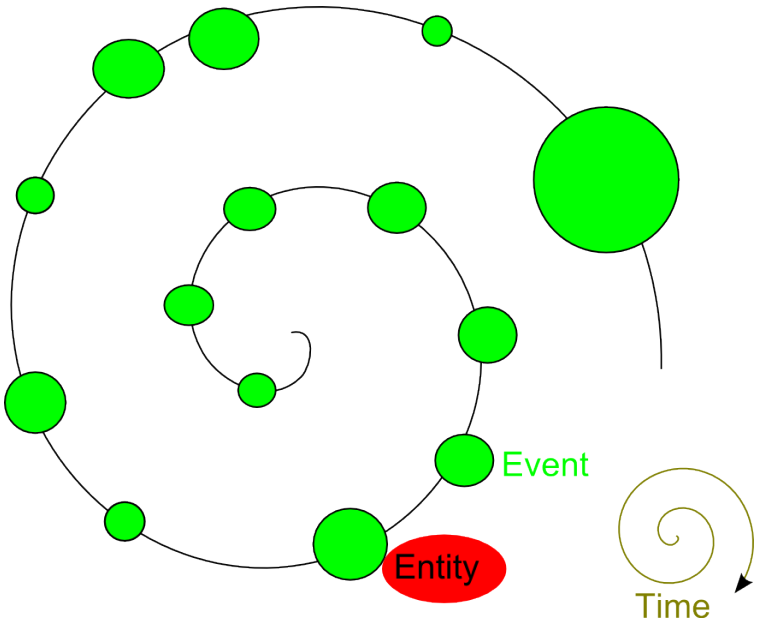
CHALMERS



Visualisation of time-based Entity-Relationship graph structures: the Spiral Entity-Relationship diagram

Master of Science Thesis in the Master Degree Programme Interaction Design

ALEJANDRO VALENZUELA ROCA

Department of Applied Information Technology
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden, 2013
Report No. 2013:140
ISSN: 1651-4769

Visualisation of time-based Entity-Relationship graph structures: the Spiral Entity-Relationship diagram
ALEJANDRO VALENZUELA ROCA

Examiner: STAFFAN BJÖRK

Report No. 2013:140
Chalmers University of Technology
Department of Applied Information Technology
SE-412 96 Göteborg
Sweden
Telephone +46 (0)31-772 1000

Cover:
The Spiral Diagram proposal, which is explained in section 7.

Department of Applied Information Technology
Göteborg, Sweden, November 2013

Acknowledgements

I would like to thank my parents, who are always there to help me out. My friends from both sides of the world, for their friendship and their support. To my examiner for his infinite patience and all the administrative help, and finally to the people of Sweden, for their generosity.

Abstract

An interactive system called the “Spiral Entity-Relationship Diagram” is proposed to visualise data structures common in query results from some online services (Recorded Future, Twitter). The proposed diagram is designed so that it can accommodate many query results and adding more results on-the-fly is possible. Query results can be distinguished in relevance by their colour and the user is able to manipulate the diagram by panning and zooming, as well as sorting and filtering results by their specific characteristics.

Keywords: Spiral, Information Visualisation, Entity-Relationship, Recorded Future, Twitter, Processing

Table of Contents

1 Introduction

On the Internet, information is being produced at an accelerating rate. For example, each second, Twitter, a popular social media network, receives an average of 5,700 short messages called tweets (Krikorian, 2013). Even though the average user will only receive a very small fraction of those tweets, the Twitter network and as a consequence their user's timelines can become very busy, especially during important events. Similarly, the number of news articles aggregated by Recorded Future (an online news articles analysis system) each second is enormous, considering it tracks many of the most popular news sites around the world in several languages.

While these are by no means the only information sources on the Internet, they are relevant to a very great number of people: about 200 million users in Twitter are active each month (Wickre, 2013) and news articles are important to almost everyone.

For many people it is important to make as much use as possible of this information; therefore the existence of companies such as Recorded Future, as well as universities and independent researchers, who collect, analyse and visualise this information.

This project is an attempt to break down a certain type of this generated content (namely Tweets and articles gathered by Recorded Future) into the entity-relationship model, in order to represent it in a way that allows the user to quickly find the information she is looking for, or if she is not looking for anything in particular, to help her spot something that can be of interest.

Formally speaking, the research question is:

"How can the Recorded Future entity-relationship data be represented conveniently?"

Together with Recorded Future, I agreed to propose ideas and create a prototype that visualises this kind of data. As a limitation for the project, due to the time available for development, it was decided that the prototype should be of exploratory nature. This means that the prototype I create will not necessarily be ready to be used as Recorded Future's own interface, and I will not be concerned with details such as the data security, commercial viability or performance of the application.

In order to do this, I will define three fundamental concepts: entity, relationship and event, in the context of this work, then, after a literature search, consider existing methodologies and solutions and propose several visualisations whose advantages and disadvantages are here evaluated. From the best visualisation, I will propose a solution that I consider accomplishes the goal of displaying the information for the user in the most convenient way possible: An interactive graph which displays items from the aforementioned sources, attached to a spiral shape.

The proposed solution will then be implemented iteratively in various prototypes and tested informally by a small group of users to gather feedback.

2 Background

2.1 Information overload

According to (Mulder et al, 2006) , "information overload is the feeling of stress when information load goes beyond processing capacity". It is a subjective feeling in the sense that people have different capacities for coping with information load, but its effects are tangible: it causes disorientation and impairs decision making, thereby causing further stress and discomfort.

Common causes of information overload can be an excess of data, a high level of fragmentation, irrelevant information including e-mails, and social or project pressure.

"Knowledge workers" are people who primarily work with gathering, storing, transforming and making decisions based on information. Typically, the result of their work appears in the form of reports, designs and advice - which means that mostly mental and not manual work is involved; for these reasons, it is expected that knowledge workers experience information overload very often.
This expectation is confirmed by the study: Although the study claims that no “hard” conclusions can be made, it found out that most of the knowledge workers interviewed reported experiencing information overload more than once a week.

Working with Recorded Future data or Twitter data usually involves working with many data items simultaneously and taking decisions based on the information gathered. This coincides with the description for what a knowledge worker would do, but also means that the possibility of experiencing information overload is very likely when working with data coming from either of these two services.
One of the aims of the project is to reduce the possibility of experiencing information overload while working with these services.

2.2 What is Recorded Future?

Recorded Future is a company based in Gothenburg, Sweden, that collects and processes news, blogs, twitter and many other Internet-published material. It analyses these texts and stores their characteristics in a database, which is accessible to its clients via several interfaces.
Its main interface is shown in figure 1.

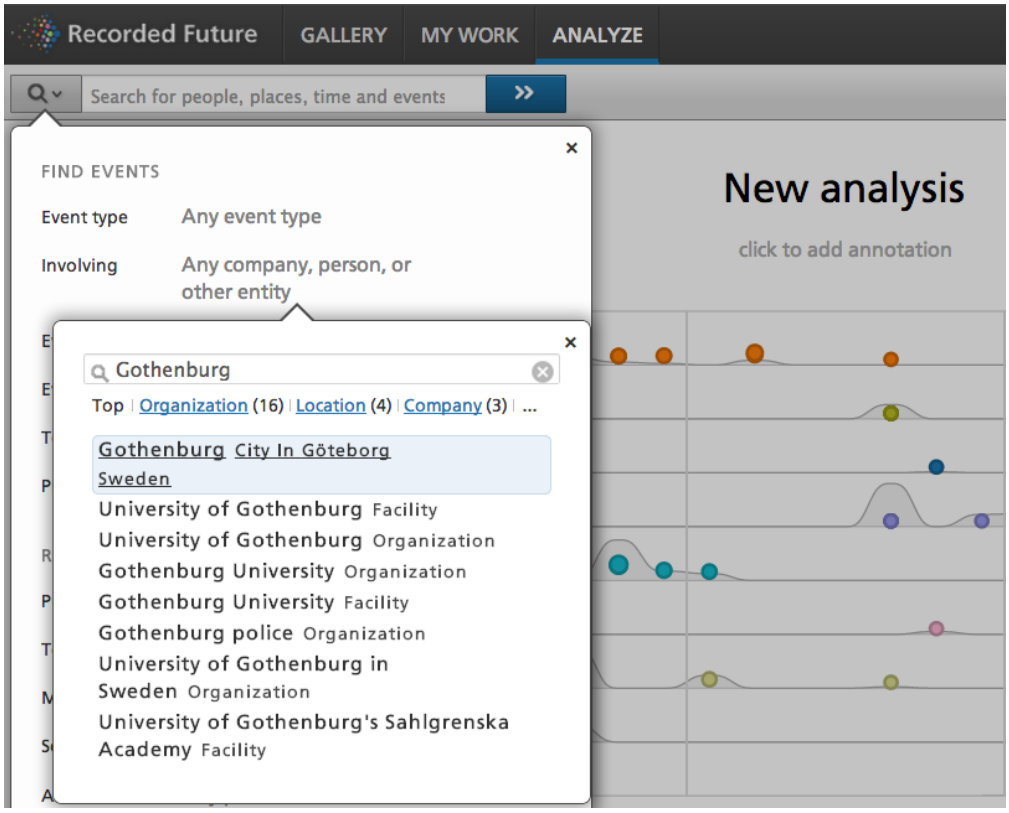


Figure 1: Recorded Future's main interface

The database can also be queried using Recorded Future's tools or the Recorded Future API[1]; the system understands among other constructs, queries involving determinate well-known people and companies, and time-past and time-future. For example, to query the database about what plans a specific company such as Google has for the next month. The results will feature all the articles available in that moment, in which Google has stated publicly what they intend to do in the month after the query was placed, as well as the speculation by third parties on what the company will be doing in the same period of time.

The service is aimed at decision makers in the governmental and private sectors to be able to anticipate possible outcomes and act accordingly.
As a specific example, Recorded Future can be used by security intelligence analysts, to predict where certain online secretive groups are located based on the incidence of their activities as reported in mass media.
By tracking when most incidences occur and matching them to a typical work schedule, it is possible to detect the timezone and therefore likely location of these groups (Holden, 2013).

2.3 What is Twitter?

[1] Application Programming Interface - a set of subroutines, remote calls, libraries and/or classes that can be used by programs to access a service, to enable inter-operation and communication between programs.

Twitter is a social media network which allows users to express themselves with small text fragments up to 140 characters long. Its headquarters are in San Francisco, California, U.S.A.

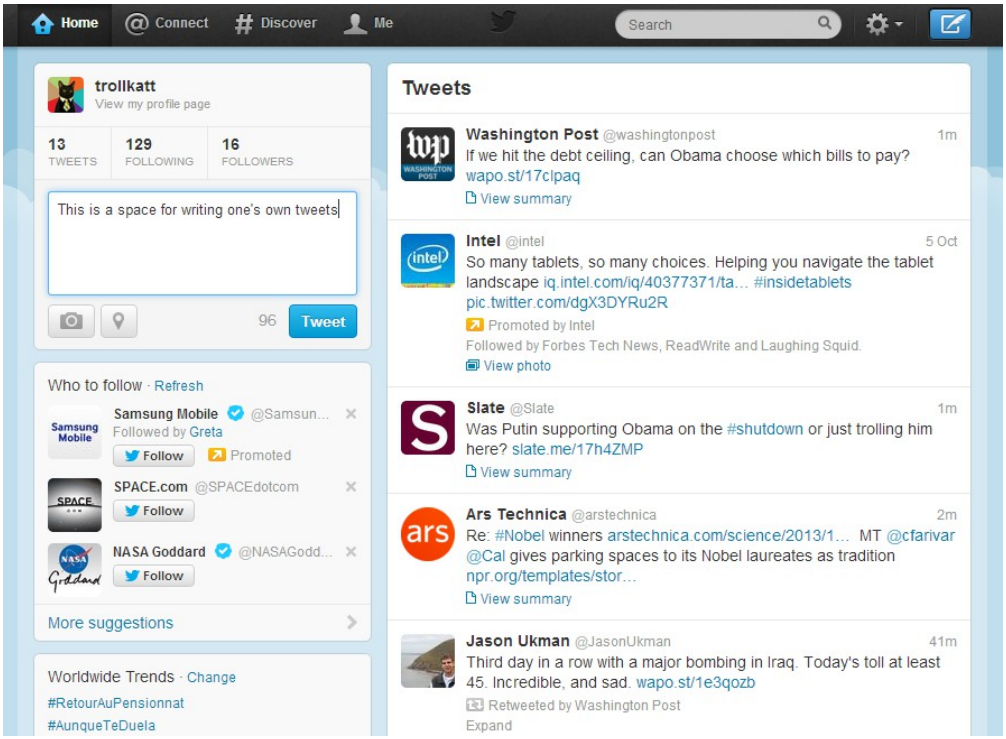


Figure 2: Twitter's main web interface

As shown in Figure 2, These text fragments called "tweets" are expressed publicly for everyone to read if they're interested (and if the user has not made them private). Pictures and websites can be included in tweets as links, and words can be arbitrarily marked with a "#" to give them a special significance.

Words marked with a # are known as hashtags, and they are used by Twitter to match tweets from similar topics. For instance, during a World-Cup related event, people who are writing about it might add the *#WorldCup* hashtags so that other users can know about them if they search.

