, more animated transitions were proposed and the introduction of the D3 framework[BOH11], which simplifies building interactive visualisations, implementing animated transitions became easier. While [HR07, Fis10] introduce animated transitions between bar and pie charts, none implemented them to be controllable. Further, their proposed animated transitions between bar- and pie-charts have occlusions, which motivated this research. This chapter will be structured into similar animated transitions (2.1) and studies on animation (2.2).

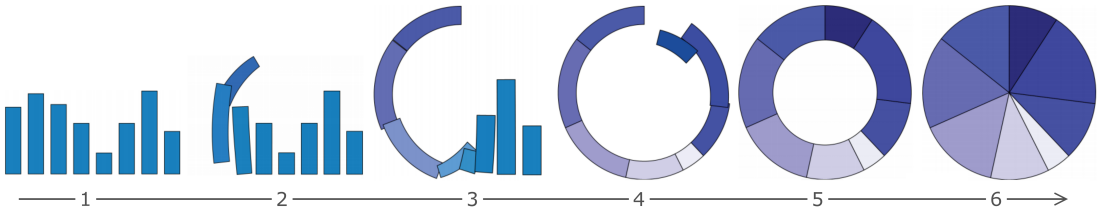

Figure 2.1: Danyel Fisher's approach on bar to pie chart animation [Fis10]

2.1 Animated transitions between bar and pie charts

As already mentioned, Heer and Robertson[HR07] implemented animated transitions. One between scatter plots and bar charts, a transition between stacked bars and grouped bars and another one between bar and pie chart. For this purpose, they developed a framework called DynaVis, a visualisation system featuring animated data graphics. In their user study, they compared staged animated transitions to plain animation to the static visualisations. The difference between the staged animated transitions and the plain animation is, that in the plain animation only one stage exists where all elements move to their destination at once, while in the staged version there is a time delay which should allow the user to follow the elements more easily. For the bar to pie chart animation, they transformed all bars in the bar chart at once into a doughnut chart, which might be confusing and left space for improvement. With the two controlled experiments they performed they found that animation makes it easier to follow two targets across different visualisations and it makes it easier to estimate changes in value.

In 2010 Danyel Fisher [Fis10] used the DynaVis framework to create two animated transitions between bar and pie charts. One he described being less successful because in the animation the length of the bar is not transformed into the length of the segment in the pie chart, which he then describes that it violates the animation principle of maintaining the invariant. This principle describes that the representation of a data value should not change in height during the animation. In his second approach, which can be seen in Figure 2.1, he followed this rule and created an easy to follow animated transitions between bar and pie chart. But unfortunately, he did not test his approaches if they are beneficial to the visualisation and he did not implement them controllable. Most likely because the DynaVis framework does not support this.

2.2 Studies on animation and other animated transitions

A general question in animation is which transition timing functions should be used for animation or if a linear timing function is sufficient. Research on this topic and the use of timing functions in animated transitions showed that ease-in/ease-out is the most beneficial timing function [DBJ+11]. Dragicevic et al. concluded that this might be because this timing function maximises the predictability of the motion. Because of their

findings, ease-in/ease-in was also used for the animations in this thesis.

A summary of many animation studies has been done in 2002 [TMB02]. The authors tried to conclude when animation might be useful in general and when it might not be beneficial. Their major findings, which are also relevant for this work, are that animation might only be beneficial when the animation convey additional information[BB99] and when the animation is not too fast and complex for the user to properly perceive and follow it.

Tools for animation, for instance, google motion chart, help to build animated transitions more easily. A comparison of those and when they are helpful was done by Battista in 2011 [BC11]. But those tools are primarily useful when doing animations within one type of visualisation, like value changes or sorting operations, for example for trend visualisation [RFF+08].

Before animated transitions for simple charts were implemented and studied, researchers developed animated transitions for other purposes as well, for instance for radial graph layouts [YFDH01], or looking at animation in trend visualisation and the benefits of small multiples displays [RFF+08]. Or animated transitions in tree-map visualisations for learning spatial relationships [BCK05].

When introducing new visualisations it is useful to teach the visualisation by analogy [RM15]. For this purpose animated transitions might also be useful to break up new visualisations into more familiar charts. When designing new visualisations it is also important to consider the efficiency of screen usage, because the size of the visualisation also has a great impact on the usefulness of a visualisation [HKA09]. Another approach to convey changes in the visualisations more clearly is to break up the changes in atomic changes and animate those consecutively [MWTI19]. To minimise the amount of occlusions during the animation Mizuno1 et al. cleverly ordered the atomic changes. Others applied transitions to map representations of graphs [HET+20].

CHAPTER 3

Controllable Animated Transitions

In this chapter three animated transitions will be introduced. Two approaches for a bar to pie chart transition and one transition between the different levels of a hierarchical bar chart.

3.1 Terminology

Apart of common terms used in computer science the meaning of some terms in this thesis have a more specific meaning in the context of animated transitions. Therefore a brief definition of the terms in this thesis:

- **Controllable:** We consider an animation controllable if the user can start and stop the animation at any time, control the animation speed and jump to any point in the animation. For this purpose, a seek-bar is used along the implemented animated transitions. A seek-bar is a bar which allows the user to control the progress of the animation which a slider, similarly to the control bar of a video player. The term in this context does not include being able to control parameters of the animation (e.g. sizes, scales, spacing, colours, etc.).
- **Animated transition:** An animated transition refers to the animation which transforms one visualisation into another one by applying a series of transformation matrices.
- **Steps and stages:** The term step is only used in this chapter and refers to the labels in the different approaches for the bar to pie chart animation (Figure 3.1 and 3.2), while the term stage describes an animation part that transforms between

two visualisation types. For example in the bar to pie chart animation four stages exist, bar chart to Gantt chart, Gantt chart to stacked bar chart, stacked bar chart to doughnut chart and doughnut chart to a pie chart.

- **Element:** An element is the visual representation of a data value, for example, a single bar in the bar chart is an element as well as a pie in the pie chart.

3.2 Bar to pie chart animation

The purpose of a bar to pie chart animation is to smoothly transition between the bar representation of a data set to its pie chart representation. Since bar charts are used to compare absolute and pie charts to compare relative values, this transition promises to better convey the relationship between absolute and relative values. There are multiple ways to perform this animation. The most straight forward way would be a linear interpolation between the bar and pie chart. Linear Interpolation leads to overlaps between the bars of the bar chart/ segments of the pie chart. This overlapping is confusing and makes it difficult to track the bars along the animation. Which is why one focus of designing the transitions was to prevent overlaps, which we will later see is guaranteed for the bars in the proposed approaches. Another point of focus was to move as few elements in the animation as possible at a given time so the user can follow the animation more easily. Since we want to maintain the invariant, which is the length of the bar in the bar-chart, this length should be transformed into the arc length of a segment in the pie-chart. Using the fixed length side of a bar in the bar-chart is also possible, but this would result in the transformation of a fixed length into a variable length and vice versa. In other words, using the height of a bar as the radius of the pie-chart makes no sense, because the height of the bars is variable while the radius of the pie-chart always stays the same. Due to those limitations, it seems that using a doughnut-chart as an intermediate stage is mandatory.

In Figure 3.1 and Figure 3.2 the two schematics of the approached transitions between bar- and pie-chart can be seen. Later referred to as rotating bars approach (RBA) and sugar cane approach (SCA). Note that the schematics stop with a doughnut-chart instead of a pie chart. This is due to the trivial closing transition between doughnut and pie-charts. Figure 3.1 was the initial proposed animation. It utilises a radial bar chart as an intermediate stage in order to reach a circular state, which happens in step 1-4. Afterwards, the bars rotate and bend to form a doughnut chart, which is shown in steps 5-8.

The arguably better approach (SCA) to perform the animated transition is shown in Figure 3.2. It utilises a waterfall bar chart or also known as Gantt chart where the axes are swapped and the elements are non-overlapping, as an intermediate stage (step 3), even though it is a waterfall bar chart with strictly monotonously increasing values. Steps 1-3 show the transition between the normal bar chart and a waterfall bar chart by simply translating the bars up. Note that in step 2 five bars are still aligned, meaning that in

