FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Web Implementation of Radial Sets using D3.js

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Business Informatics

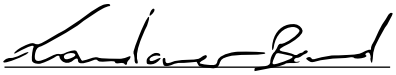
eingereicht von

Bernd Landauer
Matrikelnummer 0825716

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Mag. Dr. Silvia Miksch
Mitwirkung: Bilal Alsallakh, M.Sc.

Wien, 25.07.2013



(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Web Implementation of Radial Sets using D3.js

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Business Informatics

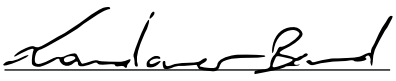
by

Bernd Landauer
Registration Number 0825716

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao.Univ.Prof. Mag. Dr. Silvia Miksch
Assistance: Bilal Alsallakh, M.Sc.

Vienna, 25.07.2013



(Signature of Author)

(Signature of Advisor)

Abstract

Radial Sets is a new technique for visualizing large overlapping sets in a more scalable way then classical methods such as Euler diagrams. Currently available implementation Radial Sets has some issues with accessibility as their is no version that the user can easily run within the browser without installing additional plugins. This impacts the accessibility as these plugins impose difficulties or security issues for many potential users who want to test Radial Sets. With the emerging technologies available in modern web browsers, it is possible to create a functional and interactive implementation of Radial Sets. The web-based version of Radial Sets make the visualization available for a larger number of users and contribute to its acceptance. This work presents a web-based implementation of Radial Sets. The created implementation is available and can be extended further to implement additional features of Radial Sets. The web implementation can be built with different software libraries. After comparing the libraries, I selected the D3 visualization toolkit as the best suited one will be used for the implementation. I elaborate on the implementation details to show how the different views and interactions can be supported using D3 and JavaScript functionalities. The results show that the web implementation can handle up to 50000 elements with acceptable interactive performance. This enables demonstrating Radial Sets using several datasets in the web browser which makes this technique highly accessible for a wide range of potential users.

Contents

CHAPTER 1

Introduction

Information visualization utilizes computer graphics and interaction to assist humans in solving problems [30]. To reach the masses, a visualization software must be as accessible as possible. Currently the web is the most accessible space for presenting information. Having web-based implementation of a visualization technique enables a wide range of users to run and interact with the visualization, with minimal technical overhead. In the last years the browsers and the web-technologies like HTML5 [htm], CSS3 [css] and JavaScript became more powerful. Now the browser can display visualizations, previously only that desktop applications could. With the recent availability of web-based visualization frameworks, a variety of vendors have been providing visualization-based solutions in their web applications. Also, several news agencies have started to offer their readers informative and interactive visualizations such as "New York Times", "The Guardian" and "Der Standard". These web-based visualizations do not require the users to install additional software, as they use technologies that are supported by all modern browsers.

1.1 Problem Description

One of the fundamental types of data is set-typed data, such data are commonly used to represent the memberships of a collection of elements in different sets. Euler diagrams are the most common representation for this type of data however, they are inherently limited in terms of the number of sets they can handle [27]. For this purpose, a new technique has been developed to handle a larger number of sets [27], as I will describe in Chapter 2. This technique, called Radial Sets, has currently no web-based implementation. Running the original Java-based research prototype requires the use of additional software. The need for Java or other software to run Radial Sets poses a problem, because it imposes a barrier to many potential users who might be interested in using Radial Sets. Current Java versions have many security issues, so many users could avoid the Java implementation [26]. The goal of this work is to create a web implementation of Radial Sets. With this implementation, the accessibility of Radial Sets will

improve. Another important advantage is that the users can upload their data easily via their web browsers. This is vital because the Java web client typically does not have access to the file system, which makes it very cumbersome for users to visualize their own data using the Java-version Radial Sets. However, the web-based version of the visualization might impose a limit on the number of elements that can be handled and how much this number impacts the browser performance especially while interacting with the visualization.

CHAPTER 2

Description of Radial Sets

Radial Sets is a new technique for visualizing large overlapping sets. It uses frequency-based representations to depict how the elements belong to the sets, and how the sets overlap. More details about Radial Sets can be found in [27]. Radial Sets employ multiple and coordinated views to explore the set memberships to the elements at different levels of detail. Figure 2.1 shows a screenshot of the desktop version of Radial Sets. The web version implemented in this work is also composed of four views.

- **Sets by Cardinality** (Figure 2.1b) showing the sets by their cardinality.
- **Elements of Degree** (Figure 2.1c) a breakdown of the set elements by their degrees.
- **Main View** (Figure 2.1a) where the Radial Sets will be displayed.
- **Detail View** (Figure 2.1d) where the selected elements will be displayed.

The current selected entries will be highlighted in red. For more details about these view, refer to the article on Radial Sets by Alsallakh et al. [27]. To visualize the overlapping sets, three kinds of visual elements are being used (Figure 2.2):

- the sector represent the sets
- histograms inside each sector represent the elements, they are grouped by their connection to other sets
- links between the sectors to represent overlaps between the sets.

The web-version of Radial Sets implemented in this work intends mainly to showcase the main features of the technique. These features encompass:

- The Radial Sets main view with all sectors, histogram and links
- The four views described above

Figure 2.1: Original Radial Sets application in Java. (a) view where Radial Sets will be displayed, (b) the sets by their cardinality, (c) a breakdown of the set elements by their degrees, (d) details about selected elements, (e) an additional detail view that I will not cover in the web-implementation. The red highlighted areas represent the selected elements.

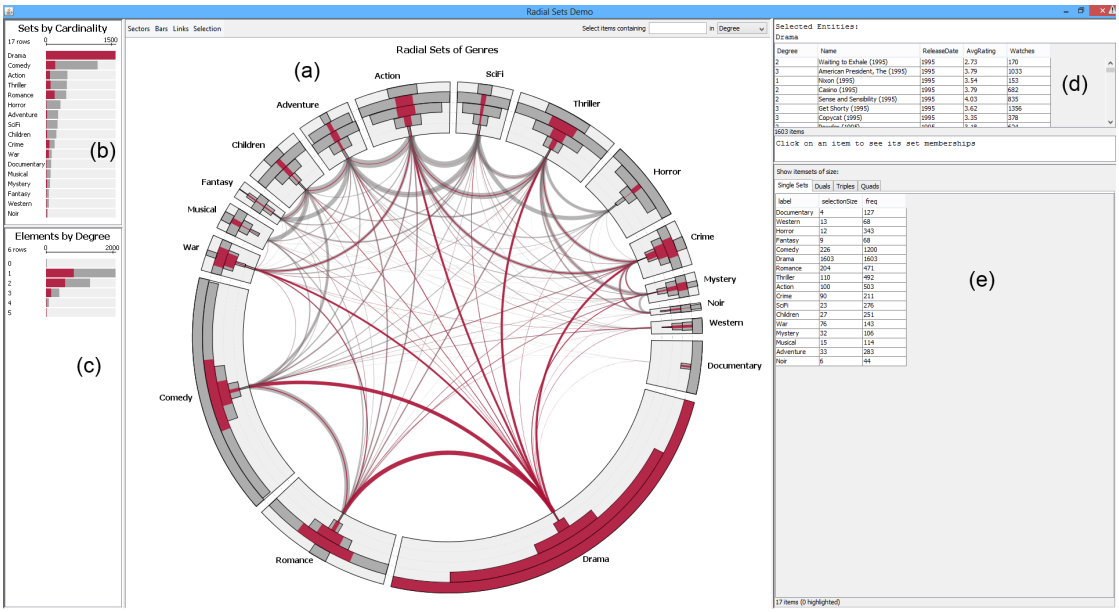
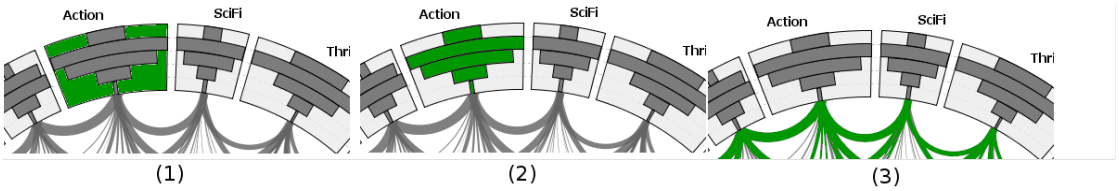