When an interesting Twitter user is discovered, one can "follow" them. All the users' tweets will appear in the Home timeline, along with the tweets from other users that one is subscribed to and one's own tweets.

It is an especially useful tool to get to know what is happening with the topics and people one is interested in as well as "breaking news". It is also possible to have a conversation with other users, although the limited length nature of tweets make it inconvenient. Writing another user's screen name is known as a "Twitter mention". Since Twitter usernames are always preceded by an "@", it is possible to keep track of mentions easily.

3 Theory

3.1 The concept of entities, events and relationships

An entity, for this project's purposes, refers to objects, companies or people. Abstract ideas are also included and considered entities. Nouns and entities are synonymous in this case. However, only a subset of the nouns that exist in the events are analysed. Those deemed "relevant" in each happening are considered as entities, and the rest are ignored to avoid clutter.

An event can be a news item, a quote, etc. Something that happens somewhere in the world, involving one or more entities, and in the context of this work, is an abstraction for an article scanned by Recorded Future or a tweet obtained from Twitter.

A relationship is the link between an entity and an event - the participation or involvement of the entity or entities in the event. The involvement of an entity in an event can vary drastically from one event to another. For example, two different news items in the mass media about a well-known actor would have a different level of involvement if the first news item is about a movie in which he is cast the leading role, and the second news item lists him as attending a festival with many other actors. For this reason, all relationships are considered abstract and no distinction is made among them.

3.2 Visualisation

Many of today's processes, especially in science and engineering generate vast amounts of data that needs to be analysed. Typically, this raw data will require transformation to be useful due to its low level of abstraction - measurements taken directly from sensors in physical processes are often not as useful as when they are analysed.

The most effective transformation will depend on the nature of the data that is being represented. For example, if the data consists of discrete values or elements of different nature, different graphical symbols might be the most practical way to represent them.
If the data consists of continuous values, the most effective way to represent them might be instead, with a colour gradient or a change in relative size. Transformations can also involve grouping items with similar values together or removing those within a certain range - filtering.

This process of transforming the data to a representation that can be understood and used better is known as Visualisation.

Visualisation has been of interest since long before computers were available (Spence, 2007). Among the first documented work with data graphics are Priestley's time charts (timelines) in 1765 and Playfair's invention of the bar chart in 1786 as well as the pie chart in 1801. Joseph Minard's 1869 graph of Napoleon's campaign against Russia (Tufte, 2001) is also another example.

In 1967 "The Semiology of Graphics", the first theory of informational graphics, was published by a French cartographer named Jacques Bertin. In "The Semiology of Graphics", Bertin identifies the main components of diagrams and lays down a framework to design them. In 1977, Tukey emphatically

recommends the usage of pictures within statistics in order to quickly gain insights into the data, downplaying the graphics' quality - instead focusing on how useful these pictures can be even if they are "simple"; as a follow-up, Cleveland and McGill wrote in 1998 the book Dynamic Graphics for Statistics.

Tufte, another of the great theorists of data graphics, published in 1983 his theory, which emphasises maximising the density of useful information. The interest by the scientific community in visualisation was such, that the NSF launched in 1985 an initiative in Scientific Visualisation and in 1990, the IEEE organised the first conference on Visualisation. New fields, such as Automatic Presentation, marrying Artificial Intelligence to Visualisation, have sprung up, and the interest on the subject keeps increasing.

Nowadays, the advances in computer graphics and in processing of information have made interactive visualisation feasible - the person observing the data is able to modify the data transformations in real-time and observe the results. This has important applications where the user is able to take immediate decisions at the same time information is being produced and visualised and modify the process that is generating said information. It is a considerable difference in respect to the way visualisation used to be done before the advent of computers in everyday life: drawn or printed on paper, and static. Being able to instantly see the results of modifying the way the data is visualised (in some cases even the data itself) helps when formulating "what if?" scenarios - users are simply able to try many more alternatives and choose what is most suitable for their interests.

3.2.1 How does Visualisation help the users perform a task better?

Visualisation helps by amplifying cognition in the following ways (Card et al, 1999):

- It reduces the cognitive load of users by "offloading" data that would otherwise have to be kept in mind by the user to another medium.
- It enhances the user's capabilities to detect patterns by making certain characteristics in the data highlighted.
- It encodes information in a way that is modifiable by the user (if it is interactive).

3.3 Graphs

The present project uses graphs to visualise information. "Graph" is a term that covers a wide range of different representations; In this project however, the term "graph" will be used as a shorthand for "network graph".
Network graphs and concepts related to it are defined as follows:

3.3.1 Network graphs

A network graph, as shown in Figure 3 is a representation of a collection of vertices, some of which are joined together by edges. Vertices are an abstraction of objects and edges are an abstraction of the relationship between these objects.
Vertices are usually represented with a point and edges are represented by lines.

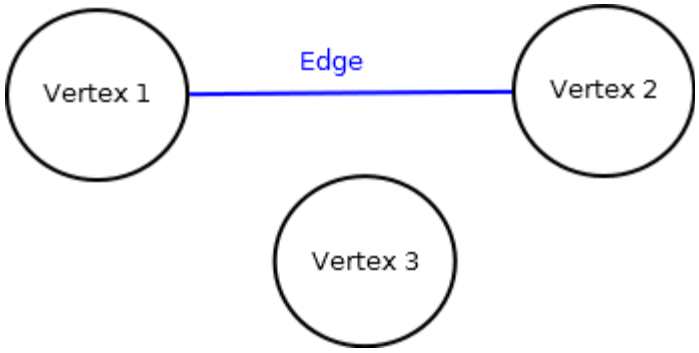


Figure 3: A small Network Graph

3.3.2 Vertices

A vertex is the fundamental unit of a graph. Vertices are also known as nodes in the context of network graphs. An abstract representation of an object. It can be labeled or unlabeled, and it can contain associated information (a quantity, a data point, etc.).

3.3.3 Edges

Edges link two vertices; these links can be unidirectional or bidirectional - that is, it is possible to traverse in only one direction or both directions. It is also possible to associate information to these links (for example, the time or cost needed to traverse it).

Visual properties such as colour or size of both vertices and edges can be used to convey information and also to acquire the user's attention. For instance, the bigger vertices stand out in comparison to other vertices, therefore they can be made to convey importance.

Depending on the graph, the positioning of vertices respective to other vertices might or might not have an associated meaning. When the graph is an abstract representation of some physical arrangement, for instance, a map for a public transportation network covering an entire city such as a subway system, the positioning of the vertices may loosely match the geographic location of the stations and the edges' length may also loosely match the distance between each station. These types of maps are frequently found on the information booths of the subway system in a size suitable to be looked at by several people simultaneously. In other depictions of the same subway, the vertices that represent the stations may be laid out flat on a line and equidistant of each other, perhaps so that the representation may fit in a smaller space such as a subway travel card.

The positioning of vertices greatly affects readability: In a study by (Purchase et al, 1995), it was found that increasing the number of edge bends decreases understandability of the graph. Increasing the number of edge crossings decreases understandability even further. In the study cited, when dealing with both the dense and the sparse graphs, on average the participants made almost twice as many errors when there was excessive edge crossing; edge bending did not cause such an effect, but was still close.

In conclusion even though vertex positioning may not formally have a meaning, when the graph is used for representation, careful placement of vertices will result in a more effective visualisation.

4 Planning

Due to the novel type of the data to the author, some time was needed to get acquainted with its nature. A general timeplan was formed, which included as one of the first steps conducting a literature search to determine how others had handled the data type. At this point, neither the platform nor the methodologies were decided. Those were decided after the initial literature search as well as analysing the data.

4.1 Time plan

The project was originally planned to be completed approximately within 20 weeks.
While the implementation was completed approximately according to plan, the report was delayed for personal reasons.

The original time plan is shown in Table 1

Weeks	Activities
1st	Planning and agreements with Recorded Future
2nd-4th	Familiarisation with the data type Literature search Testing frameworks for implementation
5th	Idea generation by brainstorming Discussing ideas with Recorded Future
6-10th	Prototypes get implemented and iterated
11th	Limited testing of final prototype
11th-20th	Report writing Fixing the prototype if necessary

Table 1: The original timeplan

4.2 Literature search

The literature search was initially conducted by consulting the following books related to Visualisation and Interaction Design:

- Preece, Jenny, Yvonne Rogers, and Helen Sharp. 2002. Interaction Design: Beyond Human-Computer Interaction. John Wiley \& Sons, Inc.
- Card, Stuart K., Jock D. Mackinlay, and Ben Shneiderman. 1999. Readings in Information Visualization: Using Vision to Think (Interactive Technologies). Morgan Kaufmann. http://www.amazon.com/Readings-Information-Visualization-Interactive-Technologies/dp/1558605339.

- Spence, Robert. 2007. Information Visualization: Design for Interaction (2nd Edition). Prentice Hall. http://www.amazon.com/Information-Visualization-Design-Interaction-2nd/dp/0132065509.

Additionally, several articles were consulted online, mainly from the Association for Computing Machinery Digital Library, http://dl.acm.org/ . Some of the keywords that were searched for include “graph theory”, “visualisation”, etc.

Although the literature search was supposed to be done only in the beginning, it was found to be very useful to formulate ideas and reading about how other people tackled similar problems. As a result, literature search was conducted often. Through this continuous literature search, Benjamin Fry’s methodology was found, which shaped how the project was developed.

In total, more than 95 articles were consulted, although only a fraction of them were applicable.

5 Previous work

A literature search was conducted for projects and articles in these three categories and a few examples of related projects follow:

5.1 Spiral Calendar

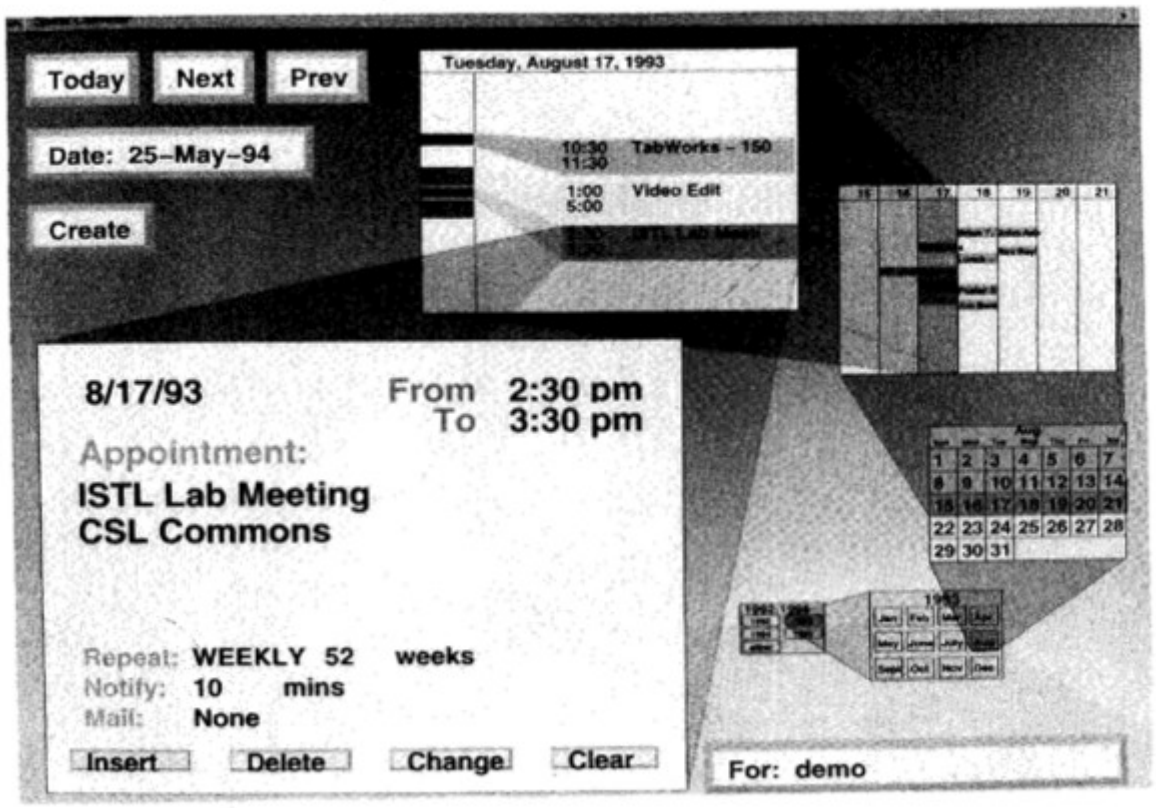


Figure 4: The Spiral Calendar's main interface

Mackinlay, Robertson and DeLine proposed in 1994 a program called the Spiral Calendar (Mackinlay et al, 1994) which allowed users to access their personal calendars with different degrees of granularity (Figure 4).

The application starts by displaying a general calendar, which moves into the background in a spiral motion when the user selects a specific item within the calendar. For example, it may start with the yearly view, which moves into the background when the user selects a month and presents the monthly view instead; the monthly view moves into the background when the user selects a day, and so on until a specific task view is reached.

The main view is large enough to be normally interacted with, and the views that slide into the background are still operational, so the user has access to both details and the context. The animations

exist to support the user in maintaining their mental model of the data, and were adjusted to last less than one second to avoid disrupting the user's work.

(Card et al, 1999) mention the advantage of using the spiral shape for a layout as an effective way to reduce the information cost structure, because the information that is more likely to be used at the time appears in the front, while the other views are sent into the background.