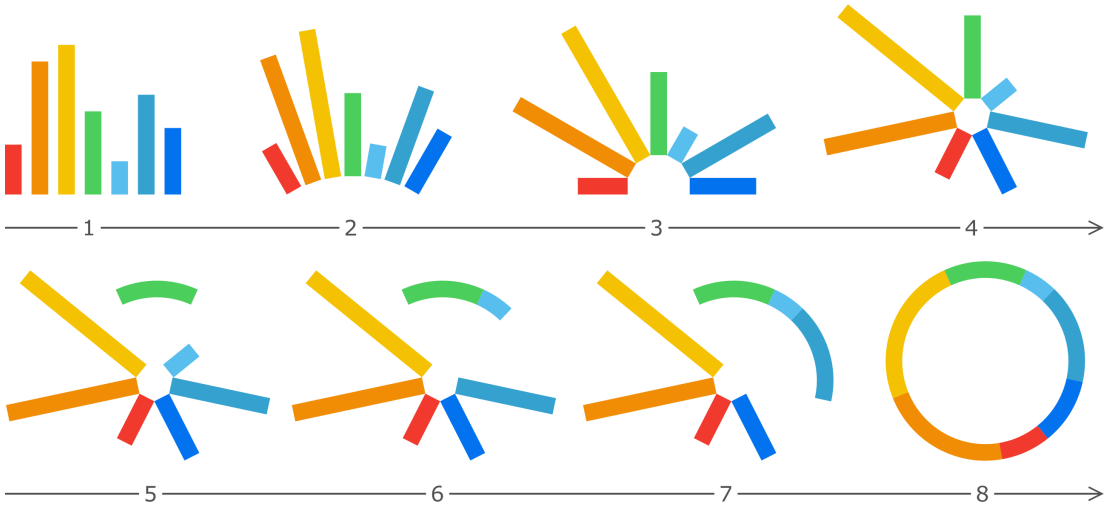

Figure 3.1: Rotating bars approach (RBA) - Initially proposed animated transition between bar and pie charts

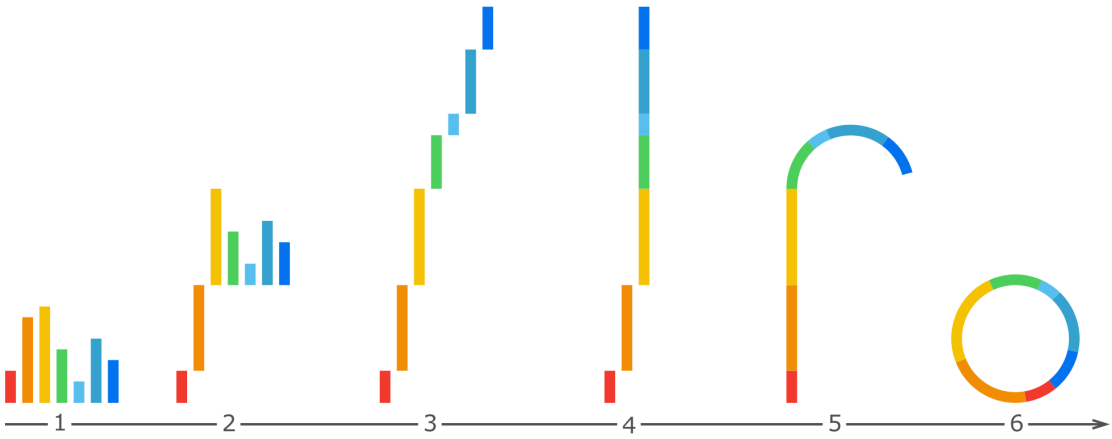

Figure 3.2: Sugar cane approach (SCA) - Better and implemented animated transition between bar and pie charts

this case every bar starts the translation at the same time but the endpoints are reached at different times. In step 4 the bars get stacked together, forming a stacked bar chart with only one long bar. Notice that the bars are aligned in this transition as well. Step 5 shows a more complicated animation, by rolling the stacked bar chart into a doughnut chart.

The two approaches for the bar to pie chart animation both have advantages and disadvantages, which were discussed to decide which animation to implement. Advantages of the RBA are that the animation takes less screen space then the SCA since the waterfall bar chart takes up a lot of vertical space. Another advantage of the RBA is, that no or little zooming out/ zooming in is required during the animation, since the animation can only extend to the sum of the two largest elements, and only if they happen to be opposite of each other in step 4, while in the SCA the vertical space needed is always the sum of all elements. A major disadvantage of the RBA compared to the SCA is, that it requires a lot of rotations, while in the SCA the only rotations are needed between the stacked bar chart and the doughnut chart (step 5). Another good point about the SCA is the simplicity of step 1-4, making it easier to follow the animation as well as easier for the implementation. Further, the SCA guarantees, that the elements never overlap during the animation, while in the RBA elements may overlap in step 5-8 (this depends on how and when the rotation of the bars is done). Another advantage of the SCA is labelling. When the bars of the bar chart are labelled vertically on the left side of each bar the labels never overlap the bars during the animation, while in the RBA there is no simple solution to add labels that will never overlap the bars. How the labelling problem was solved in the SCA can be seen in the results section.

A summary of the advantages of the SCA and why this approach was picked for implementation:

- Simple animation in step 1-4 (only translations needed)
- Only 1 rotation stage
- Labels do not overlap the bars (but may overlap each other though)

3.3 Hierarchical bar chart animation

The animation for the hierarchical bar chart is derived from the SCA of the bar to pie chart animations. It is not a single chart but an animation between the different levels of hierarchical data. The problem with displaying all levels of the hierarchy at once is that it creates clutter (Figure 3.3 illustrates this problem), which makes the chart harder to read. Further, information is displayed which might not be needed. For instance in Figure 3.3 the two lower levels (product categories and the consumer, corporate, home office and small business categorisation) might not be needed, instead the user is only interested in the data per destination (central, east, south, west). An animation for hierarchical bar chart should solve this problem, by only displaying a single layer at once

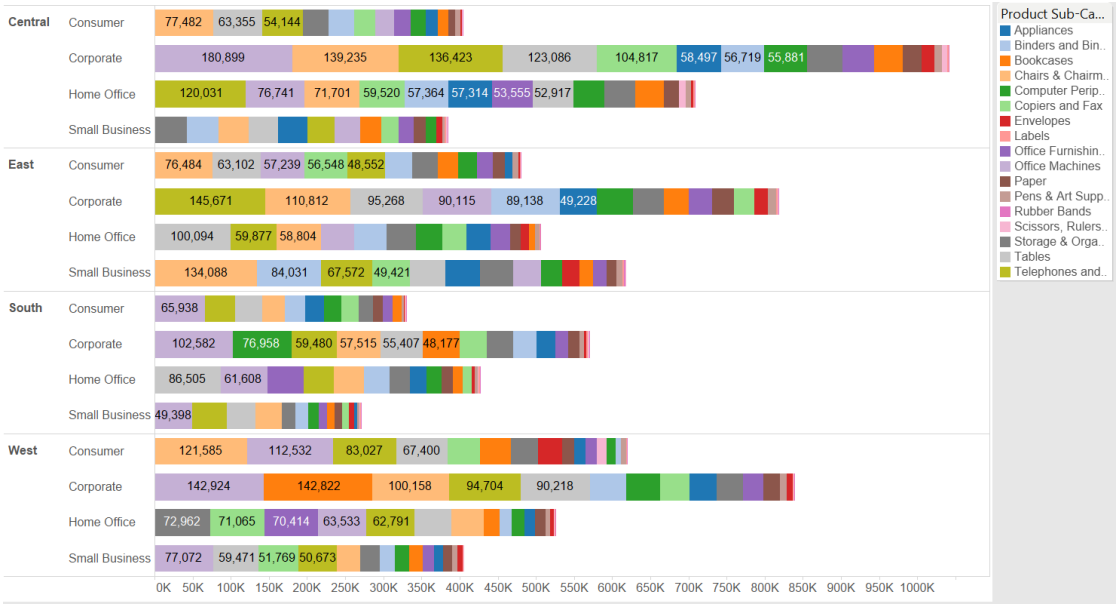

Figure 3.3: A hierarchical bar chart with all layers visible [Tre19]

and transitioning between adjacent layers. For this purpose grouped stacked bar-charts are formed as intermediate stages to represent the hierarchical data. This means that all elements which share the same parent element get grouped together using the first animation stage of the approached animated transition between bar-chart to pie-chart. Therefore the transition has as many stages as the data has layers. At each given point only the labels of the current level are visible. The stacked bar-chart is then further transformed into a bar-chart by blending from the bar colours of the previous level to the colour of the next level. The labels of the previous level, which are located at along the left side of each stacked bar at this point, are simply fading out and the label of the next level is fading in for each bar. Note that the transition between a stacked bar-chart to a regular bar-chart requires the bars to have distinct colours or some other way to differentiate them in the stacked bar-chart. For instance a border around the bars or vertical spacing could be used. For this thesis a colour distinction was used. For this purpose each element got assigned a colour from a colour-set which is predefined.

CHAPTER 4

Implementation

The animations were implemented using JavaScript together with WebGL. The lightweight utility library WebGLUtils was included to simplify the usage of WebGL. Further, the CSS Framework W3.CSS was used for styling the GUI of the animation as well as the poll. WebGL was used for rendering because it allows implementing interpolation calculation on the GPU instead of the CPU, which creates a much smoother and faster animation. Using HTML canvas would only utilise rendering on the GPU, but the calculations for the interpolations would need to be done entirely on the CPU. The system the implementation was tested on has an Intel Core i7-6700k and a NVIDIA GTX-1060 6GB together with 16 GB of DDR-4 memory.