Figure 2.2: Description of Sectors



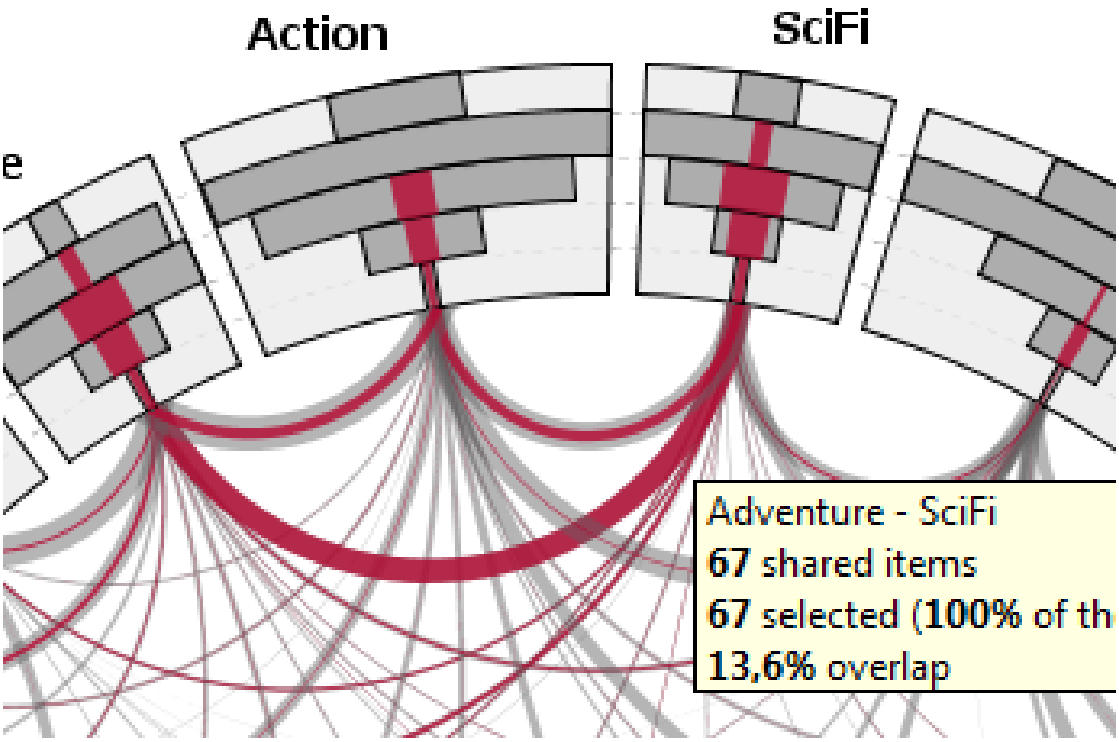
- The interplay between the four views and by highlighting the currently selected elements in each view.
- Selection mechanisms that let the user perform set operations such as union, intersection and subtraction(difference) between the sets.
- Information tooltips that are shown when hovering over sectors, histograms, links and selected elements (Figure 2.3).

The following features will not be include in the web-based implementation:

- Hyperlinks that would link to additional information.
- Bubbles instead of arcs between sectors.

- Detail view (Figure 2.1e) that give, additional information of the currently selected elements and how they overlap.

Figure 2.3: Detailed Sector in the main view with active selection



CHAPTER 3

State-of-the Art

I identified the following criteria for selecting a web-based library to support the visualization of Radial Sets as described in Section 4.1

- The library should be regularly updated and should be supported by an active community.
- The elements created by the library must be easy to manipulate and integrate with other HTML elements.
- Also the library must be able to handle large amount of data.
- The ability to create custom graphic elements without restriction to a predefined set of charts provided by the library.

To meet the search criteria, the libraries should be regularly updated and should be supported by an active community. The elements created by the library must be easy to manipulate and integrate with other HTML elements. Also the library must be able to handle large amount of data. And the last important criterion is the ability to create custom graphic elements without restriction to a predefined set of charts provided by the library. From my working experience and research on web applications, I considered the following web-based software libraries. They can be classified into JavaScript-based libraries and libraries- based on other technologies.

3.1 JavaScript Visualization Libraries

JavaScript is the de-facto standard programming language for developing advanced client-side applications that run within the web browser. In the following section, I list five candidate libraries I considered for developing the web-based version of Radial Sets along with relevant features and information about them:

- **D3.js** [Bostock]
  D3.js is an open-source library for manipulating documents based on data, by using

HTML, SVG and CSS. D3 makes it possible to bind large amount of data to a complex visualization.
Latest release v3.1.5: 7 April 2013

- **RaphaelJS** [rap]
  RaphaelJS is an open-source visualization library that simplifies creating and manipulating vector graphics (SVG).
  Latest release of v2.0: 1 October 2011

- **Highcharts** [hig]
  A library with a huge amount of predefined interactive charts and other visualizations. The charts are easy to use and have an aesthetic look. Both commercial and free licenses available.
  Latest release date: 22 March 2013

- **Processing.js** [Resig et al.]
  Processing.js is the sister project of the popular Processing [pro] visual programming language, designed for the web. This open-source library allows the developer to create full interactive animations.
  Latest release date: 9 August 2012

- **GoogleCharts** [goo]
  GoogleCharts is a JavaScript chart library from Google. The library also offers a wide range of different charts to select from. The created charts are displayed in HTML5 and SVG. The library is available under a free license.
  Latest release date: 24 September 2012

3.2 Other Web-based Visualization Libraries

The following technologies are outdated, will not be supported in near future, have problems with browsers or are not available for some devices. All those libraries use additional software to function and are therefor not suitable for the web-based Radial Sets application.