5.2 WiGis

The Web-based Interactive Graph Interfaces by (Gretarsson et al., 2010), a project at the University of California in Santa Barbara, focuses on visualising relationships between data items in the form of an interactive graph on the web (Figure 5). It employs an adaptive client and server-side approach to graph redrawing and recalculation with graphs consisting of hundreds of thousands of nodes.

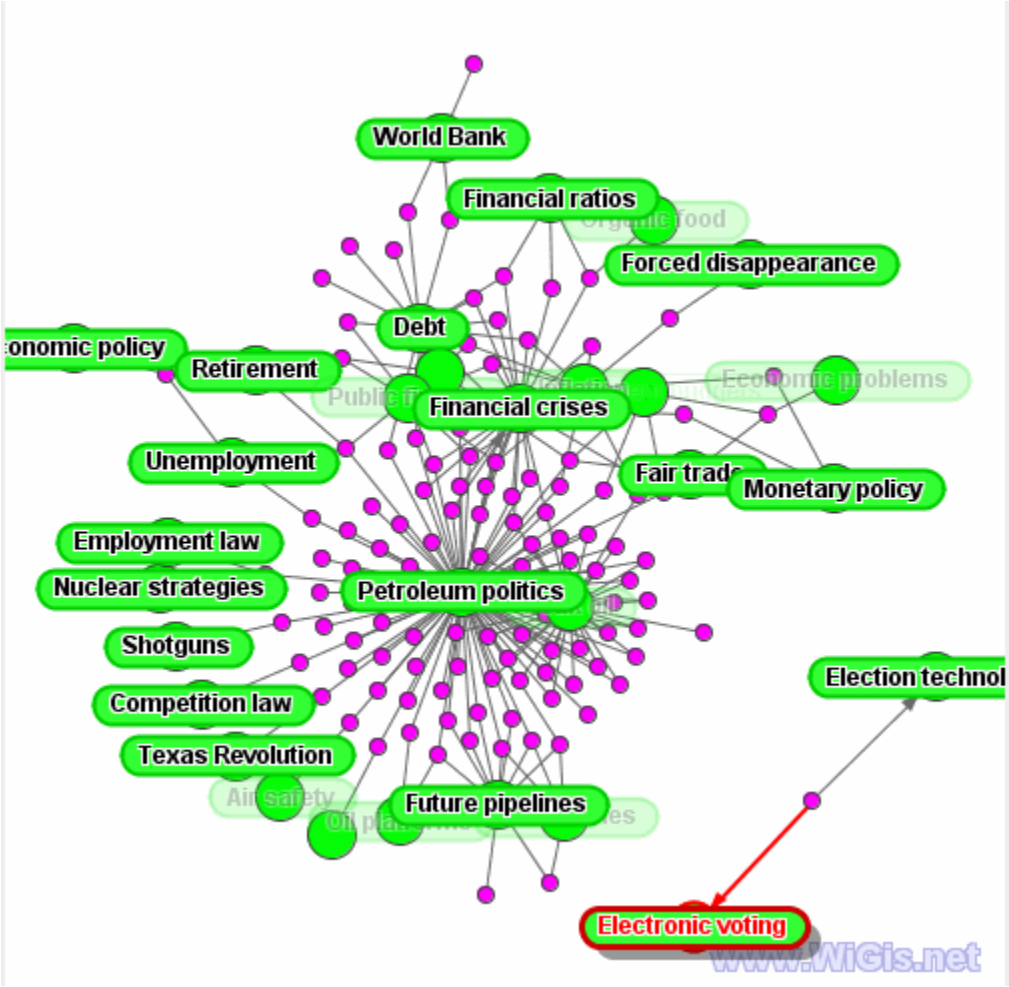


Figure 5: The main WiGis interface

Their system specialises in displaying files consisting of thousands of nodes whose data, properties (colour, etc.) and edges are specified in a file. The layouts available in the system are: Fructerman-

Reingold, Circular, MDS, Disjoint Graph and Spring. WiGis allow the user to relocate any node manually for better observation (e.g. in case its label overlaps with another node's).

5.3 New York Times' Twitter-traffic for midterm U.S.A. election visualisation

For the 2010 Midterm election in the United States, (Jackson et al, 2010) built a visualisation that displayed the relative frequency of Twitter messages containing information about the candidates (including messages generated from the candidates themselves), from the 21st of October 2010 up to the 2nd of November, the day of the election (Figure 6).

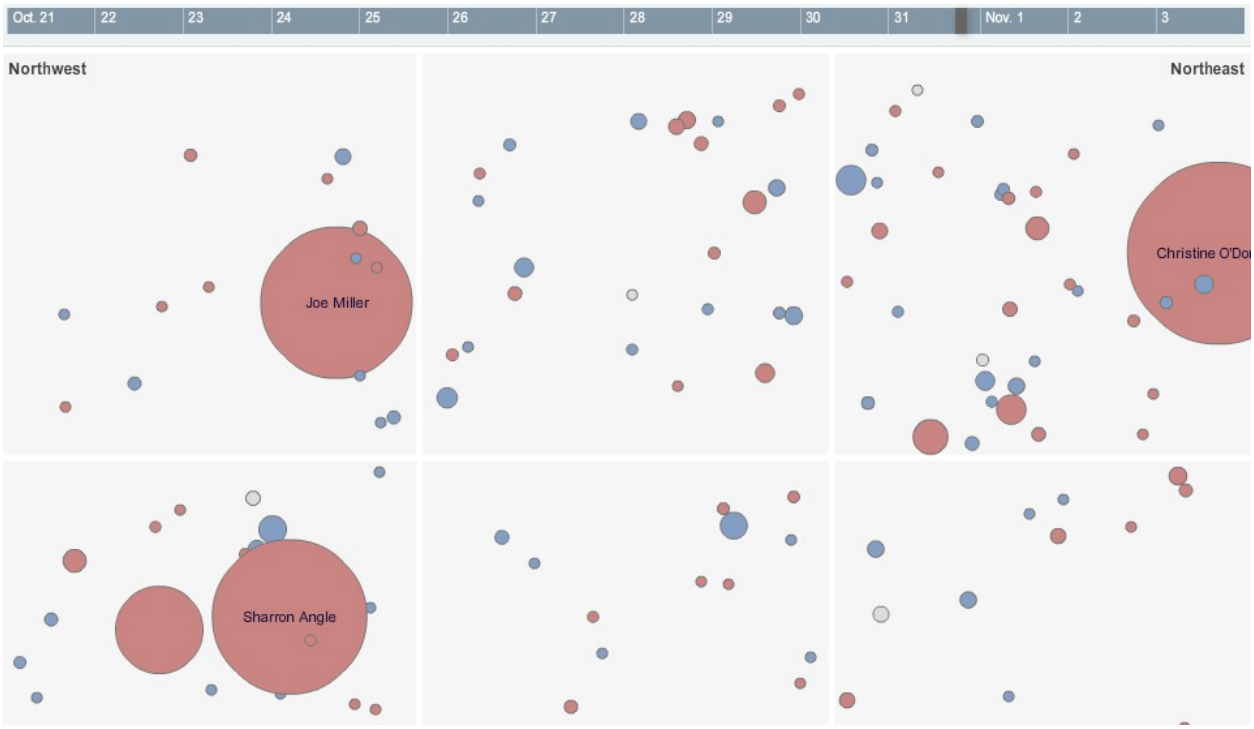


Figure 6: New York Times' Twitter traffic visualisation

Each node represents a candidate and they are coloured according to their political association.
Size is used to represent how active each candidate is: each time a candidate is talked to, mentioned or emits a message in twitter their node gets a bigger radius, but this radius decays over time, so if the candidate ceases to be mentioned, their node will become smaller.

The current project and the New York Times' visualisation are related in the sense that they attempt to visualise events happening within a time context, associated to entities: in the New York Times' visualisation, political candidates become entities and the events are the messages they write in twitter as well as the ones written about them by the general public. The focus, of the visualisation however is not in the content of the messages but rather in how many each candidate sends or receives, as a measure of "popularity".

6 Methodologies, methods and frameworks

For this project, several methodologies were used:

- Literature search, described previously, was continuously conducted, to help with solving the problems that arose.
- As an overarching methodology, Benjamin Fry's steps for creating interactive visualisations was used.
- To generate ideas for the possible specialised visualisations, brainstorming was used in several steps.
- The visualisation idea considered as the best one from the brainstorming was iterated several times to test and improve it.
- Several frameworks were tested during the pre-planning phase, however only the one which was actually used (Processing) is described in detail.

6.1 Visualisation method - Benjamin Fry's methodology to formulating an interactive visualisation program

According to (Fry, 2008), the visualisation process consists of seven steps:

1. **Acquiring the data**

The data is obtained from the sources - from a file in the computer, from a server on a network, or even from sensors measuring the phenomena being analysed.

2. **Parsing the data**

The data obtained from the sources must in many cases be rearranged or transformed to be useful. For instance, measurement data might have been stored as a long one-dimensional list, whereas handling it as a matrix of several dimensions might make operations easier to perform.

3. **Filtering the data**

In many cases, not all the data is useful. Choosing which part of the data is not useful is a task that must be done with care. An erroneous decision could discard the answer to the problem in this step. Leaving all the information in, on the other hand, might overwhelm the system or the user.

4. **Mining the data**

Applying methods to discern patterns in the data. One such example would be detecting the locations of minima and maxima in measurement data, then grouping them together. Marking these groups to be highlighted in the final representation might give the user better insight into the data.

5. **Representing the data using a general model**

The data is represented using simple models such as bar graphs, lists or a table.

6. **Representing the data using a model better suited for the information's nature**

A representation that is more specialised is proposed. This representation should make the data easier to comprehend and be visually engaging, and could also be unique to the problem that is being solved. Suppose there is a process that generates data critical to the safety of an industrial complex, in several geographic locations simultaneously. Instead of a simple 2D bar graph showing these measurements, a 3D one that is superimposed over a map and highlights interesting measurements with a special colour might enable the user to make faster decisions, because the user would not need to mentally connect each bar to its corresponding location.

7. **Adding interaction**

The user is given tools to modify the representation, such as filters, the possibility to highlight certain region, showing views with a greater amount of details, a general "overview" where only the most important details are shown, etc. The interaction can be continuous - for instance, the user moving a lever that modifies the visibility of the data with immediate feedback, or stepped - the user is presented a group of parameters that can be modified and sees the change once she has finished her selection. These parameters can be incrementally applied to obtain the desired data that meets all of the user's requirements.

(Fry, 2008) mentions that though these steps are a useful guide, visualisations can differ greatly. Therefore these steps are not to be taken as an absolute step-by-step guide, but rather for the designer to apply those steps that are deemed convenient. For this project, most were considered useful and thus were applied successfully to the entity-relationship model.
The Spiral Diagram is the visualisation which was found to be the most suitable after following all of the steps previously mentioned, except the "data mining"; it was found to be unnecessary to make further transformations of the data.

(Card et al, 1999) makes one important observation: that even though visualisation started as a reference to representations intended to be perceived by sight, nowadays the term has been extended to encompass any representation that can be perceived by humans: auditory, tactile, etc. For example, a tactile map designed for the blind is also considered a visualisation. This is important because it both broadens the possibilities when designing a representation and makes visualisation more accessible to users with impaired capabilities.

Card divides visualisation in two broad categories: Information Visualisation and Scientific Visualisation. Both are similar since they involve representing data through computerised and other means. The main difference between them is that in Scientific Visualisation, the data must come from measurements, being the result of physical processes, while in Information Visualisation, the data can instead come from non-physical processes, such as business processes or social interactions.

While there are no limiting factors to the kinds of representations or models that can be used for either classification, certain models are better suited to Scientific Visualisation than to Information Visualisation.

6.2 Design method - Brainstorming

Ideas are written on cards or pieces of paper for a certain length of time. No idea is judged or discarded until the production period was over; crazy ideas are welcome.

Once the idea production period is over, all the ideas are evaluated.

6.3 Frameworks

The following frameworks were evaluated:

- Mono/GTK#
- GNU C++/OpenGL
- Processing

A general description of each one follows

6.3.1 Mono/GTK#

Mono is “an open source, cross-platform, implementation of C# and the CLR that is binary compatible with Microsoft.NET”[2]
The main purpose of Mono is to provide an open source implementation of the specifications for Microsoft’s C# language.
It is a general-purpose programming framework that provides automatic memory management (trash collection; which eases development as the programmer does not need to handle it) among other features. C# can be used to create any kind of application, provided there is a Mono virtual machine or a .NET virtual machine in the target platform. It currently works in GNU/Linux, Microsoft Windows, Mac OS X, BSD and Sun Solaris. It also works on the iPhone, Nintendo Wii and Sony Playstation 3.
Other common uses for Mono are scripting and embedding inside another application.
Mono programs are generally written using the Monodevelop IDE, shown in Figure 7.