Figure 4.1 illustrates the basic workflow. The user starts by picking a CSV (comma separated values) file containing data in a simple format. Examples for the file formats can be seen in Figure 4.2 and Figure 4.3. Both contain the animation type in the first row, followed by the data. The file format for the bar to pie chart animations consists of 2 entries per row, the name of the bar followed by its value. The file format for the hierarchical bar chart is defined similarly. The second row contains the number of levels

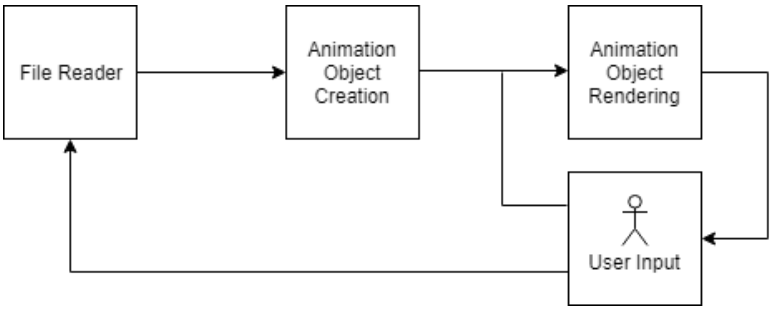

Figure 4.1: Basic Workflow of the Implementation

bartopie	
Anton	20
Berta	40
Ceasar	100
Doris	50
Emil	25
Friedrich	20

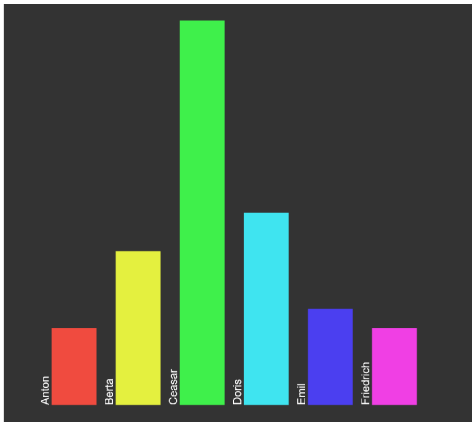

Figure 4.2: Bar-to-pie file format example and its output

the hierarchy has, afterwards, the data starts level by level. In the example given (Figure 4.3) the top-level hierarchy is "food", containing two sub-categories, the second level, "coffee" and "bread". This schematic leads down to the bottom level, containing the names of the bars together with their values. The file reader simply stores this data into a two-dimensional array and calls the corresponding animation object creation method depending on the animation type. Those methods create all animation objects needed for the animation and calculates their properties. Afterwards, the rendering loops simply renders all the objects depending on the progress of the animation, which is controllable by the user.

4.1 Animation primitives

In the context of this thesis an animation primitive consists of a class containing all its parameters, for instance, the start and endpoints, the interval in which the primitive is visible, the colour etc.. Each animation primitive has its own dedicated shader, which calculates the actual positions an animation object has at a given time. Meaning the shader defines the path the object moves along as well as the interpolation type used (linear, ease). For the implementation of the bar to pie chart animation and the hierarchical bar chart animation, only two animation primitives were needed (Figure 4.4). A linear animation primitive, which simply does linear interpolation between the start and the end position, and a circular animation primitive, which is used to perform the transition between the stacked bar chart and the doughnut chart. The circular animation primitive uses linear interpolation together with circular interpolation. This creates the rolling movement needed for the stacked bar chart to a doughnut chart transition. Note that in Figure 4.4 rectangles were used for illustration, but the animation object can consist of arbitrary shape and number of triangles. For the two animations, only rectangles were used.

hierarchicalbar							
3							
food							
coffee				bread			
cappuccino		espresso		whole wheat		multi grain	
online	113	online	40	supermarket	24	supermarket	84
store	250	store	69	bakery	62	bakery	14
				online	34	online	76

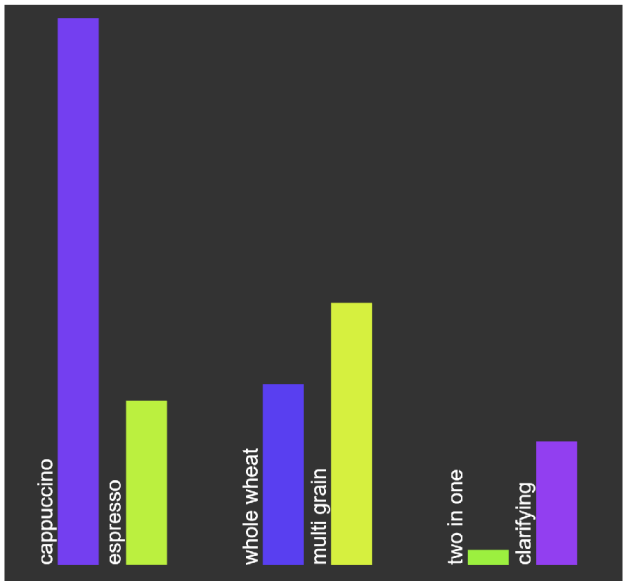

Figure 4.3: Example for the hierarchical bar chart file format (second level of the data is shown in the bar chart)

4.2 Animation construction

In the animation construction (in Figure 4.1 it is called animation object creation), all the animation primitives are initiated by calculating all the positions and time values needed for each animation stage. Further, this step adds camera points, which describe the movement and zoom of the camera, making the camera follow the animation and always keeping the whole animation visible. Since the spatial extension of each animation stage can be calculated from the bar chart data, the camera zoom can be pre-calculated. Figure 4.5 illustrates the initiation of the first animation stage of the SCA. Note that the camera aspect ratio is not getting changed, changing the aspect ratio seems to be confusing and changes the bars XY proportions. A downside of this is that it leaves empty spaces on both sides of the chart.

For the third Stage of the bar to pie chart animation many small segments are generated to create the illusion of a bar bending into a circle, but, as common in computer graphics, the circle is approximated by many triangles. A disadvantage of this method is, that the vast amount of small triangles create aliasing problems, which can sometimes reveal

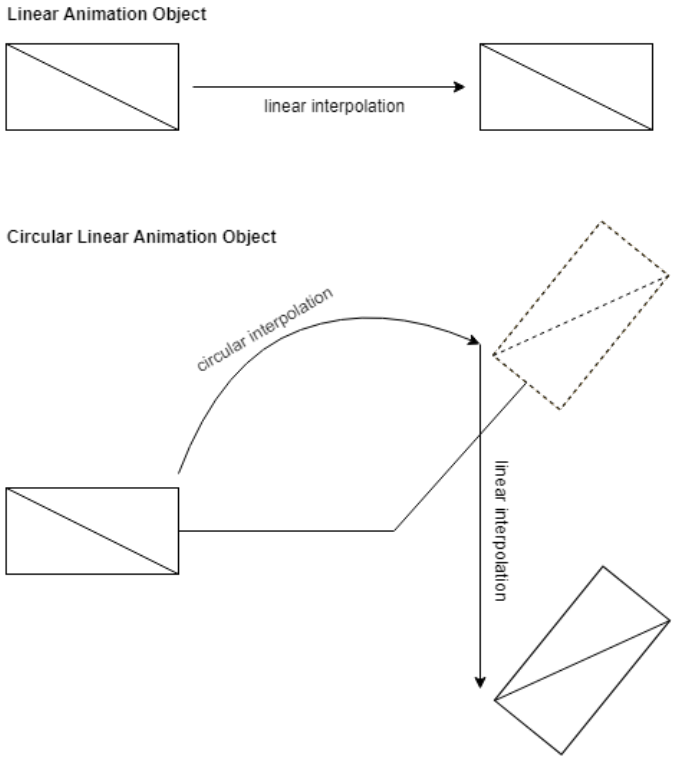

Figure 4.4: Animation primitives used in the animations

parts of the background between the segments.

4.3 Camera, text and labels

The camera implemented for the animations is a simplified version of a regular OpenGL camera[Lea]. The camera movement is restricted to a plane and its projection matrix consists of X and Y translations values and a zoom value, therefore not supporting roll, pitch or yaw movements. The camera includes an array of translation and zoom values with a corresponding time value. These values are used for linear interpolation to perform a smooth camera transition. For example in Figure 4.5 the camera performs a X translation as well as a zoom.