- **Flare (Flash)** [3]
  Flare is a software library based on Flash [6]. Flash is a program used to create animation, interactive graphics or play videos in the browser. The library only works if the Flash plugin is installed on the user's browser. Flare has been discontinued and is no longer updated.
  Latest release date: 24 January 2009

- **Dynamic Data Display (Silverlight)** [ddd]
  Dynamic Data Display is an interactive visualization library for dynamic data within Silverlight [18] applications. Like Flash, Silverlight must be installed on the user's device. Although Silverlight works on all major operating systems, Microsoft the developer of Silverlight dropped Silverlight and there will not be a support after 12 Oct. 2021 [17].
  Latest release date: 17 October 2011

3.3 Web-based Implementation of Similar Techniques

There are two other methods for visualizing overlapping sets, most notably Venn and Euler diagrams. For these methods web-based implementations exist, both using D3 JavaScript library and SVG [20; 21] (Figure 3.1) and using *canvas* elements [19] (Figure 3.2). This Technique is easy to understand but the scalability is limited. Performance problems should not be an issue in this visualization, because it represents less graphical elements then Radial Sets. Generally the Euler diagram within the canvas element is faster then the one with SVG. But only the Euler diagram created with SVG elements can be exported and printed without additional programming effort.

Figure 3.1: Venn diagram of movies with D3.js

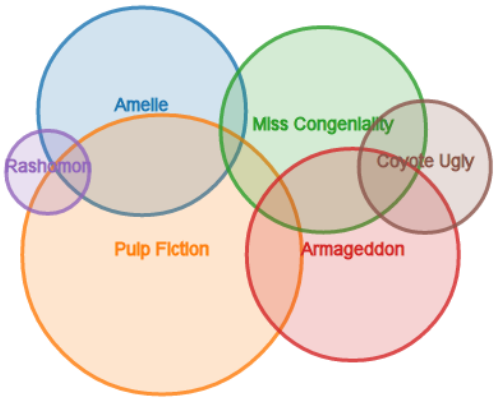
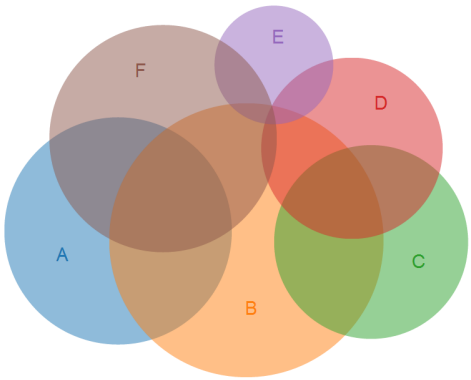


Figure 3.2: Another Venn diagram with D3.js



3.4 Comparison of Libraries

In this section I compare the technologies listed in Section 4.1 based on the features required for implementing the web-based version of Radial Sets.

- **Highcharts** [hig]
  Benefits: Offers a variety of predefined charts that are easy to use. Currently the most beautiful chart library I could find, best suitable when you want to create many different charts.
  Disadvantages: Limited to the set of charts, given by the library creators with some customizations.

- **GoogleCharts** [goo]
  The Google chart library has most of the benefits and disadvantages of the Highcharts library, but focuses more on the customizability of the library. This enables creating custom charts other than the built-in ones.
  Benefits: A variety of predefined charts as well as a set of methods to customize these charts. The ability to create custom charts.
  Disadvantages: Although the developer can create custom charts, it's more complicated in comparison to other libraries that have this feature too.

- **Processing.js** [Resig et al.]
  Unlike the other libraries mentioned so far Processing.js does not generate vector graphics or HTML elements, instead it uses the *canvas* element to display the graphical elements via JavaScript. Also the developer can create powerful animation in 2D or 3D.
  Benefits: The ability create complex interactive animated graphics both in 2D and in 3D.
  Disadvantages: Very complex and not needed for the creation of Radial Sets.

- **RaphaelJS** [rap]
  This library employs SVG graphics to display data graphically in the browser.
  Benefits: Easy to understand and to create charts.
  Disadvantages: The developer is limited to a set of methods that are implemented within the library. Also the library has not been updated for a while, that could be an indication for a shrinking contributor base. The developer must create each part of the graphic one by one, even if all the parts are from the same type, for example a circle.

- **D3.js** [Bostock]
  D3 is the most advanced library to create SVG graphics. No abstraction layer between the document object model (DOM) and the developer methods.
  Benefits: D3.js is the most elaborate and most flexible SVG library. Also currently, it has the most active contributor community. D3 allows direct access to the created SVG graphical elements using a small set of methods, the developer can access all SVG attributes without restrictions. Moreover, D3 provides a selector engine like jQuery which enables defining flexible query to select specific data elements or graphical elements.
  Disadvantages: D3 is rather a complex library having a large number of methods and is relatively harder to learn than RahpaelJS.

3.5 State of the Art Conclusion

For this work, I will use the D3 library for the following reasons:

- It is the newest library and has being designed to follow the best practice and avoid the pitfalls of the other libraries.
- It has been widely adopted by web developers and is undergoing more active development compared to the other libraries.
- D3 is quite suited to create complex visualizations and is not only restricted to a limited set of build-in visualizations like in many other libraries. This is important because Radial Sets is a novel visualization technique whose visual design does not resemble common charts that are predefined in the other libraries.

Moreover, D3 is not just a library for creating diagrams or interactive visualizations but also, like with jQuery the developer can select and bind data to elements. Instead of having an abstraction layer between the DOM element and the programmatic commands, the developer has full access to all attributes of DOM/SVG elements. For a detailed comparison between D3 and other common visualization libraries, the reader can refer to the work by Bostock et al. [jsl; cha; 29].

CHAPTER 4

Result

In this chapter I will describe the result of my work. I will explain each view and how the views are created. Using D3 and HTML elements Figure 4.1 and Figure 4.2 depict the current state of the implemented web-based version of Radial Sets.