[2] http://www.mono-project.com/Main_Page

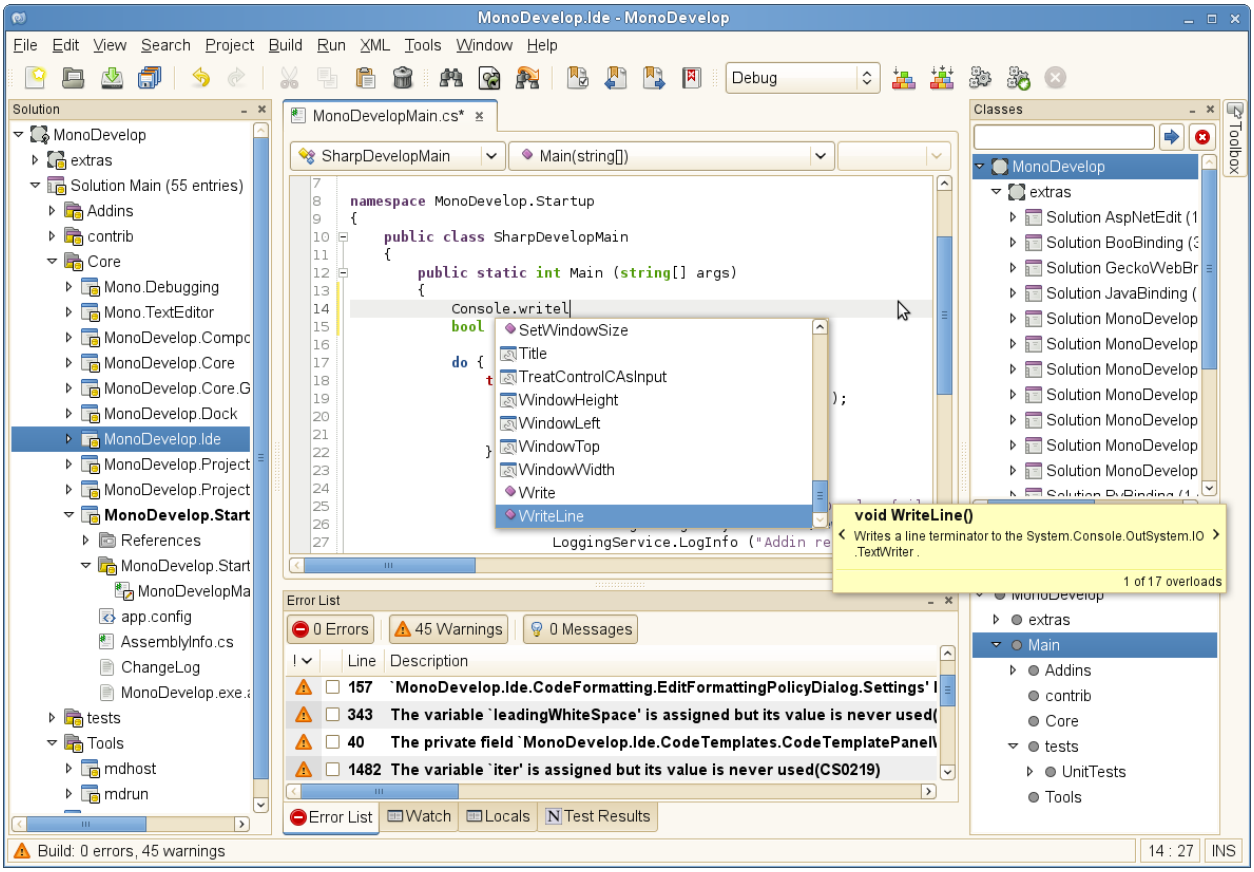


Figure 7: The main Monodevelop window

GTK#

GTK# is the .NET implementation of GTK+, a UI library designed to be used by the GNU Image Manipulation Program (GIMP). It was adopted by the GNOME desktop project for general UI development in applications in GNU/Linux. However, GTK+ has been ported to Windows, Mac OS X and BSD operating systems, among others.
GTK+ has several sub-libraries providing different functions such as threading, text layout and rendering, and 2D graphics.

6.3.2 GNU C++/OpenGL

The GNU C++ compiler is the part of the GNU Compiler Collection (GCC) that processes C++ code. C++ is one of the most widely used programming languages. All kinds of programs including game engines, utilities and even device drivers can be written in C++.
The GNU C++ compiler does not feature automatic memory management, but programs written in the C++ language can be written to perform very efficiently when dealing with massive amounts of data or operations, compared to other languages (especially those that require a virtual machine to operate).

The GNU Compiler Collection is available for nearly every operating system in existence, including GNU/Linux, Windows, Mac OS X and BSD. There are a multitude of text editors and IDE[3]s that can be used to write code in C++ . One such IDE is Codeblocks, which is available for GNU/Linux, Windows and Mac OS X. CodeBlocks is shown in Figure 8

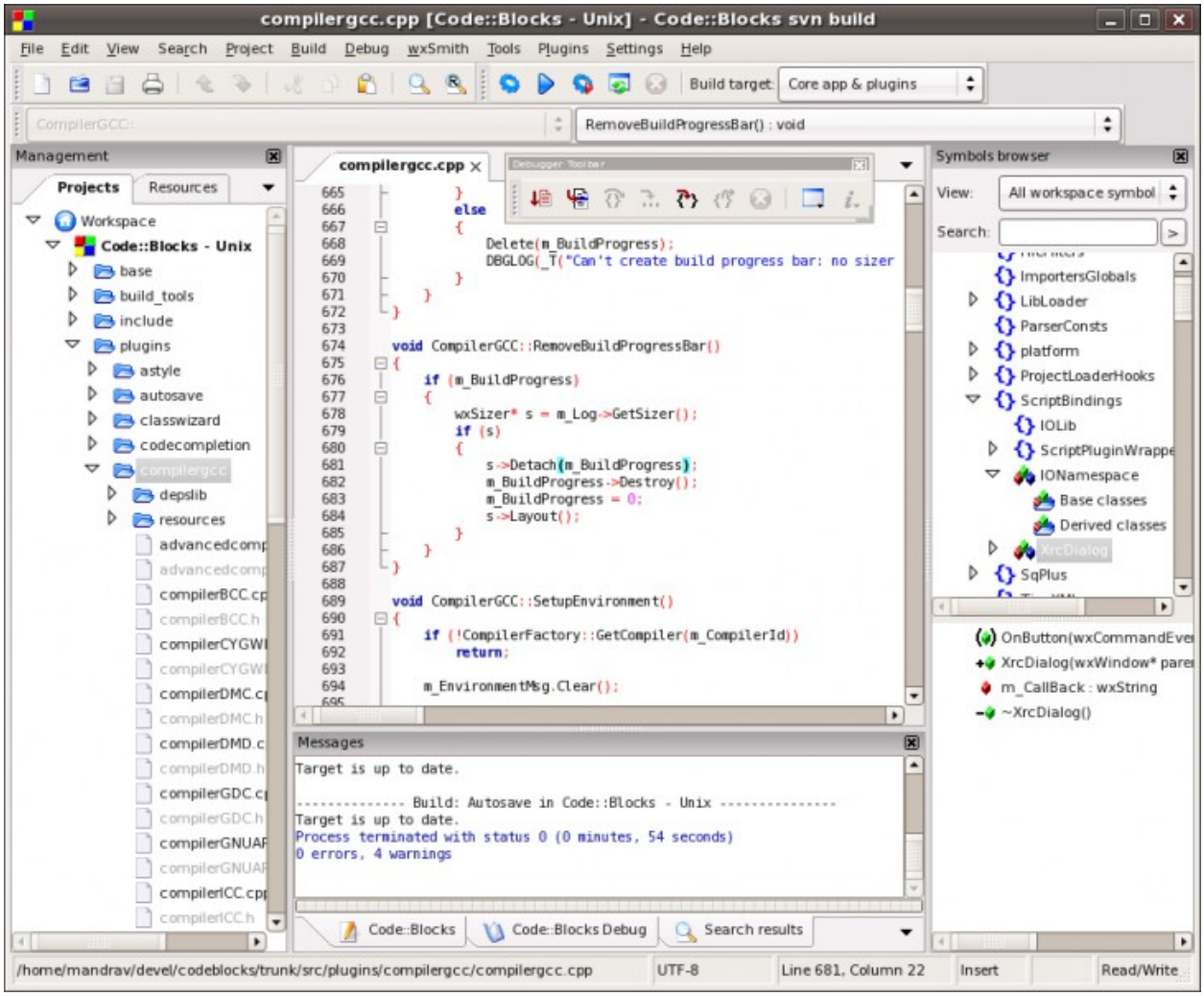


Figure 8: The CodeBlocks IDE

OpenGL

OpenGL was created by Silicon Graphics in 1992; it is an API created with the purpose of rendering 2D and 3D computer graphics. Nowadays it is managed by the Khronos Group, a non-profit technology consortium.

The OpenGL API is usually exposed as C-language libraries (which are compatible with C++), and though its implementation depends on the platform and graphics processing unit vendor, the standard set of OpenGL functions is guaranteed to work on any and all implementations.

Several implementations of OpenGL exist for GNU/Linux, Windows, Mac OS X, BSD and others.

[3] Integrated Development Environment - generally a text editor with code completion, compiling and debugging facilities.

OpenGL is used in many game engines and other graphics-intensive applications such as simulators and visualisation tools.

6.3.3 The Processing framework

The Processing framework specialises in creating custom visualisation models. It was started in 2001 by Casey Reas and Benjamin Fry at MIT Media Lab. Initially its purpose was to provide a software sketchbook and to teach the basics of programming languages focused on visualisation. Nowadays it is actively used by students, artists, designers, researchers and other professionals including production work[4].
Its main tool, the code/sketch editor is shown in Figure 9

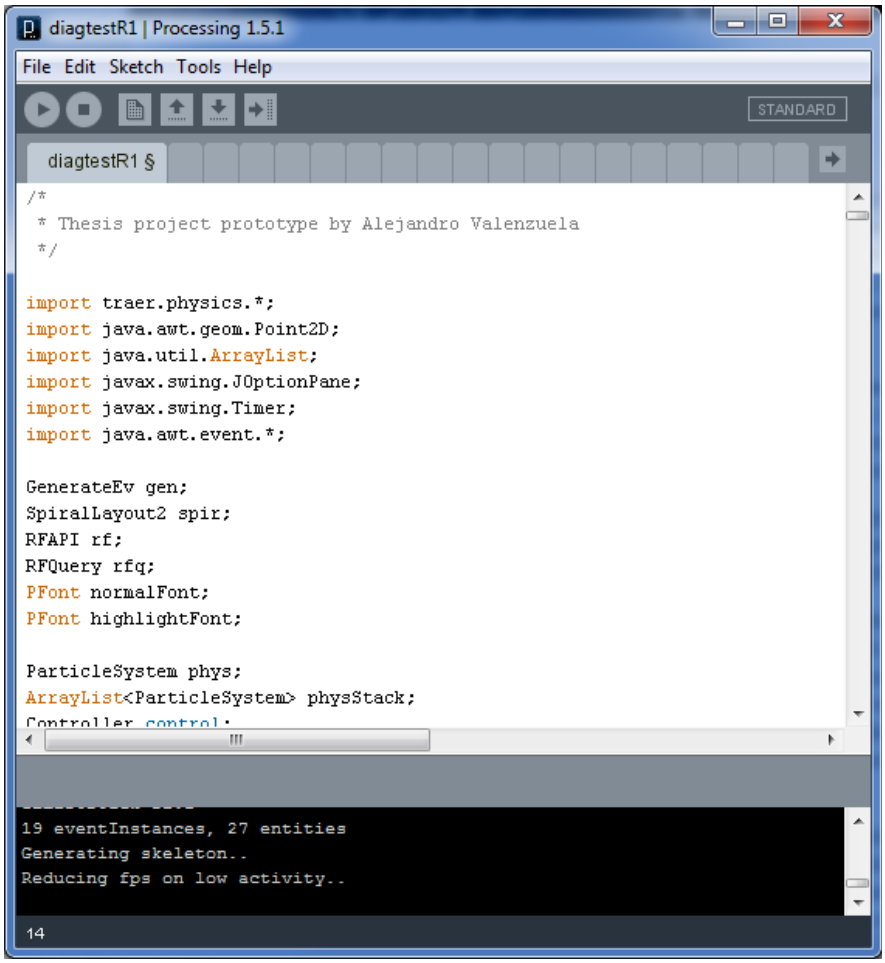


Figure 9: The Processing editor

Processing is based on the Java language but uses a simplified syntax and graphics model.
Its API has functions pertaining 2D and 3D graphics, image, audio and video processing, network and serial communication. A great number of Java libraries are also compatible, increasing its usefulness. Additionally it can export a stand-alone executable in Windows, GNU/Linux and Mac OS X. This enables Processing users to redistribute their programs for others to use without needing to install the framework.

[4] http://processing.org/overview/

Processing has an integrated editor in which users can develop programs called “sketches”, which can invoke both the Processing API and external library calls. The editor is rather simple if compared with traditional editors such as Eclipse or Visual Studio. It includes code highlighting and reduced controls: “Run”, “Save”, “Open” and “Export”. As such it is highly recommendable for creating prototypes and testing ideas rapidly as well as for beginners since the set-up procedure for Processing is fast and straightforward. Moreover the reduced controls make it more-or-less impossible for new users to be confused. This is in line with the goals of the Processing project, yet it does not limit the complexity or usefulness of the programs that can be created with Processing, nor does it exclude experienced programmers from using the framework. A more complete editor such as Eclipse[5] is however recommendable when the programs or “sketches” become complex.