For labels and text, there are two ways to implement them in a Web application with WebGL. One is by using HTML and CSS to render and position text. The disadvantage of rendering them as HTML elements is that it only works in a 2D context because HTML elements don not have a z location. Depth can only be achieved with this approach by adjusting the render order. In this context, this would not be an issue since the animation does not have any depth and is performed on a 2d plane solely. But this approach would require to calculate the screen space coordinates for the text first, which is why the

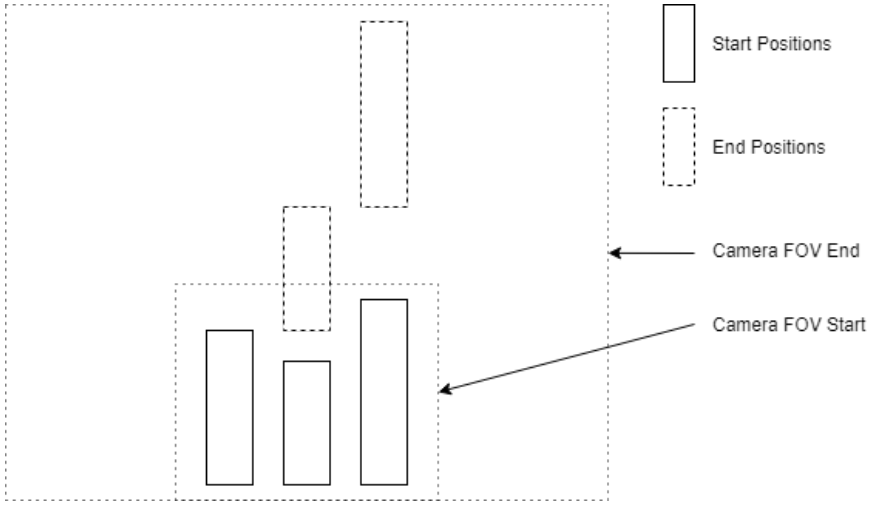

Figure 4.5: Construction of the animation stage 1 of the SCA

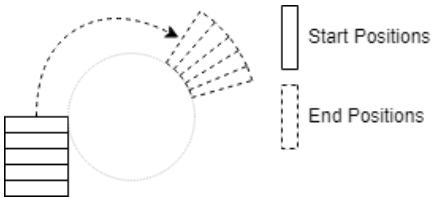

Figure 4.6: Construction of the animation stage 2 of the SCA

second way of text rendering in the browser was used, which is rendering it using WebGL textures. A pseudo HTML element is used to render the text first, which is then saved into a texture which later gets used to display the text in the scene. A disadvantage of this is that it sometimes results in lower quality of the text because of aliasing issues. Using textures as the text allows us to handle the labels just like all other elements in the scene.

CHAPTER 5

Results

Unfortunately, it is laborious to showcase an animation with images, which is why a video was also submitted as a supplementary, which shows both animation types. Though, images of the animations are included here. The collage in Figure 5.1 shows the key-frames of the bar to pie chart animation (the collages are ordered in rows). Figure 5.2, 5.3 and 5.4 display the animation stages 1-3 in more detail. Figure 5.5 illustrates the second implemented animated transition, the hierarchical bar chart. In this example with three levels as shown in 4.3. The green bar at the bottom on the seek bar indicates the animation progress.