Figure 4.1: Radial Sets without selected elements

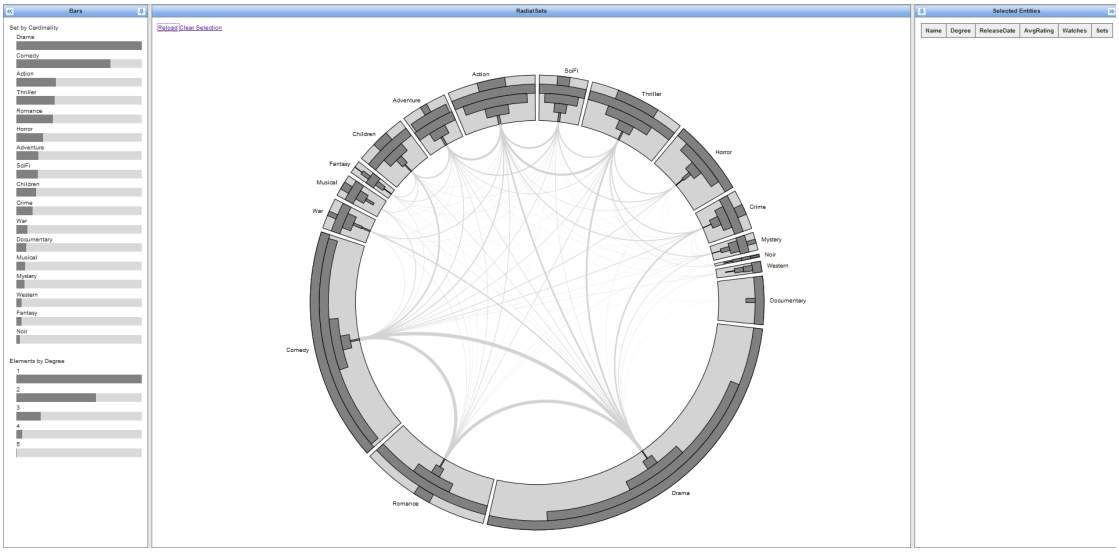
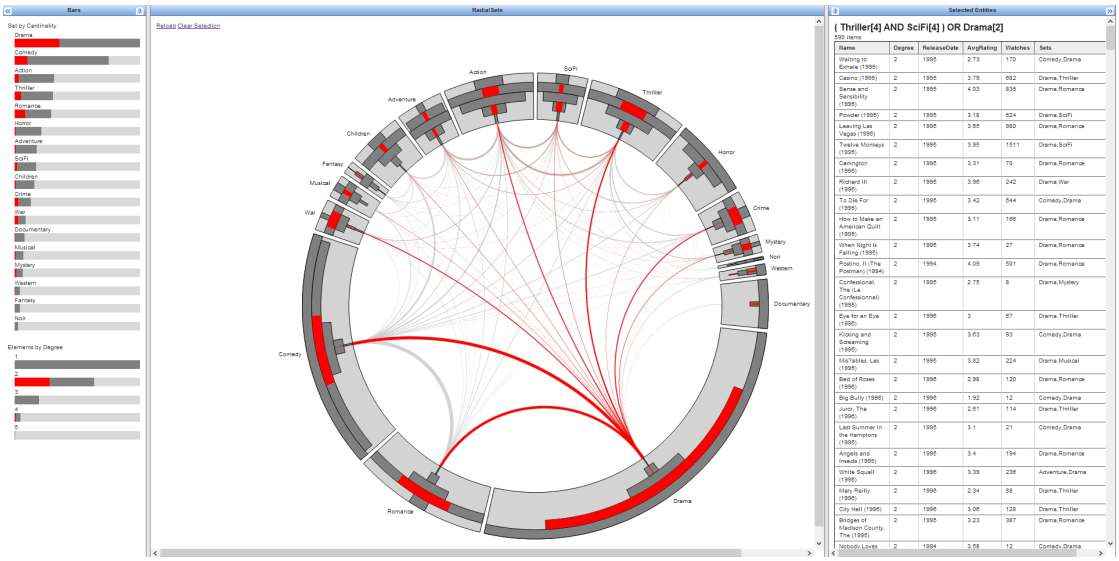


Figure 4.2: Radial Sets with selected elements



4.1 Bar Charts

The ”Sets of Cardinality” view is located at the upper left side of the user interface and is created using a list of *li* elements and CSS styles. To create these elements I group all entries by there sets. By clicking a bar all elements contained in the respective set are selected. The program highlights and updates each view to highlight the new selection. On hovering over a bar a tooltip is displayed for more information about the set (Figure 4.3).
The ”Elements by Degree” view is located at the bottom left side of the interface and is created in the same way as the ”Sets of Cardinality” (Figure 4.4). The difference is that the elements are grouped by their degrees instead.

Figure 4.3: Selected set with tooltip

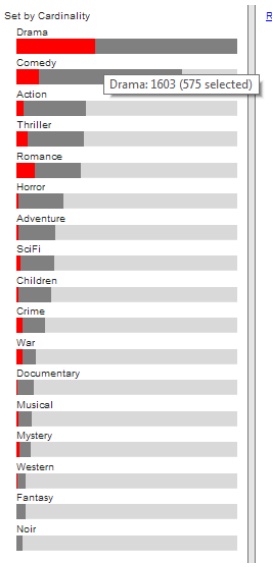
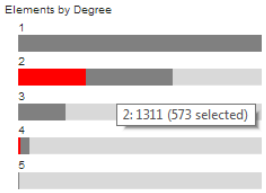


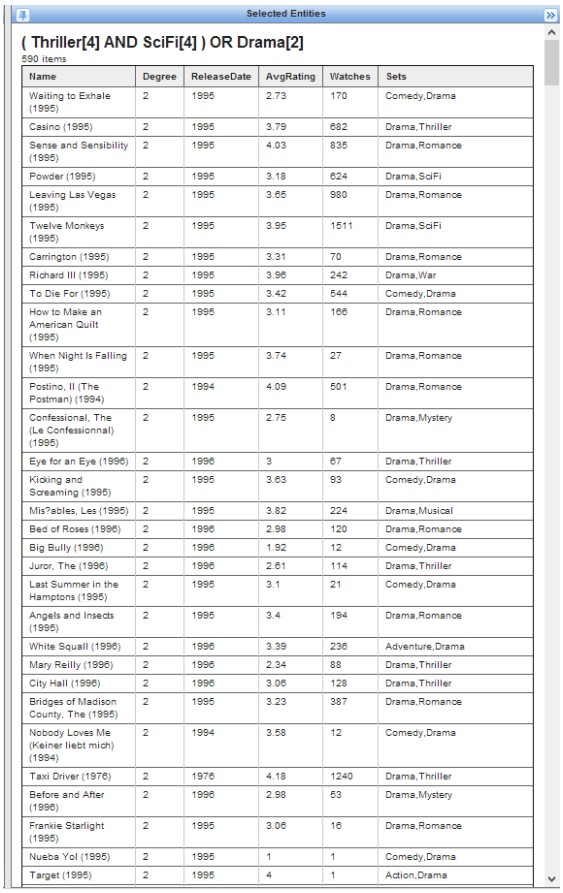
Figure 4.4: Selected elements by degree with tooltip



4.2 Selecting the Elements Interactivity

The ”Selected Entities” view is located on the right side of the user interface and is structured like a table. After reading the CSV file which contains the elements and their set memberships, the table is filled with all entries. These entries are hidden in the beginning. When the user make a selection all entries within selection are displayed in this table. At the top of the table a counter shows the number of entries selected. The view also contains detailed information about the current selection and the set operations performed to create this selection (Figure 4.5).