Programs created with Processing tend to differ from traditional programs in that they do not usually have a “traditional” interface: buttons, text boxes, list views and other components are nonexistent or do not have the usual aspect or expected functionality. Whatever components are present for the user to interact with a Processing program, are usually custom-made for it.
Another way in which Processing programs differ is that their usage pattern is iterative and exploratory. The user is generally expected to interact with the program by performing small adjustments with the mouse and observing the result, repeating the operation until the relevant information becomes apparent, or the user gains some kind of insight.
Arguably, the lack of traditional interface controls could be counted as a disadvantage for the framework, but in the case of this project -and the projects which the framework is intended for- complicated interface controls are not usually needed.

Overview of a Processing sketch

A Processing sketch is a text file created using the Processing editor, which calls instructions from its API which is a subset of Java. The programming instructions’ syntax is that of Java, though many of the functions usually do not exist in Java.
In its simplest form the sketch is comprised of two programming “blocks” or subroutines: the setup and the draw blocks.

The setup block describes the initial conditions for the Processing sketch and initialises its environment. Many environment options are set to default values, so it is not always necessary to specify every option. For example, by default, the 2D renderer is enabled and geometric figures will have black borders and be filled with white colour. The number of frames per second- the number of times the screen is updated each second- can also be also specified here. Another possible setting is that the program will not automatically iterate several times per second, but rather when there is user input.

The draw block is called each time the screen needs to be updated. The purpose of this block is to paint what needs to be shown on-screen, based in the user’s input and other underlying processes.

Other blocks that can be used are related to user input - mousePressed, keyPressed, etc.

[5] http://www.eclipse.org/

Inside the blocks previously described, it is possible to call functions from external libraries once they are imported into the sketch. Importing is done through a menu option in the Processing editor. Creating and using custom classes is also supported, although only sketch files (and not custom classes or external libraries) can make use of the Processing API.

Continuous execution

Unlike other frameworks, Processing does not impose any programming model. Users are free to choose how to implement their solutions as long as they contain the set-up and draw blocks.

The only assumption Processing's API does, is that the program will continuously update the display based on user input in real-time.

The way Processing programs are expected to be used differs from the way programs are generally used. For example, many programs are intended to be opened with a goal in mind, a specific functionality used, a result obtained, and then closed.
In this manner, Processing programs are more similar to games than general applications: the user is expected to open a game perhaps (though not necessarily) with a goal in mind. What is definitely not expected from a game user, is to open the game to use one specific function from it and then exit; instead she is expected to interact with it in a continuous, real-time manner and after she is satisfied, exit.

7 Design following Fry's methodology

Since the objective of the project was to develop a suitable visualisation to display many small text fragments (approximately one paragraph), the requirements for this project were decided to be:

- The fragments should be visible in such a way that the user will not be overwhelmed, yet as much information as possible should be shown.
- The visualisation needs to be able to accommodate new text fragments in real-time.
- The visualisation should enable the user to choose what she wants to see.
- The visualisation should run at a frame rate adequate for interactive manipulation.
- As this is an experiment in visualising the data type, a prototype was decided to be sufficient in terms of complexity and functionality, therefore many aspects not essential to the visualisation like network security, system load, etc., were not prioritised.

These requirements were specified in very general terms in the beginning of the project, due to the novelty of the data's nature. More specific requirements were defined as research went on and the author became more familiar with the nature of the data.
Since Fry's methodology involves alternating design and implementation; the project's non-visible part was implemented before planning and iterating the visualisation part. This non-visible part is the same for most visualisations: it is only making the data available in a way that is easy for programs to access and use it.

7.1 Data acquisition and analysis

Initially, the data acquired were only the results returned by the Recorded Future system, which consisted of text fragments with several properties. However, due to the similarity in structure between the Recorded Future data and Twitter messages and in the interest of developing a solution that would work for more than one source (i.e. less constrained applicability), Twitter data acquisition was also implemented.

The data acquisition procedure in Recorded Future consists of the following steps:

1. A query, expressed in Recorded Future's syntax, is sent to the Recorded Future server through their API.
2. The Recorded Future server will search its database according to the specifications of the query and assemble the results.
3. The Recorded Future server returns the query results, also formatted in JSON syntax.
4. These results then need to be parsed and reorganised to be turned into useful information according to the project’s internal structure.

The data acquisition procedure in Twitter involves:

1. Querying the Twitter server using the JTwitter API
2. The Twitter server returns data which the JTwitter library parses into its own data structures
3. The data from these results then needs to be copied into the right fields in the project’s internal structure.

The internal structure of the data is analysed in the following section.

7.2 Data parsing

Data from each service is obtained in a slightly different format, which must be organised into the project's internal structure for events (which is the same regardless of the data's origin).

The internal structure for events has the following fields:

- **eventID** - stores the unique ID for this event. It corresponds to Recorded Future's own EventID or Twitter's Tweet ID.

- **startTime** and **stopTime** - Recorded Future's events have a duration, which is why there are startTime and and a stopTime fields. In the case of Twitter, both are set to the Tweet's publishing time, since the duration of an event does not make sense in the context of Twitter.

- **fragment** - stores the complete event's text.

- **participants** - a list of entities related to the event.

- **title** and **longTitle** - respectively, the abridged and unabridged versions of the title for an event. Showing details on-demand is a pattern commonly found in Software - the unabridged version of the title stores more details in case they need to be shown.

- **momentum** - the relevancy that the system reports for a specific event. In the case of Recorded Future, it is a field included with the query results. In the case of Twitter, it is the number of times an event has been re-posted (re-tweeted).

- **url** - the web-address of the article mentioned in events, In the case of Recorded Future, it is included within the search results; in the case of Twitter, it is detected and extracted from the article's text.

The data obtained from Recorded Future is obtained in JSON syntax. The relevant portions of data must be copied in the appropriate fields of the internal structure.
Recorded Future results include fields for the related entities, relevance and an associated web page.

In the case of Twitter, the data is obtained as JTwitter objects; from each Tweet object the hashtags are detected and considered related entities. The number of retweets is taken as a measure of relevance under the assumption that content that is repeated many times is likely to be more relevant for the user.

7.3 Data filtering

Initially, the only filtering being applied to the data is server-side. In the case of Recorded Future, the query used during the data acquisition phase limits the information that the server sends back as a result.

For example, it is possible to limit the query results to “company acquisitions”, “product launches”, “quotes by a specific person”, among others. This greatly affects the kind of results the Recorded Future server returns.

In the case of Twitter, the user's pre-existing timeline is the main filter. Only those messages written or re-tweeted by members of the user's timeline are returned from the query.

Once the program has the initial set of results, it's possible to filter further for specific entities, relevance or a keyword.

7.4 Representation using a general model

A few models were attempted before choosing the spiral one:

- A flat list of messages similar to a Twitter timeline.
This turned out to be less practical when implementing interaction with the related entities. To be able to interact with the related entities, either a significant amount of space would need to be left blank to fit details on-demand, or the entire list would need to be “pushed” to display those details.

- A representation with nodes automatically laid out by Dotty, a layout program.
Unfortunately no clear order pattern was apparent in the layout, which made it impossible to quickly detect which events came before the others (Figure 10).

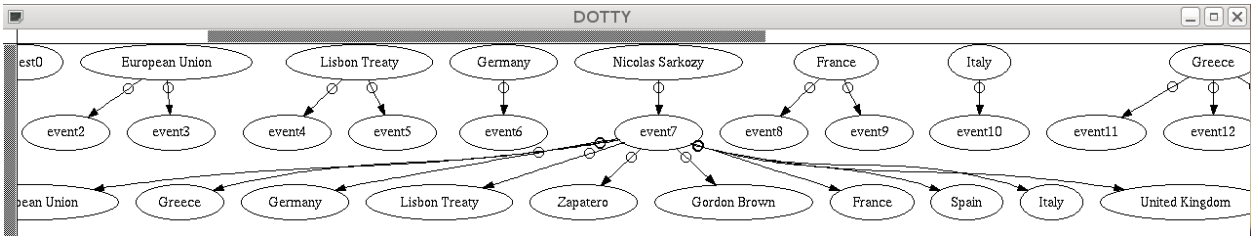


Figure 10: Nodes automatically laid out by Dotty; small number of items

When there were many items, paths between them started crossing (Figure 11).

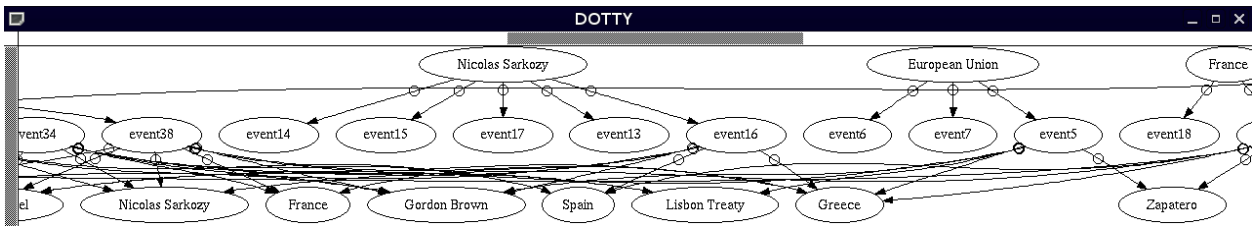


Figure 11: Nodes automatically laid out by Dotty; large number of items

Overall it was found that simple models did not work correctly because of the sheer amount of data that is desired to be displayed simultaneously or because of its interconnectedness.

7.5 Representation using a model better suited to the data

Once it was apparent that the number of messages returned from the server would usually be very high, it became evident that it was necessary to formulate a representation that allowed for many of the results to be present simultaneously (but not necessarily all of them) and to show detailed information for each result on demand. Another aspect to consider was: while the program is running, it is possible that more items are received from the data sources. When this happens, they must be placed in a location that is consistent with some kind of order, if necessary, displacing some of the existing items.

Our vision then was that the interaction between the user and the diagram should occur in the following steps:

- **Exploratory phase**

The user browses a large amount of items. In this case the objective is that she should be able to see as many events as possible, from a general perspective. Many details are unnecessary to convey the general meaning of each item shown on-screen. Thus most details will be hidden.
One detail is however quite important: the relevancy of the item assigned by the data source. To convey this, a difference in size or in colour is used in most visualisations.
The exploratory phase should then be an overview of many events, where it is possible to quickly pick which ones would be the most relevant.

After the user has found an interesting item, she might want to read more details about it. This is another usage phase: the inspection phase.

- **Inspection phase**

During the inspection phase, the user interacts with an item of her choosing. She must be able to see all the details for the chosen item, including its order in time relative to other items, the complete text fragment and all related entities. From this phase she could either select another item for inspection, go back to the exploratory phase or go into the focused exploration phase.

- **Focused exploration phase**

The focused exploration phase involves filtering items when the user has in mind what she wants to find. This phase is a simplified exploration phase. There are two ways to start the focused exploration phase: filtering by a known aspect (a keyword) and filtering by a specific characteristic from an existing item. The latter involves first locating an interesting item, inspecting it, and selecting one of its characteristics; for example, one of its related entities.
The items shown on-screen will be only the ones that have the desired aspect. Any number of filters can be applied to further reduce the number of results shown to make finding the item the user is interested in easier.
While new items can be received from data sources at any time, only those which meet the filters should be displayed. If the new items do not meet the filter, they are stored but not displayed until the filter is removed.

- **Background phase**

If the program is functioning but not currently visible, or the user is not attending to it, new items can still be received and accommodated considering the filter situation. The user can return at a later time and switch into one of the other phases.
If the program detects that it is not being attended to, it may take some shortcuts regarding animations.

7.5.1 Layouts

Before proposing layouts for representation, general graph drawing algorithms by (Di Battista et al, 1994) were considered as a guide and for inspiration.

As written previously, several representation models with different layouts were proposed and tested.

- **Parallel entity trees**

A tree-like structure, with each entity at the root and events as leaves, with the “tree”’s branches joining leaves where two or more entities participate.