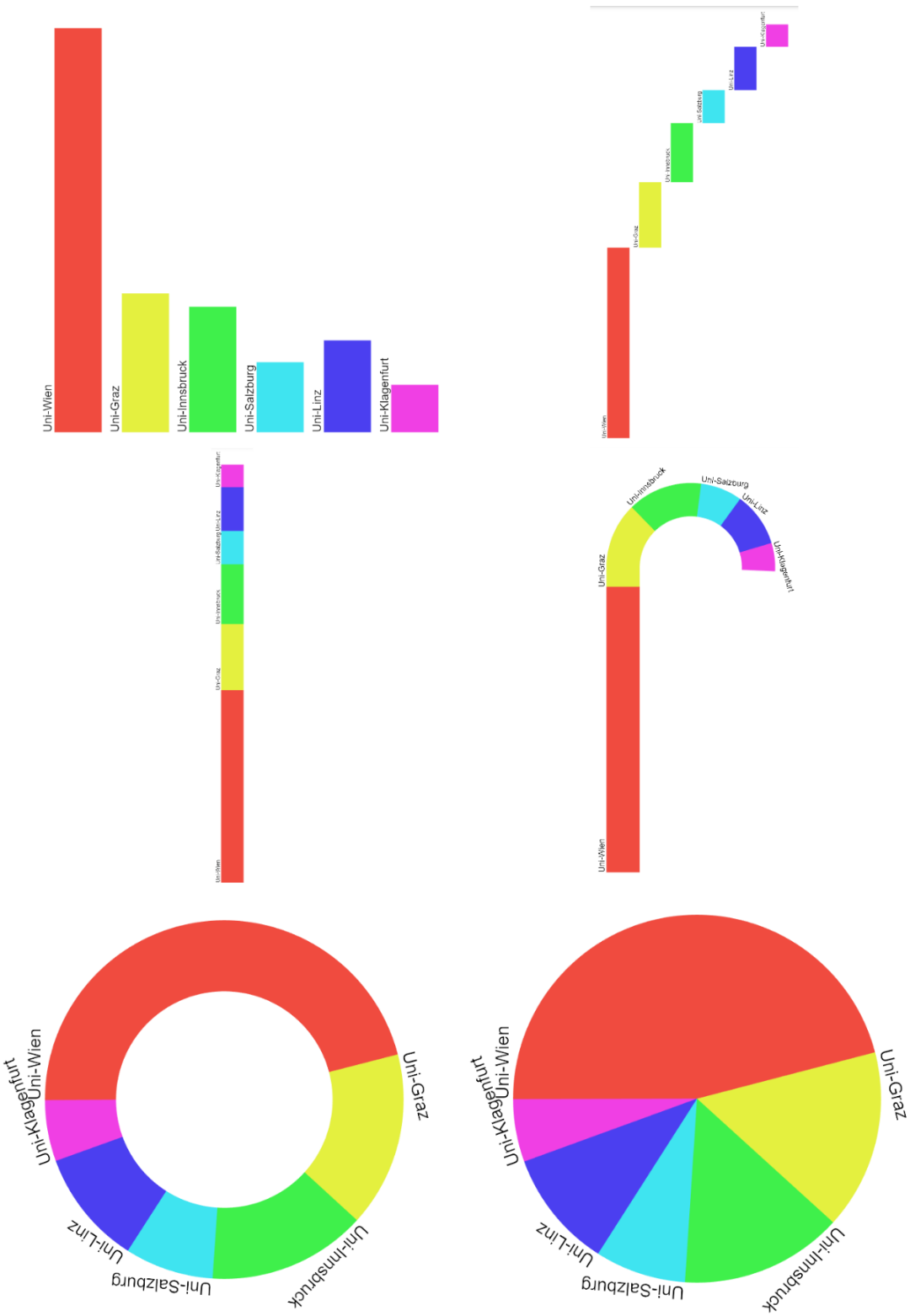

Figure 5.1: Key-frames of the bar to pie chart animation

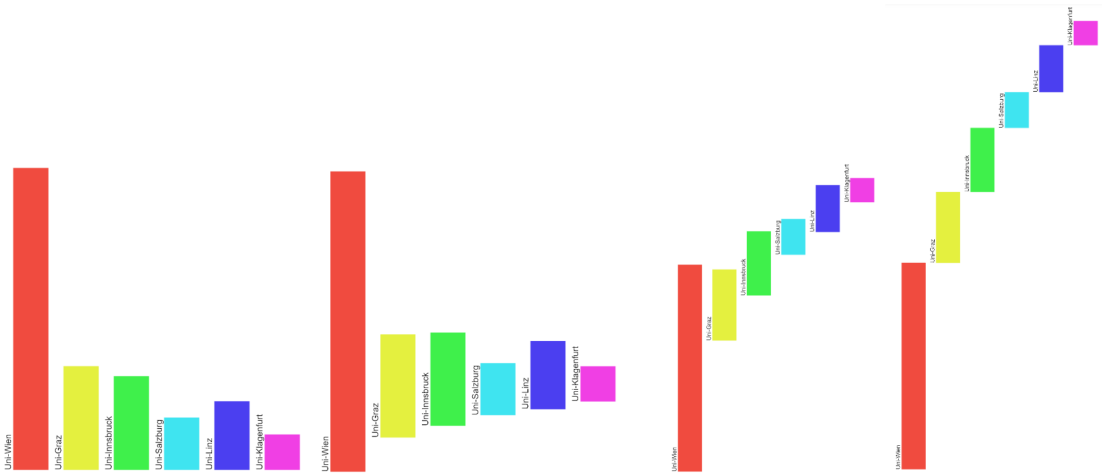

Figure 5.2: Animation stage 1 - detailed view

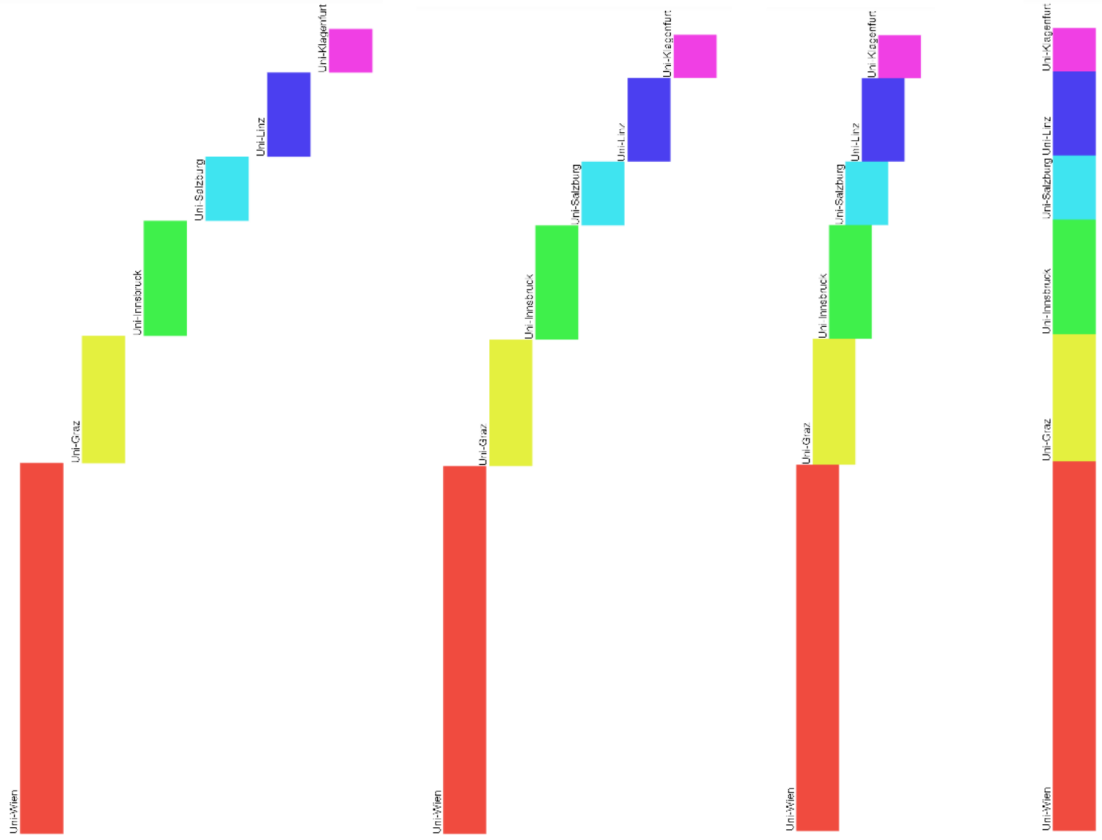

Figure 5.3: Animation stage 2 - detailed view

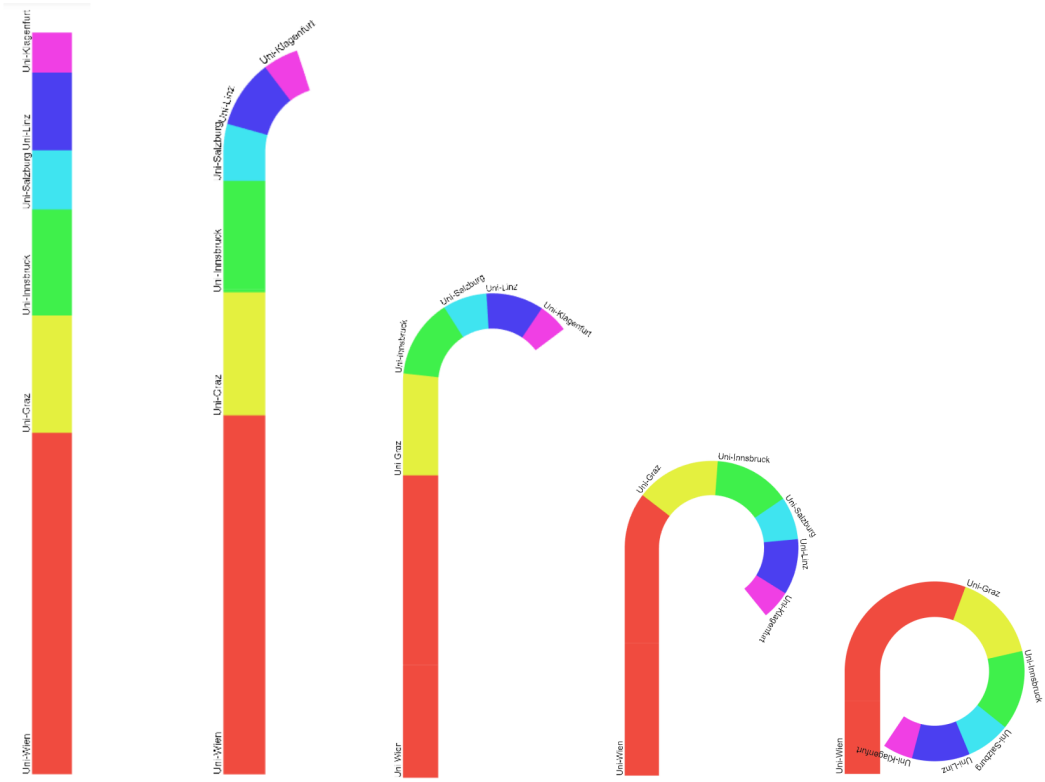

Figure 5.4: Animation stage 3 - detailed view

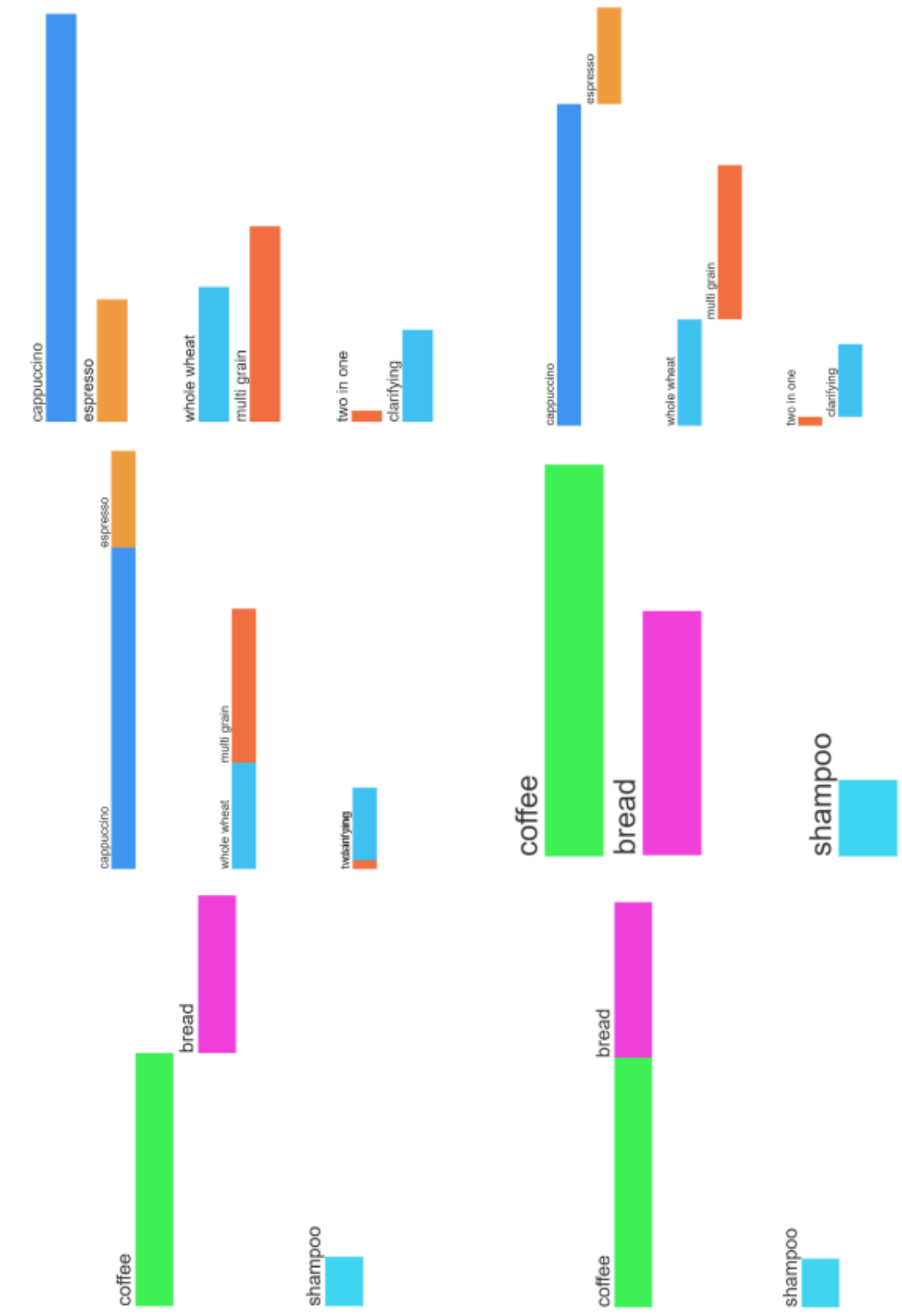

Figure 5.5: Key-frames of the hierarchical bar chart animation (data with two levels)

CHAPTER 6

Evaluation

An animated transition like the once presented in this thesis might be engaging and interesting to look at, but it is also interesting to see if those animations bring any advantages with them. Interesting to evaluate is, if animated transitions perform better in accuracy tasks compared to their static variants, if animated transitions spark users to spend more time on the visualisations, or if controllability of the animation is a desirable feature in animations. But before talking about the evaluation of the animated transitions some words about the performance of the implementation.