Figure 4.5: Selected Entries



Name	Degree	ReleaseDate	AvgRating	Watches	Sets
Waiting to Exhale (1995)	2	1995	2.73	170	Comedy,Drama
Casino (1995)	2	1995	3.79	682	Drama,Thriller
Sense and Sensibility (1995)	2	1995	4.03	835	Drama,Romance
Powder (1995)	2	1995	3.18	624	Drama,SciFi
Leaving Las Vegas (1995)	2	1995	3.65	980	Drama,Romance
Twelve Monkeys (1995)	2	1995	3.95	1511	Drama,SciFi
Carrington (1995)	2	1995	3.31	70	Drama,Romance
Richard III (1995)	2	1995	3.96	242	Drama,War
To Die For (1995)	2	1995	3.42	544	Comedy,Drama
How to Make an American Quilt (1995)	2	1995	3.11	166	Drama,Romance
When Night Is Falling (1995)	2	1995	3.74	27	Drama,Romance
Postino, Il (The Postman) (1994)	2	1994	4.09	501	Drama,Romance
Confessional, The (Le Confessionnal) (1995)	2	1995	2.75	8	Drama,Mystery
Eye for an Eye (1996)	2	1996	3	67	Drama,Thriller
Kicking and Screaming (1995)	2	1995	3.63	93	Comedy,Drama
Mis?ables, Les (1995)	2	1995	3.82	224	Drama,Musical
Bed of Roses (1996)	2	1996	2.98	120	Drama,Romance
Big Bully (1996)	2	1996	1.92	12	Comedy,Drama
Juror, The (1996)	2	1996	2.61	114	Drama,Thriller
Last Summer in the Hamptons (1995)	2	1995	3.1	21	Comedy,Drama
Angels and Insects (1995)	2	1995	3.4	194	Drama,Romance
White Squall (1996)	2	1996	3.39	236	Adventure,Drama
Mary Reilly (1996)	2	1996	2.34	88	Drama,Thriller
City Hall (1996)	2	1996	3.06	128	Drama,Thriller
Bridges of Madison County, The (1995)	2	1995	3.23	387	Drama,Romance
Nobody Loves Me (Keiner liebt mich) (1994)	2	1994	3.58	12	Comedy,Drama
Taxi Driver (1976)	2	1976	4.18	1240	Drama,Thriller
Before and After (1996)	2	1996	2.98	53	Drama,Mystery
Frankie Starlight (1995)	2	1995	3.06	16	Drama,Romance
Nueba Yol (1995)	2	1995	1	1	Comedy,Drama
Target (1995)	2	1995	4	1	Action,Drama

4.3 Logging View

When starting the Radial Sets application the developer can provide an option that displays a logging view at the bottom of the browsers window. This view intends to help the developer in debugging the application and in investigating the run-time performance in different browsers and using multiple datasets. It logs time since the application started and time between important actions. Also when the user makes a selection the log displays the selected entries. In addition each messages being logged will appear in the browser console or debugging tool (Figures 4.6 and 4.7).

Figure 4.6: Radial Sets with enabled logging

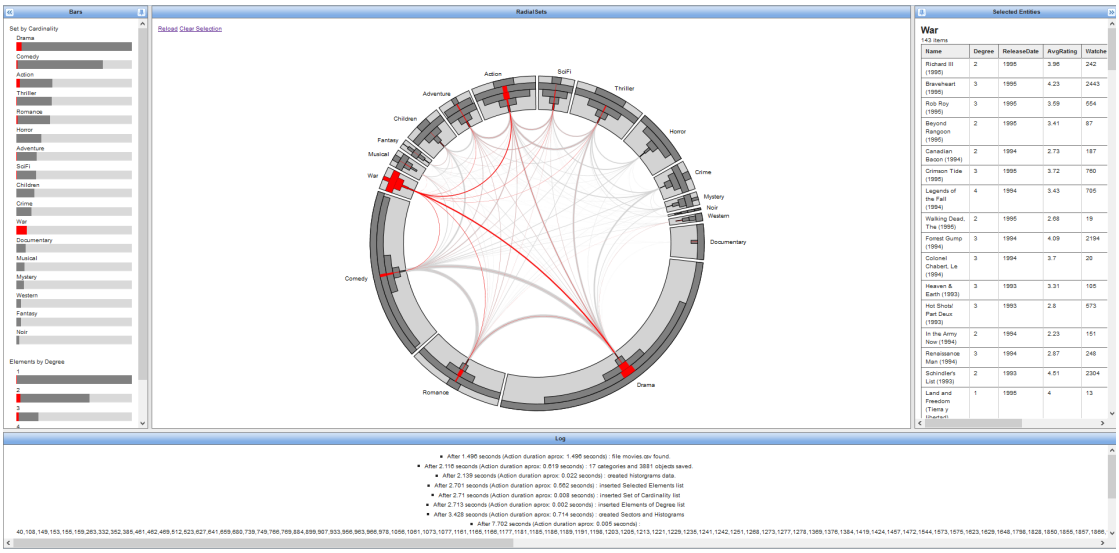
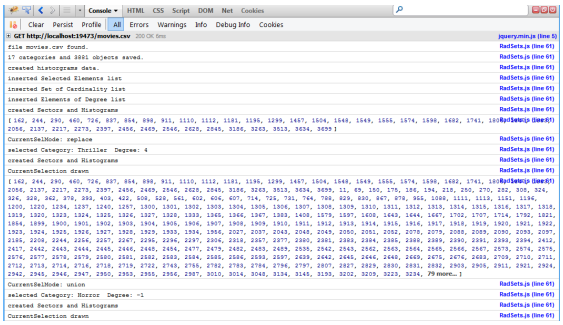