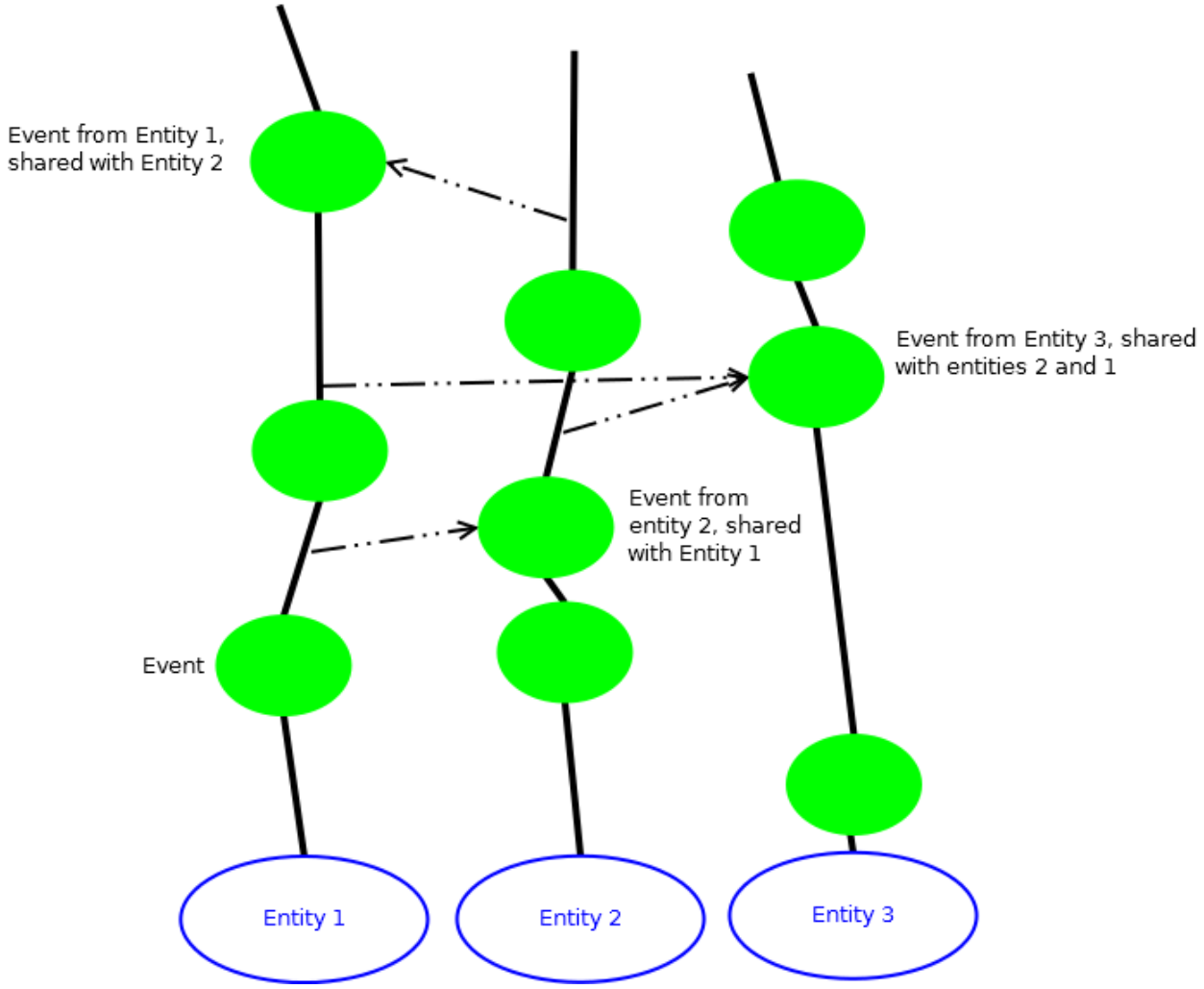


Figure 12: Tree-like layout displaying shared and non-shared events

From the root a tree trunk grows; events in which the entity is involved in as the main actor appear as leaves. Branches grow from one tree to another tree's leaves to represent events in which an entity is involved in, though not as the main actor (Figure 12).

This layout expresses the entity-relationship model explicitly and has an interesting shape that might be esthetically appealing. Unfortunately, had it been implemented and supplied with a typical set of results from a Recorded Future, it would have shown some impractical details that emerge as a consequence of the high number of entities that generally participate in an event. The way the trees branch, in combination with the Recorded Future data would have led to much entanglement between the different trees because entities generally share many events. Furthermore it would have been difficult if not impossible to know which would be the "main entity" for each event: The data does not explicitly mention who is considered as the main entity for an event and in many cases this would be open to subjective interpretation.

In the case of Twitter however, this representation might be applicable: events do not tend to involve as many entities simultaneously as Recorded Future's, and there is always an author to each message which could function as the "main entity". Branches would grow from the tree towards other trees' leaves when "mentioned" in tweets.

However, regardless of the data's origin there's typically a great number of entities. This would result in too many trees being represented simultaneously. Filtering out some of these trees would solve this problem but it is an inconvenience that needs careful consideration to avoid leaving out trees that could be relevant.

After proposing this layout and analysing its implications, it was concluded that a functioning solution would require to display certain details on-demand.

- **Spiral diagram**

After several sketches, a spiral-like layout with events growing from the center in chronological order was proposed (Figure 13).

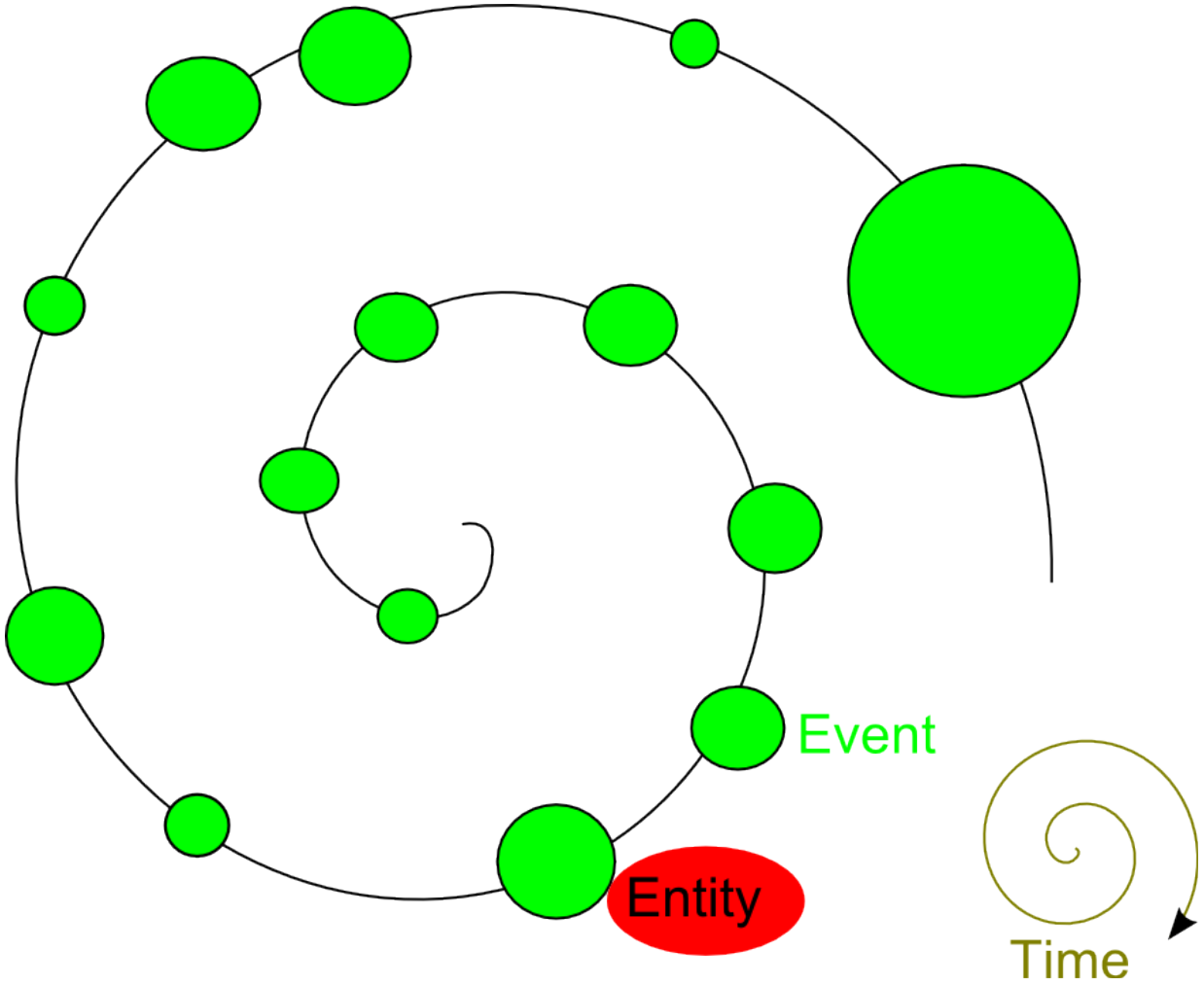


Figure 13: The Spiral Diagram concept

Related entities appear attached to the event they're related to. An idea to avoid excessive clutter is to only display the entities when the user focuses an event.

It did not exhibit the problems described in the parallel trees layout, so it was decided to further explore if it was useful due to its aesthetic appeal and to see whether there were practical advantages to a flat list.

Spirals have always been noteworthy to mankind. Its occurrence in nature and art is frequent.
In nature, for example, the polyphylla variety of Aloe exhibits a spiral arrangement of its petals (Figure 14 [6]), many orb weaver spiders create their webs in a spiral shape, many mollusks' shells grow in this shape, tornadoes and cyclones and even very big things, such as the arrangement of star systems in galaxies can be a spiral.

Figure 14: Aloe polyphylla and its spiral arrangement of petals. Photo: "brewbooks"

.

In art, the shape has appeared countless times:
One such example is shown in Figure 15:

[6] http://www.flickr.com/photos/brewbooks/184337424/

Figure 15: Tránsito en Espiral (Spiral Transit) from surrealist artist Remedios Varo

Mathematical equations for drawing spirals

The two most popular forms of spirals are the Logarithmic and the Archimedean spirals.

The archimedean spiral is best defined in polar coordinates by the equation

$$r = a \cdot \theta^{\frac{1}{n}}$$

where r is the radial distance, θ is the polar range (angle), and a and n are parameters that determine how close one loop of the spiral will be to the next one (Weisstein, 2013a)

A logarithmic spiral is defined by

$$r = a \cdot e^{(b \cdot \theta)}$$

where r is the radial distance, θ is the polar range (angle) and a and b are parameters determining how fast the distance between each loop will increase (Weisstein, 2013b).

For this project, the Archimedean spiral was chosen because having a fixed distance separating each loop in a spiral is better than an increasing distance. The latter is the case for the logarithmic spiral, where the distance between each loop would increase as the spiral winds out.

This spiral layout also had some practical advantages over a flat layout: it is easier to display extra information on-demand without overlapping other items. When new items arrive, it is possible to accommodate new items in the center and animate the displacement of the pre-existing items outwards on the spiral path, and I consider this to be less disturbing for the user than most similar rearrangements in flat lists, largely as the result of the spiral exhibiting a certain radial symmetry, which is conserved even when the displacement is being propagated. The change starts slowly in the center, but as it propagates, it becomes faster due to the increased arc length.

The spiral shape used in this project is generated by converting the Archimedean spiral polar equations cited previously to their cartesian form and considering $n=1$. The conversion is necessary for most of the Software used in the implementation.

The cartesian form of the Archimedean spiral with $n=1$ are the following mathematical equations:

$$x = a \cdot \theta \cdot \cos(\theta)$$
$$y = a \cdot \theta \cdot \sin(\theta)$$

for θ and a : real numbers; $\theta \geq 0$ and $a > 0$.

Where x and y are the cartesian coordinates and dependent on θ . θ is a real non-negative variable that, when it increases, it causes the spiral to unwind. a is a constant that defines how separate each loop of the spiral will be.

Each item to be displayed on the layout is assigned a certain value for θ . Items are laid out in a chronological order, starting with the most recent one. The intervals between these θ values are not uniform; in order to not overlap, the items that are closest to the start of the spiral need to have very different θ values assigned to them. As the items are laid out in the outer loops of the spiral, smaller increases in θ are required.

8 Visualisation implementation

Implementation was done in iterative steps. The very first step was to evaluate different frameworks in order to determine which best suited the general requirements established during the design phase.

All these frameworks have advantages and disadvantages and a trial implementation was conducted in several of them.

Mono/GTK# was the framework that was desired to use in the beginning, due to its multi-platform compatibility, general purpose, automatic memory management and ease of coding. Unfortunately it exhibited some bugs and the program crashed when several items were drawn.
The crash was caused by errors in one of the libraries GTK# is composed of. There was no workaround for such a problem at the time, and waiting for the GTK# developers to fix it (or trying to fix it oneself) was not an option due to the project's time constraints, so Mono/GTK# was discarded.

C++/OpenGL was also considered, however when Processing was evaluated, it was found that the Processing API was simpler to use than OpenGL in general, and especially in one key point: Processing has a text rendering function whereas with OpenGL it would be necessary to find a separate text-rendering API and adapt it. The fact that Processing has automatic memory management (the Java trash collector) also made it a more attractive framework, despite the performance that can be achieved with a C++ and OpenGL when dealing with massive amounts of data. For these reasons, in the end Processing was chosen over C++/OpenGL.

8.1 First prototype

The first prototype in Processing (Figure 16) showed an abridged version of each news article. The spiral shape was present and explicit. Events are drawn as simple boxes with text, all the same size; the associated entities were not present, and there was no interaction or indication of relevancy. The physics engine was not yet integrated - the items were animated while they reached their proper position, and then they became static.