6.1 Performance

Since it is an animation with a lot of objects to render in some stages, it is important to note that devices without graphics cards the animation likely lags, i.e. the frame rate drops. In particular, the circular stages in the bar to pie chart animation can cause problems. Tests with more than 1000 elements were only smooth on devices with a graphics card, while devices with only an integrated graphics processor (laptop, smartphone) run smoothly with less than 500 elements. Figure 6.1 shows the rendering performance of the circular animation stages. Around 5200 elements the animation dropped below 60 frames per second. But of course, reducing the number of elements in the circular stages affects the smoothness of the circle as well as reduce the accuracy by which those segments can represent the original bars. A solution to this problem would be to split the segments into two at the points where one bar ends and another one starts, to represent them accurately. Technically, with a very large data-set, for example, with approximately 500 data values, the animation would also start to slow down because of the many bars. But since a bar chart with 500 bars is not readable anyways this cannot be considered an issue.

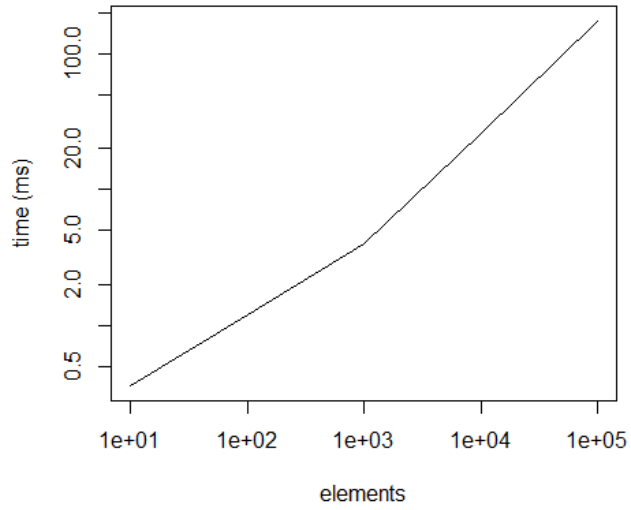

Figure 6.1: Performance of the circular stages on a GTX1060

6.2 Qualitative evaluation

Ideally, it would be desirable to test a hypothesis that compares the effectiveness of static charts to the animated transition, but the problem with this more generic hypothesis is that many parameters might affect the result but are hard to limit. For instance, the fact that a vast majority of people are already trained in reading static bar and pie charts but are not familiar with animation in charts. Or that in the static variants you can show both charts at the same time on the screen while in the animation you cannot. For those reasons, we decided to test the controllable animation against its non-controllable variant by simply prohibiting to control the animation progress. This lead to the formulation of the following hypothesis:

> A controllable animated transition between bar and pie chart improves accuracy in absolute and relative value estimation of the same data-set compared to a non-controllable animated transition.

We decided to only test the bar to pie chart animation against this hypothesis since it is the main animation of this project and it simplifies the user study for the users by only confronting them with one animated transition. Apart from that, it is reasonable to assume that results from the bar to pie chart animation also apply for the hierarchical bar chart animation since it utilises the first two stages of the bar to pie chart animation and therefore from the animation viewpoint they are very similar. To consider a result

significant, the significance level was set to $\alpha = 0.05$. As the hypothesis already states the user study tests the animation against absolute and relative value estimation, which leads us to the design of the user study.

6.2.1 Design of user study

The user study consists of three different types of tasks, an estimation task on a bar chart(1), an estimation task on a pie chart(2) and a value comparison of sums of elements(3).

Figure 6.2 displays a question type (1). The goal of this task is to select the largest and smallest element in the data, which is easiest to do in the bar chart stage of the animation. The major point of this task is, that the first animation stage where the bars get aligned on top of each other allows the user to easier compare different bar heights. But this can only be done properly in the controllable scenario where the user can pause the animation.

In Figure 6.3 the question (2) addresses the benefits of pie charts compared to bar charts in terms of relative value estimation. Since the controllability allows to quickly jump between the bar chart and the pie chart, it is easy to count the number of elements on the one hand, and on the other hand, estimate the percentage of a segment in the pie chart. This might especially be beneficial if the bars are similar in height because this allows the user to calculate the percentage of a segment if all bars are the same height. Which further makes it easier to estimate the percentage in the pie chart without the need to convert degrees to percents.

Type (3) is shown in Figure 6.4. This task type aims to utilise the animation in stage 2. In this task type, the user has to estimate which sum of elements is greater. The sum of the blue elements or the sum of the red elements. In stage 2 of the animation, those elements are aligned on top of each other which again allows the user to compare the values more accurately once the bars are aligned on top of each other.

In total, participants had to answer 18 questions, 6 of each question type, of which 3 were controllable and 3 non-controllable. Thus 9 tasks were controllable and 9 non-controllable.

6 different data-sets were used for the evaluation, 2 small, 2 medium and 2 large-sized ones. We considered data-sets to be small with 10 elements, medium ones with 20 and large ones with 30 elements. The data-sets were filled with arbitrarily chosen data: "most common forenames in Austria 2017", "national council election Austria 2017", "saving interests in Austria 2000-2019", "students per university Austria 2017/18", "list of metropolitan areas by population" and "all-time Olympic games gold medals". All data-sets were used for each evaluation task type. therefore every data-set was used 3 times but for a different task. The tasks were shuffled to avoid that a learning effect occurs.

Apart from the tasks, the user was able to choose between English and German as the survey language and the users also had to enter their age and profession. The controls

were explained by analogy to a normal video player. Further, the user was shown a training task with a data-set of 5 elements where he had to select the largest element. Purpose of this task was to make sure the user understood the controls and how to select elements in the animation. It is worth mentioning that the picking mechanism, which allows the elements in the bar to be picked along the whole animation, is implemented using another frame-buffer, where the id of an element is represented as a colour value, also known as frame-buffer picking.

The data which was collected for the evaluation includes:

- Controllability - whether or not the task was controllable
- Error - if the user answered the question correctly
- Time - the total time needed to complete the task
- Time per stage - the time a user spent in a specific stage of the animation, therefor four time-values for the four stages
- Input Events - the number of input events which were triggered by the user, a good indicator of whether or not the user dragged the seek bar or not
- Clicks on Controls - basically same as input events, but does not indicate if the user dragged the seek bar since in a drag event this counter only increases by one
- Confidence - the user was asked to enter a confidence level, from 1 - less confident to 5 - more confident
- Age
- Profession

We now have discussed how the user study was set up and which data was collected, which brings us to the evaluation of the data and the results.

6.3 Quantitative evaluation

The goal of the quantitative evaluation was to reach $n \geq 30$ participants since this value is an established boundary between small and large data samples and it allows to use the sample standard deviations to estimate the population standard deviation. In total 33 people participated. With 18 tasks absolved per person, in total 594 tasks were absolved. The average age of the participants was 26.42 years, which is not surprising since mainly students participated. A histogram of the age can be seen in Figure 6.5. Note that this forms 2 groups, a younger and an older group, which will get interesting later.

Now, the way the 594 were analysed, was by comparing the error rate of the controllable to the non-controllable in different subsets of the tasks. Table 6.1 lists the full analyses of