Figure 4.7: Radial Sets console logs in Firebug



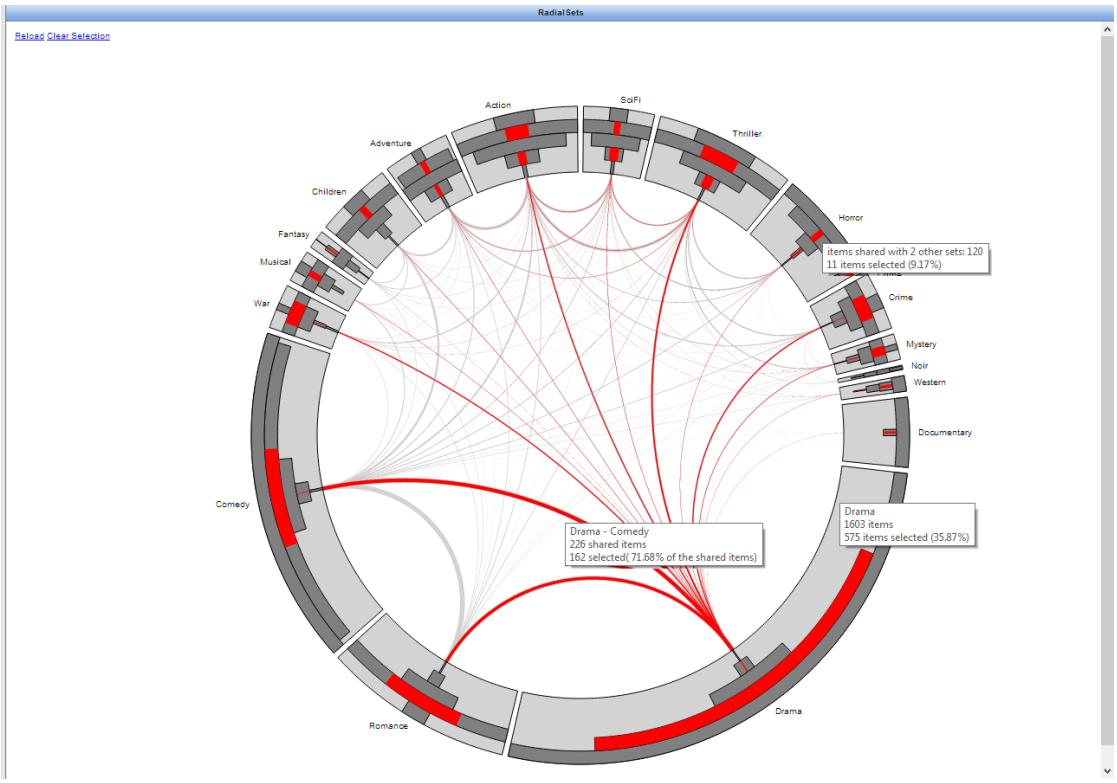
4.4 Radial Sets View

The final view is located in the center and represents the Radial Sets. On the top left of the view three commands are displayed as links.

- **Reload** to reload and resize the whole application for example if the user changes the window of the browser (note the CSV file will not be loaded again).
- **Clear Selection** will clear the current selected entries and perform the reload action.
- **Open Help** displays a dialog upon click that will tell the user how to create selection.

In this view the user can select different sectors, histograms and links to generate a selection. Tooltips will appear with additional information when the mouse-pointer enters an element (Figure 4.8).

Figure 4.8: Selected Entities in the main view with some tooltips



4.5 Modeling Radial Sets in D3.js

The Radial Sets are visualized using a SVG elements [svg]. To generate the circle, the application uses the D3 layout for a donut-chart [don] and modifies it to create spaces between the sectors. Each sector is created as a group element *g* and within each sector multiple *path* elements are created for the histograms. The selected entries are drawn over these histograms and are represented also as *path* elements (Figure 4.10).
To connect the sectors the application creates *path* elements, which are located in their own group called “ConnectionArcs”, (Figure 4.9).

Figure 4.9: Radial Sets SVG connection arcs elements in an HTML inspector tool

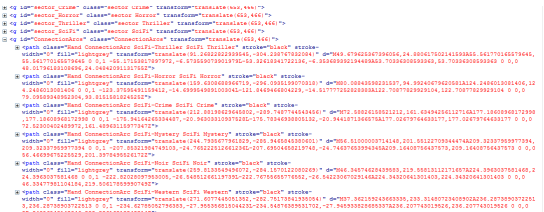


Figure 4.10: Radial Sets SVG sector and histogram elements in an HTML Inspector Tool

4.6 Performance & Browser Support

In this section I report some problems I encountered and also highlight the current limitations for the web-based Radial Sets.
Available test files:

- demo file with 3881 movies and 17 categories.
- IMDb movies file with 500,000 movies.

The performance tests where separated in the following steps:

- reading the file.
- converting the file in the object structure expected by the views.
- creating the HTML elements form the converted information.
- drawing the Radial Sets main view.
- selection of the largest set available and compute the selected entries and redrawing the Radial Sets.

The result of the performance tests indicates that not all browsers perform as expected. Mozilla Firefox has problems with handling the calculation-intensive selection operations. Chrome performs very good especially as the number of entries increases. It only shows little performance breakdowns. Internet Explorer 10 surprises, in overall performance and outperforms Firefox. The only issues encountered by Internet Explorer is the performance problem on manipulating the selected entities *table* (Figures 4.11,4.12 and 4.13).

Figure 4.11: Performance Overall

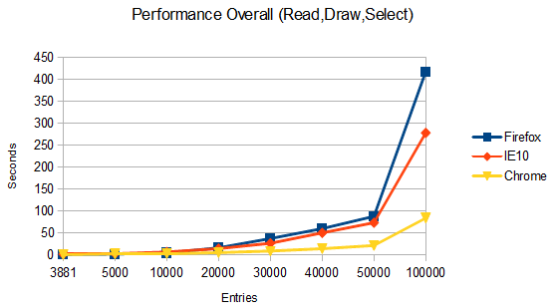


Figure 4.12: Performance only on reading data and drawing

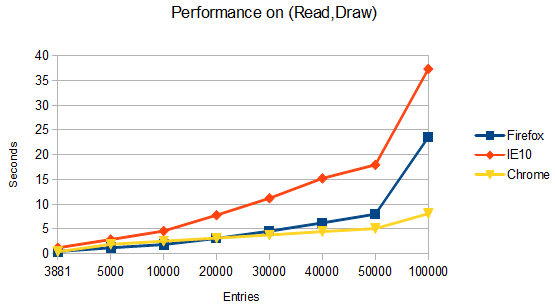
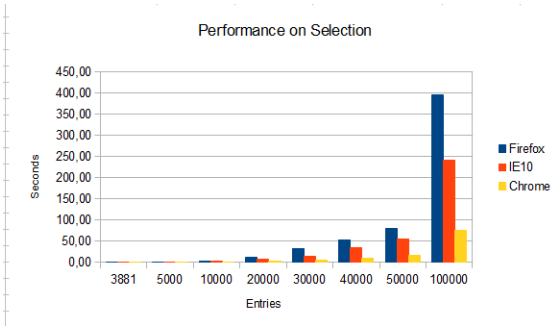


Figure 4.13: Performance when selecting the largest set



4.7 Open Issues with the current Implementation

The number of entries is currently the biggest problem with the web-based version of Radial Sets. Starting with 50.000 entries the application encounters performance problems in creating the Radial Sets and in performing a selection operation. During the development I encountered some minor drawing errors upon drawing the “ConnectionArcs” between sectors, when the number of entries exceeds 10.000. These drawing errors were due to an incorrect sort order of the sectors and have been fixed accordingly.
When the Radial Sets application is started with less then 10.000 entries no performance problems or errors were encountered.