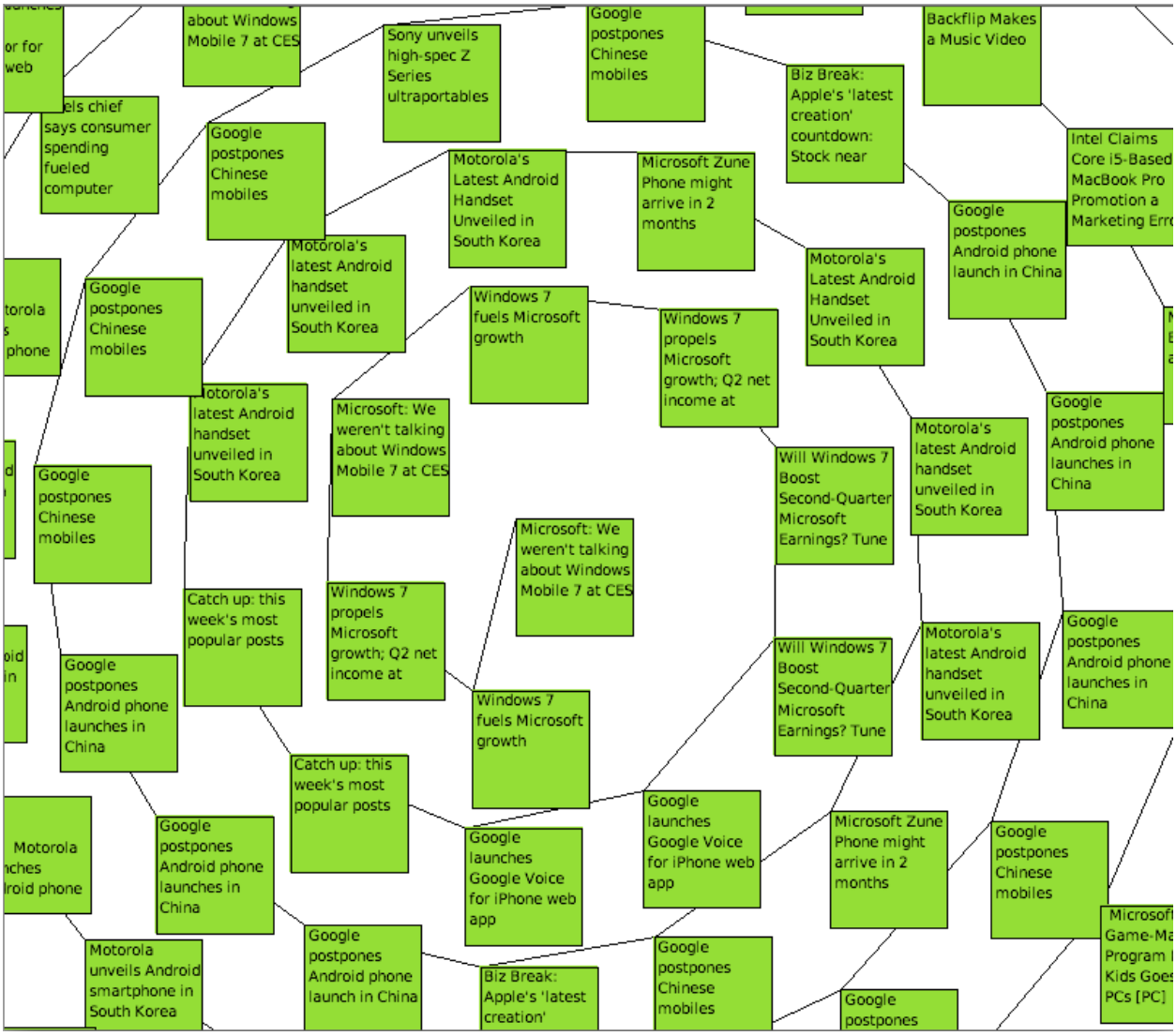


Figure 16: The first prototype

This prototype did not have Twitter functionality yet, but it was able to load Recorded Future query results from a file, or to query Recorded Future directly. Panning was not yet implemented, but it was one of the features that were implemented immediately after.
At this point the event and related entity classes were created.

8.2 Second prototype

In the second prototype (Figure 17), the physics engine was added, which allowed smoother animations and response to the user's actions.
The related entities were displayed at all times around each event, in red text colour. The spiral was not explicitly drawn anymore. It was possible to pan the view to look at different places.

It had serious usability problems caused by the related entities overlapping one another. It was also difficult to discern the spiral shape since it wasn't explicitly drawn.

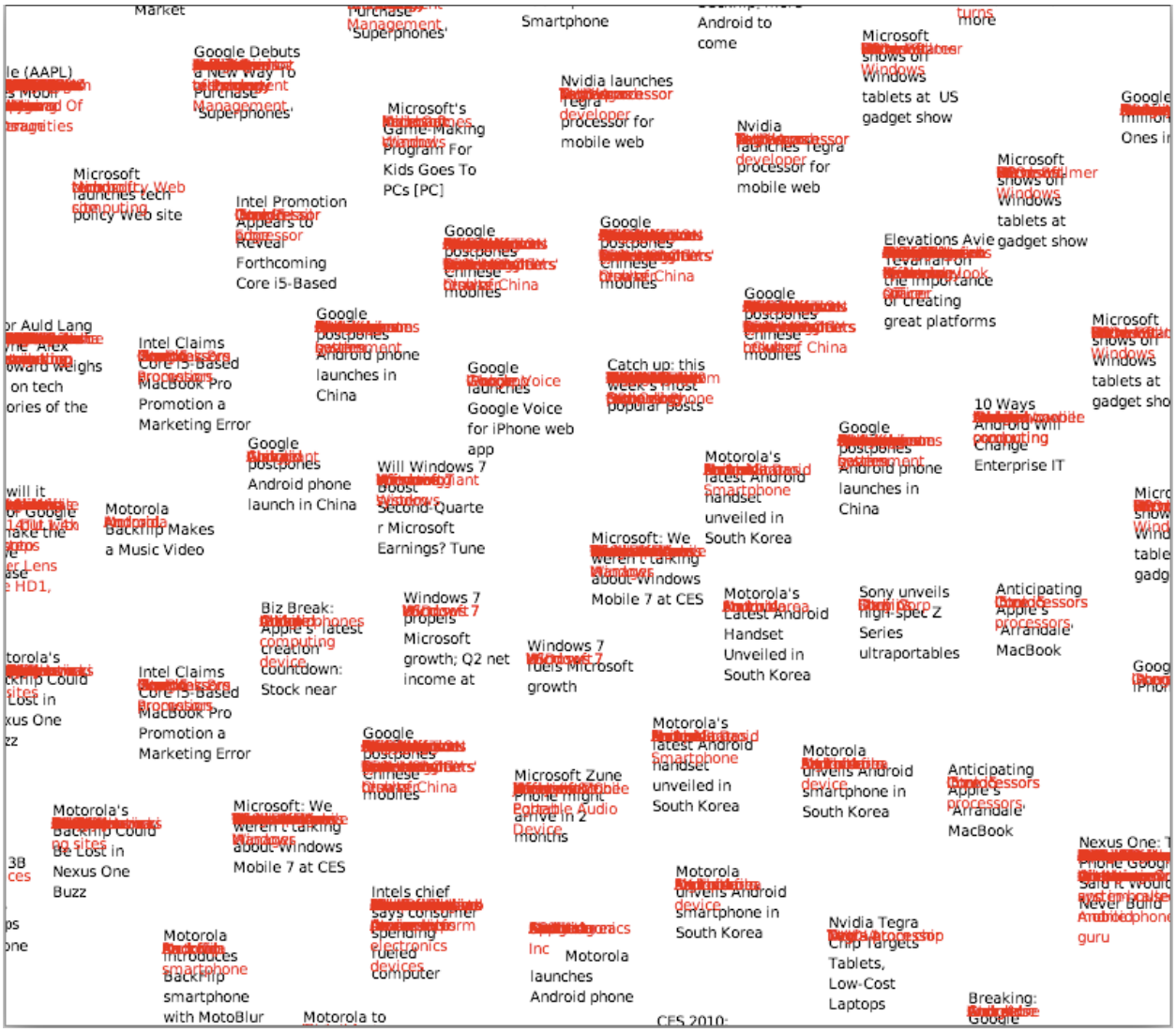


Figure 17: The second prototype

In this version, Twitter support was implemented, which led to the creation of the Data Source modules.

Data source modules are used to obtain data from the different servers - Recorded Future or Twitter - or provide saved results from queries for demonstration purposes when no Internet connection is available (this is also useful during development to quickly test changes since it is faster than performing the query, discards any possibility of connection errors and the visualisation can be compared against expectations and previous runs). These modules perform the connection to the specified server, authenticate and request data. Once the data is received, they communicate it to the rest of the program, and if possible, keep on periodically querying the server for new results.
They convert the results into instances of a class that stores the data irrespective of which server is being used. Thus the rest of the program is agnostic to which data source is actually providing the results.
The filtering system was not completely implemented, but its beginnings were set in place.

8.3 Third prototype

Related entities were only shown when clicking on each event (Figure 18). The most popular ones were shown in the topmost part of the program, though this feature was removed once the filtering system was fully implemented in the definitive version.

The main motivation for removing this feature was that the number of related entities could be very big and it was impractical to show only a few ones as possible filters. They also added clutter to the screen.

The spiral was drawn explicitly again. Events now had a different font colour depending on how relevant they were. The events had no background, which made them hard to read sometimes.

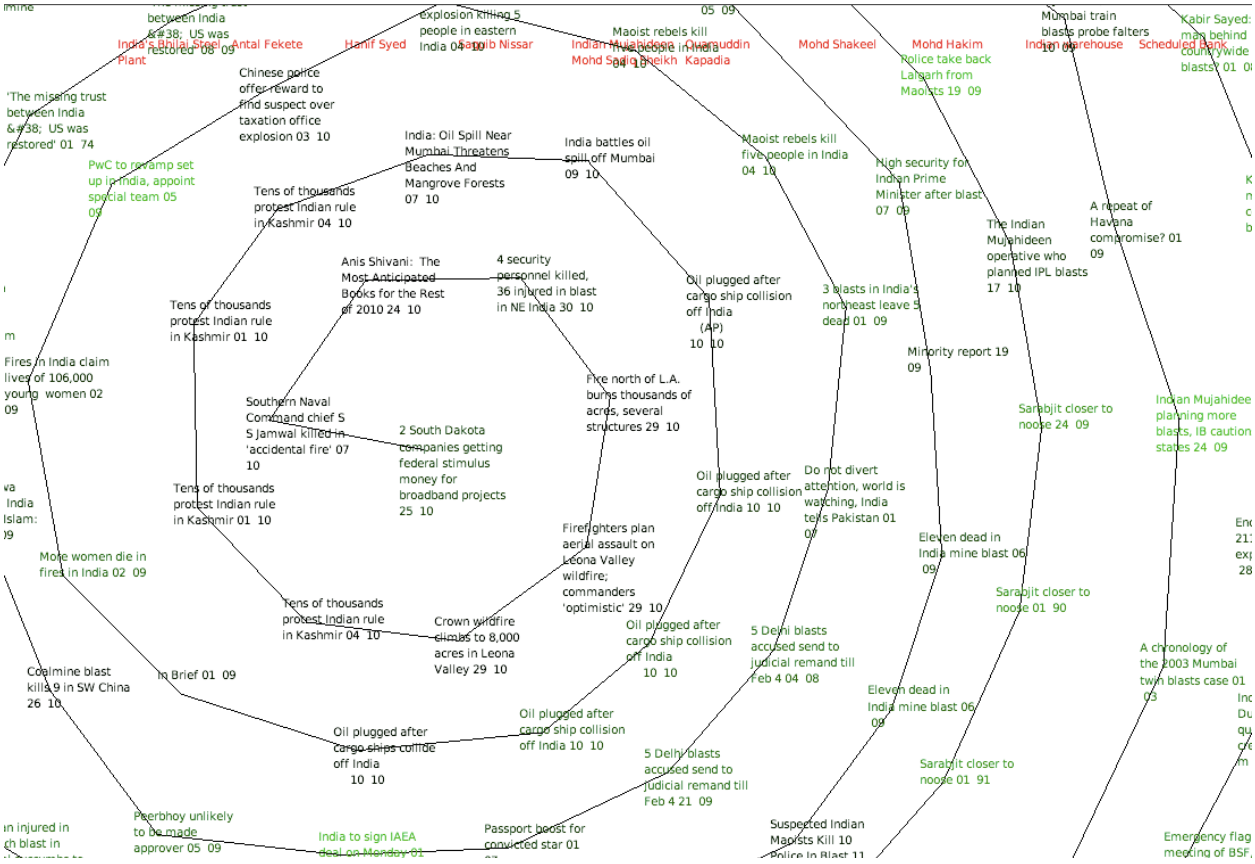


Figure 18: The third prototype

8.4 Definitive structure of the Spiral Diagram sketch

For its final version, the structure of the Spiral Diagram Processing sketch was divided into the following parts:

- Filters
- Visualiser
- Animator

8.4.1 Filters

Filters are used to only show the data which meets the criteria the user has specified.

Programmatically, they can be set to test if a particular item meets a criterion in either specific fields or every field. The user, however, only has access to either test if an entity is present in an event or test if a term is present in all the event's fields (including entities). This was found to be useful enough for the project.

8.4.2 Visualiser

The visualiser takes all the items that met with the filters' conditions and then proceeds to lay them out according to the following rules:
- The most recent item is placed in the centre of the spiral. The others are laid out in chronological order towards the outer loops of the spiral.
- Every item initially has the same size. Each item must be separated from its predecessor with a distance two times this size. The spiral's coordinates are generated according to the Archimedean Spiral equations described in the Design section; thus the visualiser increases the angle until the distance condition is met.
- Focusing on one item will cause it to grow in size in order to display its details and its related entities will become visible. Related entities are placed at a short distance from the item; if the item is displaced, the entities are also displaced.