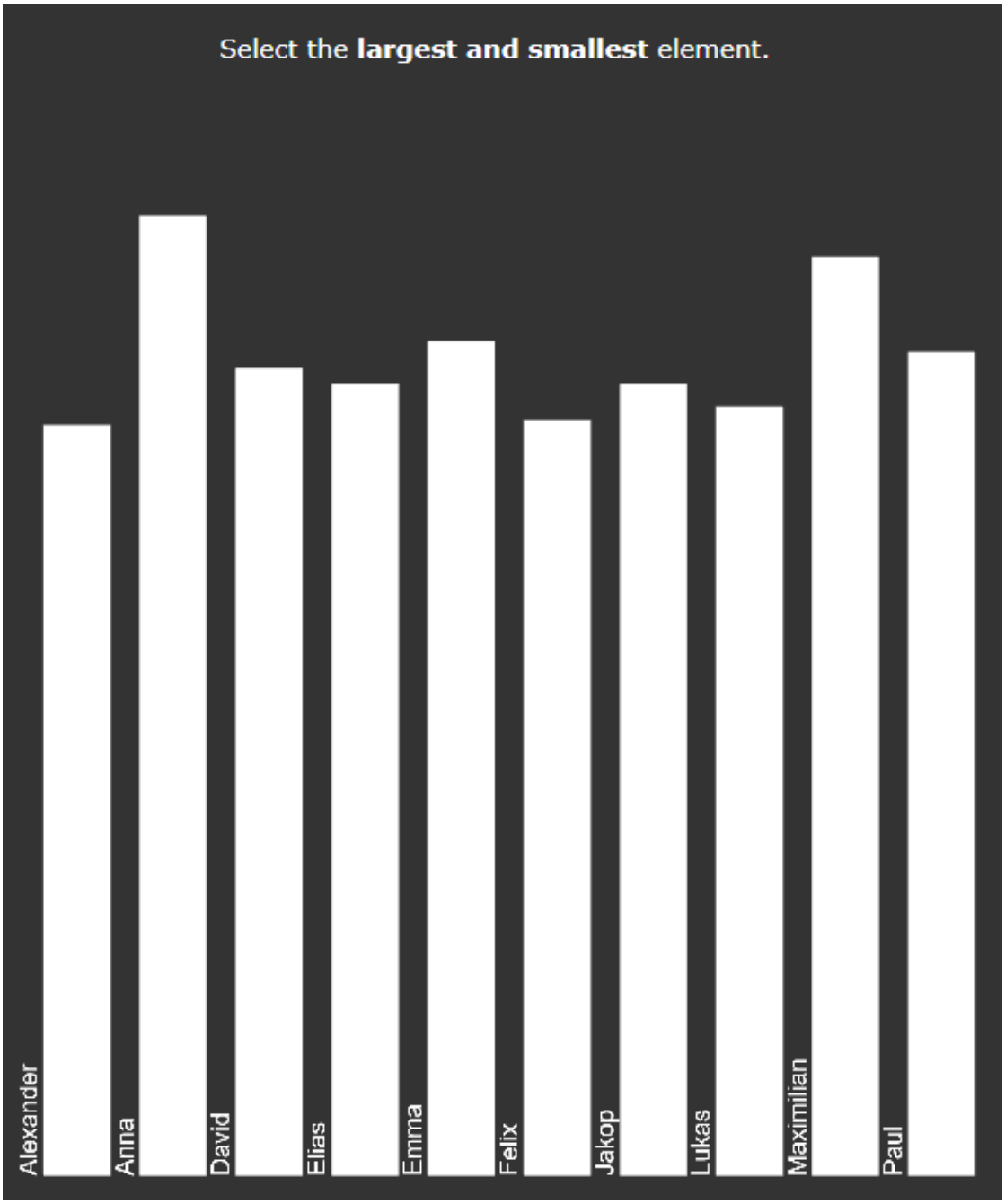

Figure 6.2: Evaluation question type with focus on bar charts(1)

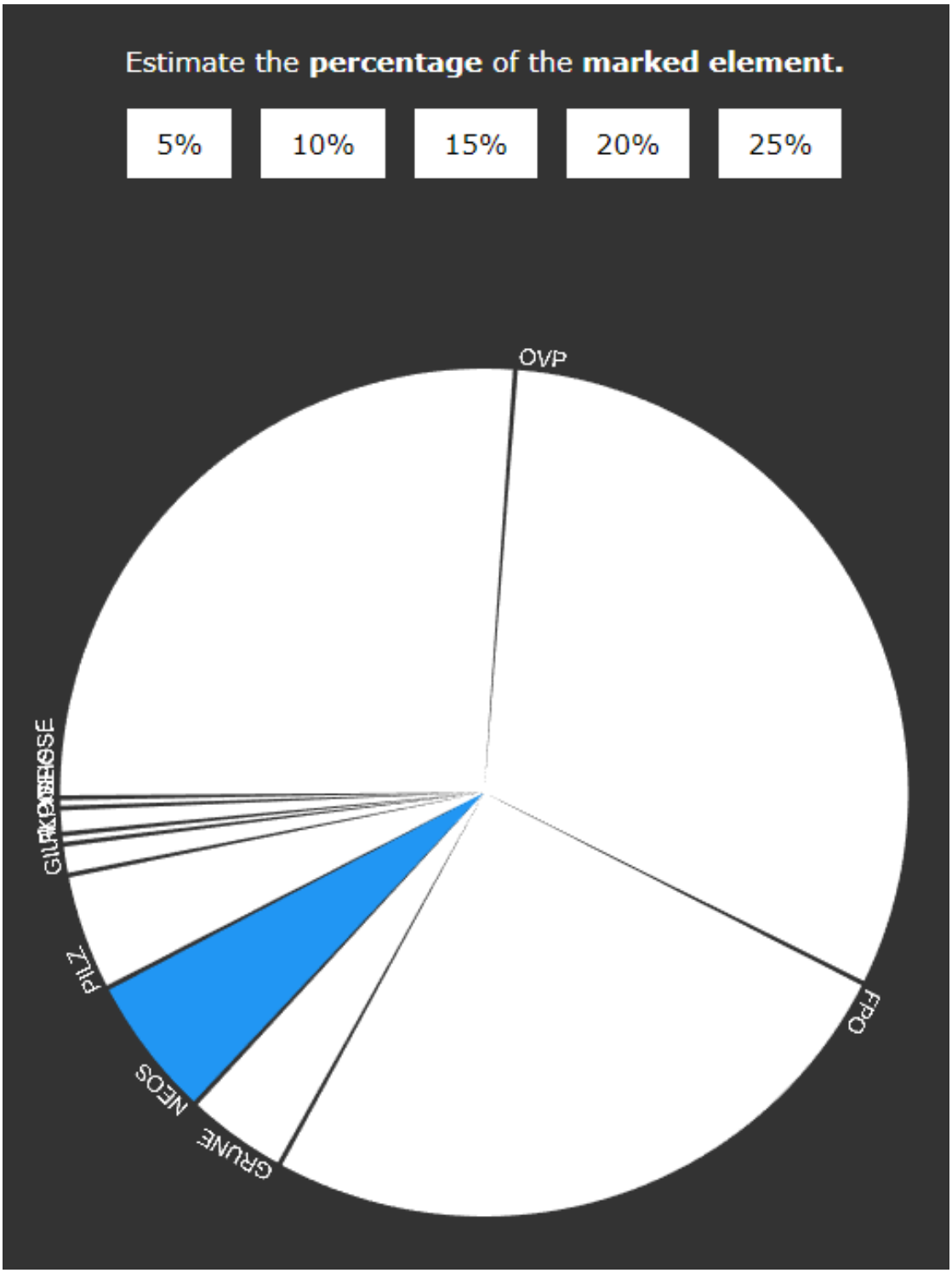

Figure 6.3: Evaluation question type with focus on pie charts(2)

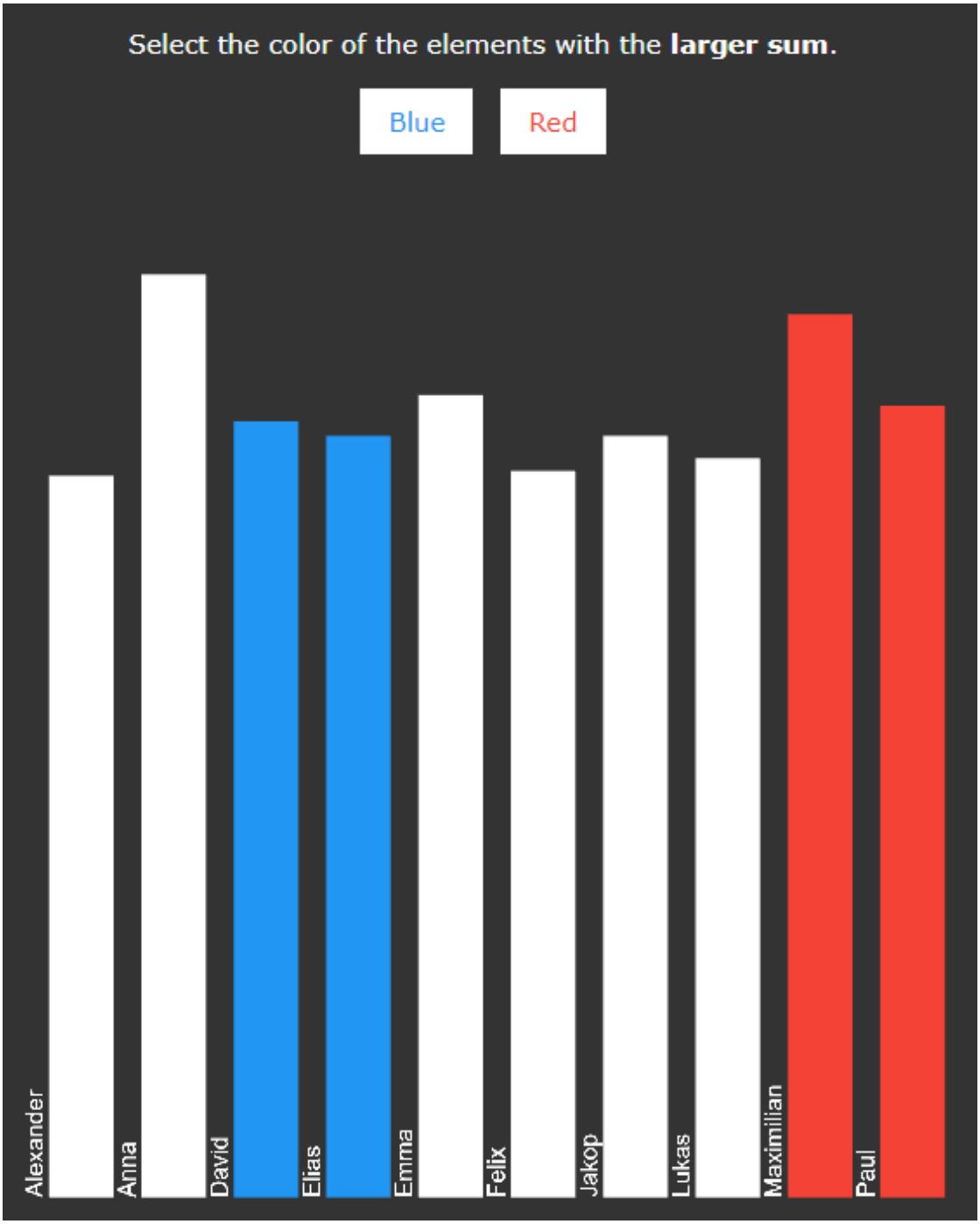

Figure 6.4: Combination question type(3)