CHAPTER 5

Implementation Details

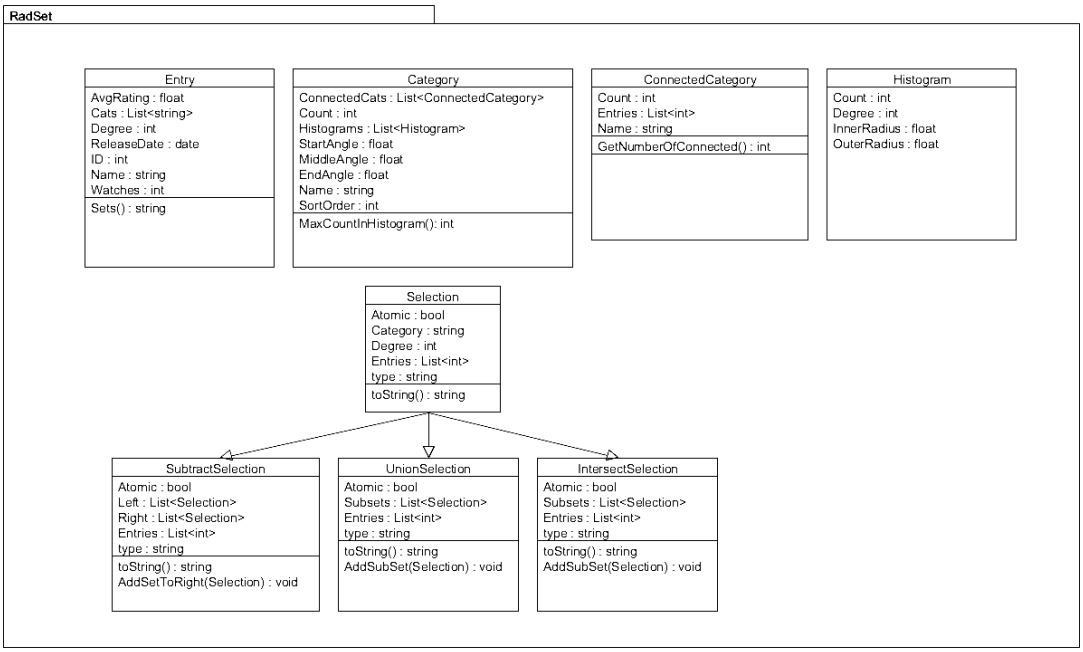
In this chapter I explain the detailed structure of the JavaScript application that implements Radial Sets. The JavaScript Module Pattern [jsM] is used in the application to separate the code in multiple files and make the code easier to maintain. The application is separated in four different files modules, which all together form the class "RadSet":

- **RadSets** Contains the utility functions and properties of the Radial Sets.
- **RadSetsConverter** This module handles the conversion of the CSV file into the internal format used in the application. It also handles the creation of the elements and the category list.
- **RadSetsDraw** This module handles the drawing of Radial Sets in each view.
- **RadSetsSelector** This module implements the selection mechanisms and the classes related to them.

Different classes are being used in the application. Figure 5.1 shows a UML diagram with all defined classes within the Radial Sets application and their methods. Radial Sets are created via the largest method in the whole application "RadSet.Draw".
Most colors are defined in the CSS file "RadSet.css". All the other style related properties are parsed and added to the CSS. For example the colors of the histograms can be changed by modifying the property "RadSet.HistoColors". This property a list of hexadecimal values representing the colors and ordered by degree.
The logging is achieved by the method "RadSet.log" and creates log entries in the browser console log and in the logging view of the application. For the current application the developer needs to have four *div* elements and needs to execute "RadSet.Init(options)" method.

Via the "options" object the developer can modify the behavior of the application by changing the Properties in this object. The possible options are :

Figure 5.1: UML of RadSet Classes



- **CategoryOrder** *Default: ["Action", "Adventure", "Children", "Fantasy", "Musical", "War", "Comedy", "Romance", "Drama", "Documentary", "Western", "Noir", "Mystery", "Crime", "Horror", "Thriller", "SciFi"]*, Defines the Order of the Sets.
- **ConnectionArcColor** *Default: "lightgrey"*, Color of the connection arcs.
- **DivElementsByDegreeID** *Default: "elementsByDegree"*, ID from the table element where the sets by degree will be displayed.
- **DivSetOfCardinalityID** *Default: "setOfCardinality"*, ID of the div element where the sets by cardinality will be displayed.
- **EnableHover** *Default: false*, Indicates if the hover effect is enabled.
- **EntryLimit** *Default: 5000*, Maximum number of entries the application will read.
- **File** *Default: "movies.csv"*, Filename of the CSV file containing the data.
- **HighlightColor** *Default: "red"*, Color of the highlighted areas and selected elements.
- **HistoColor** *Default: "grey"*, Default histogram color when no other colors are defined in "HistoColors" option.
- **HistoColors** *Default: []*, List of custom histogram colors per degree.

- **HoverColor** *Default: "red"*, color of the hover effect, only when "EnableHover" is true.
- **IgnoreEntriesWithoutCategories** *Default: "true"*, Indicates if entries without sets are ignored.
- **ListOfNonCategories** *Default: ["ReleaseDate","AvgRating","Watches","Animation"]*, List of non-set values in CSV file, all names in this list will not be categorized as sets.
- **ListOfVisableProps** *Default: ["Name", "Degree", "ReleaseDate", "AvgRating", "Watches", "Sets"]*, List of properties displayed in the selected elements view, each name in this list can represent a property or a function in the element object.
- **LogListElementID** *Default: "log"*, ID for the log element - must be a *ul* element.
- **LogSelectionEntries** *Default: false*, Indicates if the selected elements for current selection will be written in the log.
- **SectorColor** *Default: "lightgrey"*, Color of the sectors.
- **SelectionAlignConnectionArc** *Default: "center"*, Indicates which alignment the selection will have in the connection arcs.
- **SelectionAlignHistogram** *Default: "center"*, Indicates which alignment the selection will have in the histograms.
- **TableSelectedEntitiesID** *Default: "tabSelEntities"*, ID for the table where the selected elements will be displayed.

Each of these properties also has a default value. As example for loading the "movies.csv" file and setting the entry limit to 5000 the method call would look like:

Listing 5.1: Init Function

```
RadSet.Init({File:''movies.csv'',EntryLimit:5000});
```

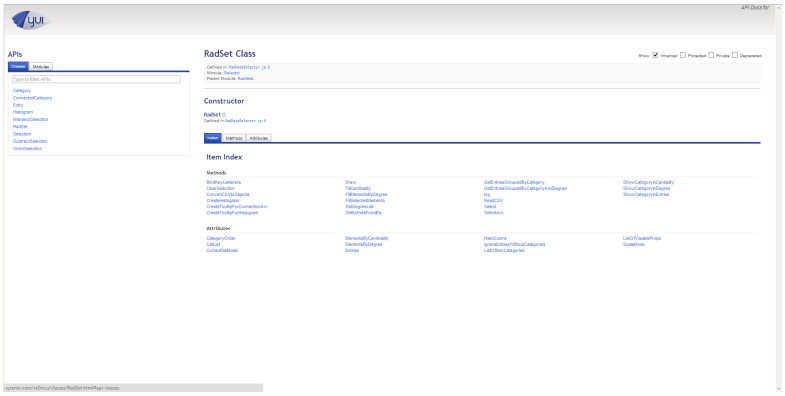
The initiation function contains the following lines of code:

Listing 5.2: RadSet.Init()

```
converts entries from the CSV file.
RadSet.ReadCSV(options.File);

// inserts tables entries from the created Entries list.
RadSet.FillSelectedElements(options.TableSelectedEntitiesID, RadSet.Entries);

// groups entries by cardinality and fills the div responsible for the sets of
cardinality.
RadSet.ElementsByCardinality =
RadSet.FillCardinality(options.DivSetOfCardinalityID, RadSet.CatList);

// here the entries will be grouped by degree and creates the bar charts for
``elements by degree``
RadSet.ElementsByDegree =
RadSet.FillElementsByDegree(options.DivElementsByDegreeID, RadSet.Entries);

// This method binds the Keyboard listeners so the selector engine could work
with keyboard shortcuts.
RadSet.BindKeyListeners();

// Some properties in RadSets are directly connected to the look of the Radial
Sets. This Method will parse all these properties to a style-tag and paste it into the HTML.
RadSet.CreateStyleTag();

// Here the application begin to draw the Radial Sets main view. Dynamically
computes the maximum possible size by the browser window. Creates sectors,
histograms and connection arcs.
RadSet.Draw();
```

Furthermore a Documentation is available, created by YuiDoc [yui]. The Documentation is located on *"http://radialsets.org"*.

Figure 5.2: Radial Sets Documentation with YuiDoc



5.1 External Libraries

The application uses multiple JavaScript libraries to improve the user experience and to implement core functionalities:

- **jQuery** [13] to perform queries over HTML elements and modify these elements.
- **jQuery-UI** [12] adds UI components to the application.
- **jQuery-Layout** [jqL] used to separate each view and adds the ability to resize and toggle these views.
- **jQuery Array Utilities** [jqA] allows JavaScript arrays to perform UNION, INTERSECT, EXCEPT.
- **D3.js** [Bostock] D3.js draws the Radial Sets main view and modify these SVG elements.

Currently the essential files for running the web-based version are:

- demo.html
- css/RadSet.css
- js/RadSets/RadSets.js
- js/RadSets/RadSetsConverter.js
- js/RadSets/RadSetsSelector.js
- js/RadSets/RadSetsDraw.js
- js/RadSet.js

Files for libraries are not included in this list.

CHAPTER 6

Conclusion and Future Work

In this work, I showed that it is possible to implement complex visualizations as a web application running on the client side. Thanks to the advanced possibilities offered by D3, both the novel visual metapher and the interactions of Radial Sets have been implemented in JavaScript. The main limitation of the implemented application is scalability. The performance degrades noticeably on browsers when the number of entries exceeds certain limits. The web-based version of Radial Sets can be improved further in several aspects. The responsivity of the application can be increased by implementing Web Workers [web], to perform intensive calculations for computing the selections in a background thread. Also, the rendering method "RadSet.Draw" needs to be refactored, into smaller methods such as creating histograms and connection arcs. This, combined with a good timeout between methods would also help to prevent "long running scripts" warnings.
Also, the currently unimplemented features need to be addressed such as hyperedges, bubbles, and the overlap analysis view. Furthermore, a more elaborate web-based user interface is needed to allow the user to upload her own datasets and to change several parameters and perform several commands such as color mapping from attributes, sector ordering scheme and filtering of certain elements.

Bibliography

[jsl] 13 javascript visualization libraries http://datavisualization.ch/tools/13-javascript-libraries-for-visualizations. Accessed 21-May-2013.

[cha] D3 experiments chartwheel http://anilomanwar.github.io/d3jsExperiments/ChartWheel.html. Accessed 21-May-2013.

[3] Diagrams with Flash http://flare.prefuse.org/. Accessed 21-May-2013.

[don] Donut chart with D3.js http://bl.ocks.org/mbostock/3887193. Accessed 9-June-2013.

[ddd] Dynamic data display http://research.microsoft.com/en-us/downloads/8812f483-1f5c-46b6-b9f4-22d1fdf7f2c1/default.aspx. Accessed 21-May-2013.

[6] Flash player http://get.adobe.com/de/flashplayer/. Accessed 21-May-2013.

[goo] Google charts https://developers.google.com/chart/. Accessed 13-April-2013.

[hig] Highcharts http://www.highcharts.com/. Accessed 13-April-2013.

[jsM] Javascript module pattern: In-depth http://www.adequatelygood.com/JavaScript-Module-Pattern-In-Depth.html. Accessed 9-June-2013.

[jqA] Jquery Array Utilities https://github.com/KristianAbrahamsen/jquery.arrayUtilities. Accessed 9-June-2013.

[jqL] Jquery UI layout http://layout.jquery-dev.net/. Accessed 9-June-2013.

[12] Jquery UI http://jqueryui.com/. Accessed 13-April-2013.

[13] Jquery http://jquery.com/. Accessed 13-April-2013.

[web] Mozilla description of web workers https://developer.mozilla.org/en-US/docs/Web/Guide/Performance/Using_web_workers. Accessed 9-June-2013.

[pro] Processing.js http://www.processing.org/. Accessed 22-May-2013.

[rap] RaphaelJS http://www.raphaeljs.com/. Accessed 13-April-2013.

[17] Silverlight lifecycle http://support.microsoft.com/lifecycle/?LN=en-us&c2=12905. Accessed 21-May-2013.

[18] Silverlight http://www.microsoft.com/silverlight/. Accessed 21-May-2013.

[19] Venn diagrams with Canvas http://www.canvasxpress.org/venn.html. Accessed 21-May-2013.

[20] Venn diagrams with D3.js http://stanford.edu/~helenlu/cs448b/example2.html. Accessed 21-May-2013.

[21] Venn diagrams with D3.js http://www.benfrederickson.com/2013/05/09/venn-diagrams-with-d3.js.html. Accessed 21-May-2013.

[css] W3C CSS http://www.w3.org/TR/css-2010/. Accessed 2-June-2013.

[htm] W3C HTML 5.1 http://www.w3.org/html/wg/drafts/html/master/. Accessed 2-June-2013.

[svg] W3C SVG http://www.w3.org/Graphics/SVG/. Accessed 4-June-2013.

[yui] Yuidoc http://yui.github.io/yuidoc/. Accessed 9-June-2013.

[26] (2013). Java security issues. http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/. Accessed 13-April-2013.

[27] Alsallakh, B., Aigner, W., Miksch, S., and Hauser, H. (2013). Radial Sets: Interactive visual analysis of large overlapping sets. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Information Visualization 2013)*, 19(12):to appear.

[Bostock] Bostock, M. D3 http://d3js.org/. Accessed 13-April-2013.

[29] Bostock, M., Ogievetsky, V., and Heer, J. (2011). D^3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309.

[30] Purchase, H. C., Andrienko, N., Jankun-Kelly, T. J., and Ward, M. (2008). Theoretical foundations of information visualization. *In information Visualization: Human-Centered Issues and Perspectives*.

[Resig et al.] Resig, J., Fry, B., and Reas, C. Processingjs http://processingjs.org/. Accessed 13-April-2013.