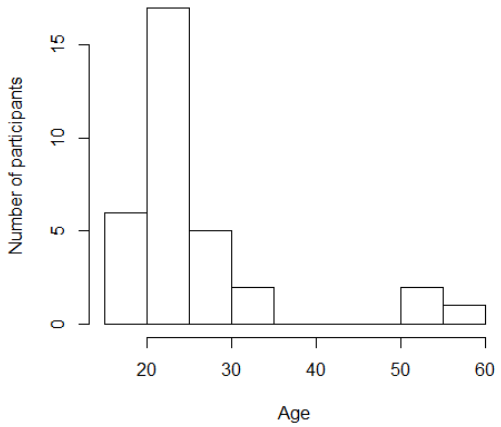

Figure 6.5: Age Distribution of the participants

Evaluation type	Controllable		Non controllable	
	# Tasks	Error rate controllable	# Tasks	Error rate non-controllable
All tasks	297	0,269	297	0,313
First animation stage left	115	0,278	123	0,260
Bar chart tasks	99	0,364	99	0,384
Pie chart tasks	99	0,354	99	0,414
Mixed tasks	99	0,091	99	0,141
Age > 40	27	0,26	27	0,518
Age <= 40	270	0,27	270	0,296

Table 6.1: Evaluation results and whether or not they were significant (green: significant)

the data. As shown, the overall conversion rate, as well as 6 subsets were analysed. The subsets include a subset with tasks where the first animation stage was left, meaning that the user either started the animation and left the first stage (the upward translation of the bars), or the user jumped to another stage by clicking onto the seek bar. The other subsets include the three different types of tasks and subsets by age. The age 40 was used as a threshold because as mentioned before, the participants formed 2 age clusters, and 40 is a boundary between those clusters. For testing the statistical significance, an A/B test in R [RTh] was performed. As the colour highlighting indicates only the test for the participants above the age of 40 was significant (with $\alpha = 0.05$). All other tests were not statistically significant. Unfortunately, with 54 tasks, which corresponds to 3 persons, the group of people above the age of 40 is very small. An analyses of the time needed per task revealed, that controllable tasks were absolved quicker with an average time of 18.91 seconds while non controllable tasks were finished after 21.17 seconds ($p = 0.0746$). This is most likely due to participants being able to jump to the wanted animation stage where the tasks were the easiest to solve.

CHAPTER 7

Conclusion and future work

Due to the results, the hypothesis must be rejected. At least in its generalised version. With 30 people below the age of 40, the result for the younger group reveals, that the controllability does not lead to better accuracy in estimation tasks. But on the other hand, the group of people aged above 40 performed better on the controllable version of the animation, which may have a various reason, which leads us to the conclusion.

7.1 Conclusion

For the small group of the participants above the age 40 (in fact, the persons in this group were even over the age of 50), the result suggests, that this group performs better on a controllable animation. A plausible reason for this might be a difference in eyesight because assumable the older participants had worse eyesight than the younger ones. While it might not be necessary for people with a better vision to have a controllable animation to estimate a value precisely, older persons might profit from being able to stop the animation and extract information from the intermediate stages of the animation. Which would support the findings of Barbara Twersky and Julie Bauer Morrison[TMB02] that animation is superior in cases where the animation contains additional information compared to the static visualisations. In our case, for instance, one intermediate stage contains more information about the sum of elements.

Some of the younger participants reported they used rulers or other helpful tools to answer the questions more precisely. But none of the 3 older people did, which might be another reason for the different results of the two groups. They might have taken the instructions to estimate the values more seriously, while the younger participants took more effort into answering the questions correctly. The evaluation supports this idea because the error rate differed in the two groups. With an error rate of 0.518 the older participants had a higher error rate than the younger participants with an error rate of 0.26.

7.2 Future work

From a technical viewpoint, the implementation could be extended to create a more generalised animation framework. Or additional animated transition between other static visualisation could be introduced and implemented. For example, more abstract animation primitives could be implemented to make it easier to build a new animation, instead of having to rely on graphic primitives where the mesh is user-defined. Another improvement would be to implement the different stages of the animated transitions as animation modules so they can be easily reused in other animated transitions.

The hypothesis, as described, only tested one case in which controllable animated transitions might be superior, but other benefits may exist, therefore more evaluation must be done to answer the general question if controllability or animated transitions are useful and in what context. For instance the evaluation did not cover a comparison of animated transition to its static counterparts. But also for the evaluation done, the control group for size of 3 the group of participants above the age of 40 is very small. Therefor to manifest the theorem that the age was relevant for the significant result more older participants would be needed.

List of Figures

List of Tables

Bibliography

[BB99] Benjamin B Bederson and Angela Boltman. Does animation help users build mental maps of spatial information? In *Proceedings 1999 IEEE Symposium on Information Visualization (InfoVis' 99)*, pages 28–35. IEEE, 1999.

[BC11] Victoria Battista and Edmond Cheng. Motion charts: Telling stories with statistics. In *American Statistical Association Joint Statistical Meetings*, volume 4473. Citeseer, 2011.

[BCK05] Thomas Bladh, David A Carr, and Matjaz Kljun. The effect of animated transitions on user navigation in 3d tree-maps. In *Ninth International Conference on Information Visualisation (IV'05)*, pages 297–305. IEEE, 2005.

[BOH11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 data-driven documents. *IEEE transactions on visualization and computer graphics*, 17(12):2301–2309, 2011.

[CM86] William S Cleveland and Robert McGill. An experiment in graphical perception. *International Journal of Man-Machine Studies*, 25(5):491–500, 1986.

[DBJ+11] Pierre Dragicevic, Anastasia Bezerianos, Waqas Javed, Niklas Elmqvist, and Jean-Daniel Fekete. Temporal distortion for animated transitions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2009–2018. ACM, 2011.

[Fis10] Danyel Fisher. Animation for visualization: opportunities and drawbacks. *Ch*, 19:329–352, 2010.

[GZL+20] T Ge, Y Zhao, B Lee, D Ren, B Chen, and Y Wang. Canis: A high-level language for data-driven chart animations. In *Computer Graphics Forum*, volume 39, pages 607–617. Wiley Online Library, 2020.

[HET+20] Seok-Hee Hong, Peter Eades, Marnijati Torkel, Weidong Huang, and Cristina Cifuentes. Dynamic graph map animation. In *2020 IEEE Pacific Visualization Symposium (PacificVis)*, pages 1–6. IEEE, 2020.

[HKA09] Jeffrey Heer, Nicholas Kong, and Maneesh Agrawala. Sizing the horizon: the effects of chart size and layering on the graphical perception of time series visualizations. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1303–1312. ACM, 2009.

[HR07] Jeffrey Heer and George Robertson. Animated transitions in statistical data graphics. *IEEE transactions on visualization and computer graphics*, 13(6):1240–1247, 2007.

[Lea] Learnopengl - camera. `https://learnopengl.com/Getting-started/Camera`. (Accessed on 09/17/2019).

[MWTI19] Kazuyo Mizuno, Hsiang-Yun Wu, Shigeo Takahashi, and Takeo Igarashi. Optimizing stepwise animation in dynamic set diagrams. In *Computer Graphics Forum*, volume 38, pages 13–24. Wiley Online Library, 2019.

[RFF+08] George Robertson, Roland Fernandez, Danyel Fisher, Bongshin Lee, and John Stasko. Effectiveness of animation in trend visualization. *IEEE transactions on visualization and computer graphics*, 14(6):1325–1332, 2008.

[RM15] Puripant Ruchikachorn and Klaus Mueller. Learning visualizations by analogy: Promoting visual literacy through visualization morphing. *IEEE transactions on visualization and computer graphics*, 21(9):1028–1044, 2015.

[RTh] R: The r project for statistical computing. `https://www.r-project.org/`. (Accessed on 01/04/2020).

[TLLS20] J Thompson, Z Liu, W Li, and J Stasko. Understanding the design space and authoring paradigms for animated data graphics. In *Computer Graphics Forum*, volume 39, pages 207–218. Wiley Online Library, 2020.

[TMB02] Barbara Tversky, Julie Bauer Morrison, and Mireille Betrancourt. Animation: can it facilitate? *International journal of human-computer studies*, 57(4):247–262, 2002.

[Tre19] Treemap vs bar chart - the end of treemap, May 2019.

[YFDH01] Ka-Ping Yee, Danyel Fisher, Rachna Dhamija, and Marti Hearst. Animated exploration of graphs with radial layout. In *Proc. IEEE InfoVis 2001*, pages 43–50, 2001.