8.4.3 Animator

The animator has the task of performing transitions, gradually displacing items to their proper places.
Transitions were chosen as opposed to just placing items in their final locations because it has been demonstrated that transitions help users comprehend the changes in programs interfaces (Tidwell, 2005).
The physics engine is also considered as part of the animation module since it provides response to user interaction and smoothen transitions: Every item is attached to an "anchor" that is the location determined by the visualiser for the item. It is joined to this anchor by a simulated spring, but it is allowed to roam freely a short distance around the anchor. Every sub-item is joined to its parent item by a simulated spring, yet it is repelled from its parent by a simulated magnetic force. Similarly, every sub-item is repelled from other sub-items that are joined to its parent item; no springs are simulated between entities because they are not conceptually "attached" among themselves. Finally, the mouse cursor has a faint attraction force to all items and sub-items that come close to it.
The resulting interaction between all these simulated forces is that initially all the items are placed in a spiral shape and float around lazily without significantly distorting the shape. When there are sub-items on display, these float around the parent item at a short distance and do not overlap one another. When the mouse cursor comes close to any item or sub-item, they react by attempting to get closer to the cursor.

8.4.4 The final version of the program

In the final version of the program, the user is shown the diagram with the events available in that moment, drawn as coloured boxes of equal sizes. The decision. In this view, items represent relationships between entities. These entities are however not visible initially.

Depending on the data source, these items are either events coming from the Recorded Future database, or Tweets from Twitter.
Only a fragment of the item's text is displayed as shown in Figure 19:

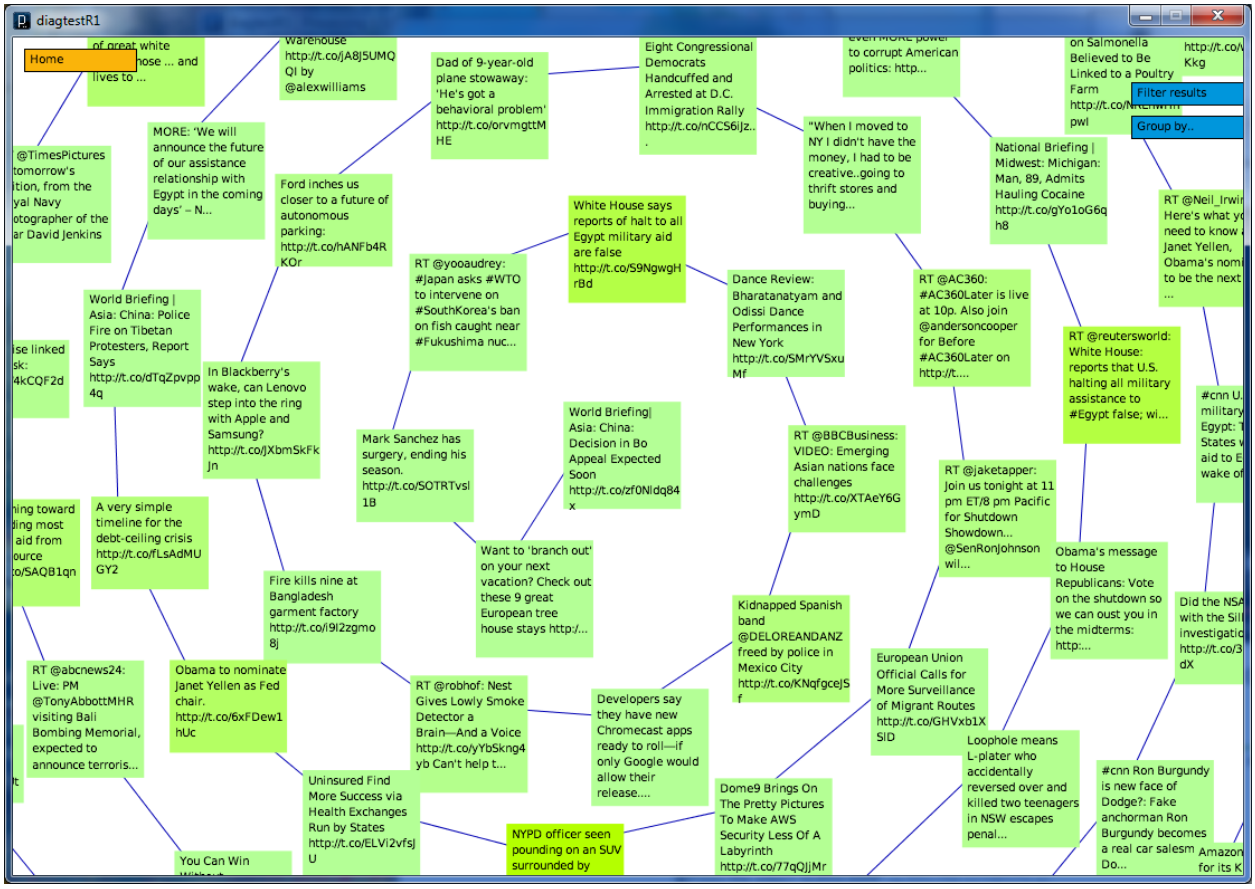


Figure 19: The initial state of the Spiral Layout

If the number of items is such that the spiral does not fit inside the window, it is possible to “pan” the view by clicking and dragging anywhere on the interface (Figure 20).

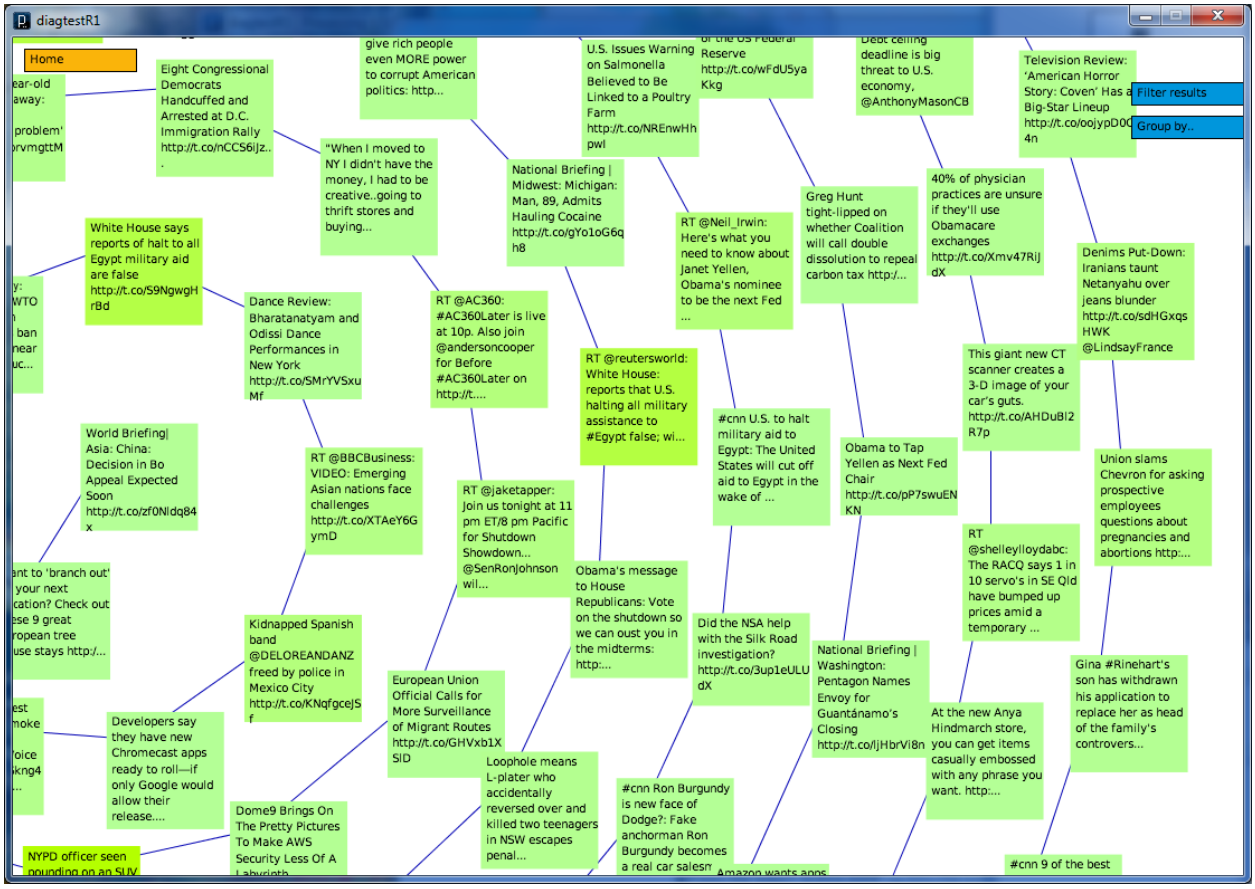


Figure 20: Panning reveals more events that were initially off-screen

The intensity of the item's hue is determined by the relevancy assigned to the item. The brighter the hue, the more relevant the item is considered to be.

Clicking an item will expand it, revealing more details. When the item is expanded, its text will be shown in its entirety. The sub-items will pop-up and hover close to the parent item; these sub-items represent entities which are related to one another vía the main item (Figure 21). The rest of the interface is darkened to help the user concentrate on the chosen item. The user can still interact with the darkened interface.

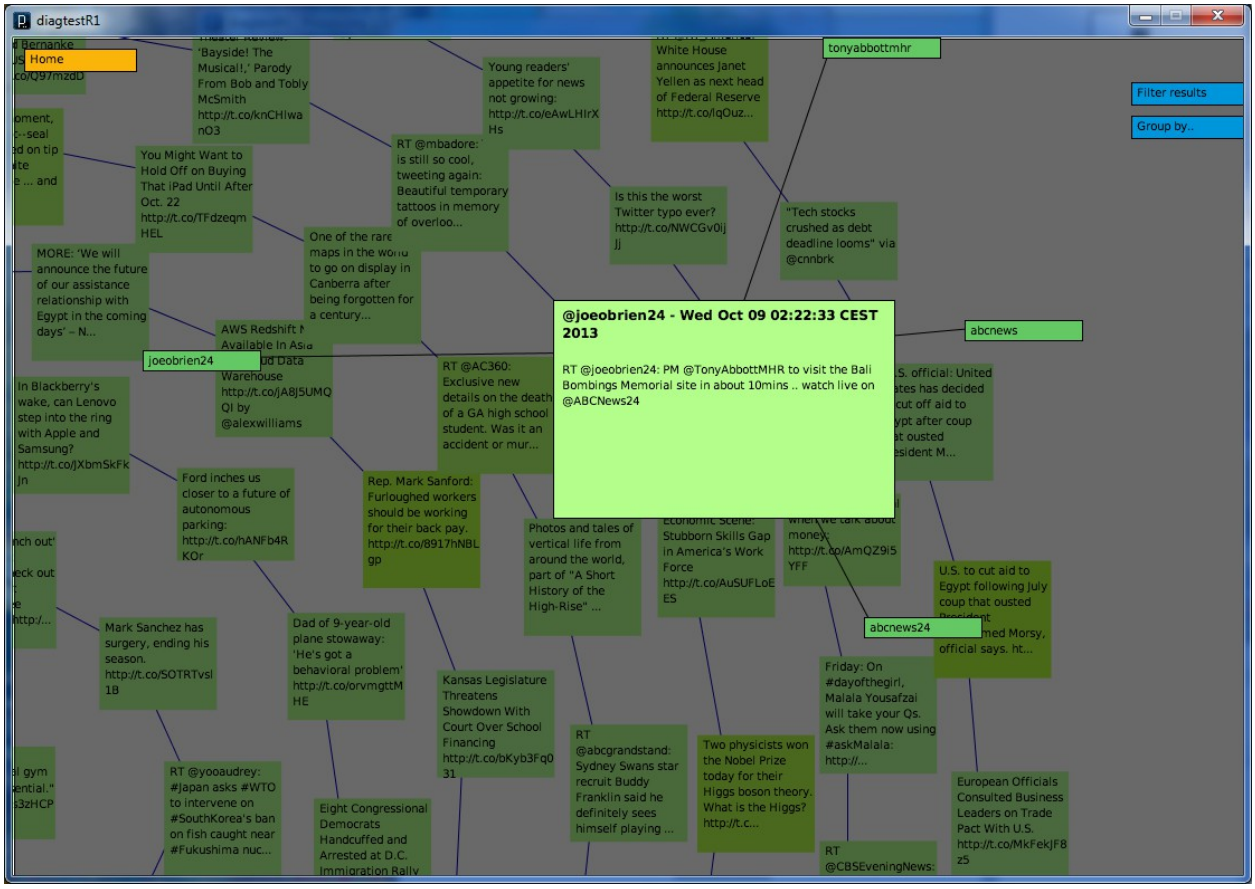


Figure 21: Clicking on an event will display its related entities and other details

The user can choose another item, click any blank space on the interface to reduce the focused item and hide its sub-items, or click a sub-item to filter, e.g. only display items that contain this sub-item. In items which contain a link, even though the link is not in itself a related entity, a button to open the associated link appears. This button does not float around but instead is attached to the expanded item. It also shares the item's colour, to avoid confusing it with a related entity.

If the user chooses another item, the previously chosen item will reduce its size and hide its details and sub-items. The chosen item will then expand and show its details in the same manner the previously chosen item did.

If the user clicks a sub-item - a related entity, a filter will be activated: a new spiral containing only the items which contain the related entity will be created. The old spiral will be hidden and close to the top border of the window, a tab with the related entity's name will appear (Figure 22).

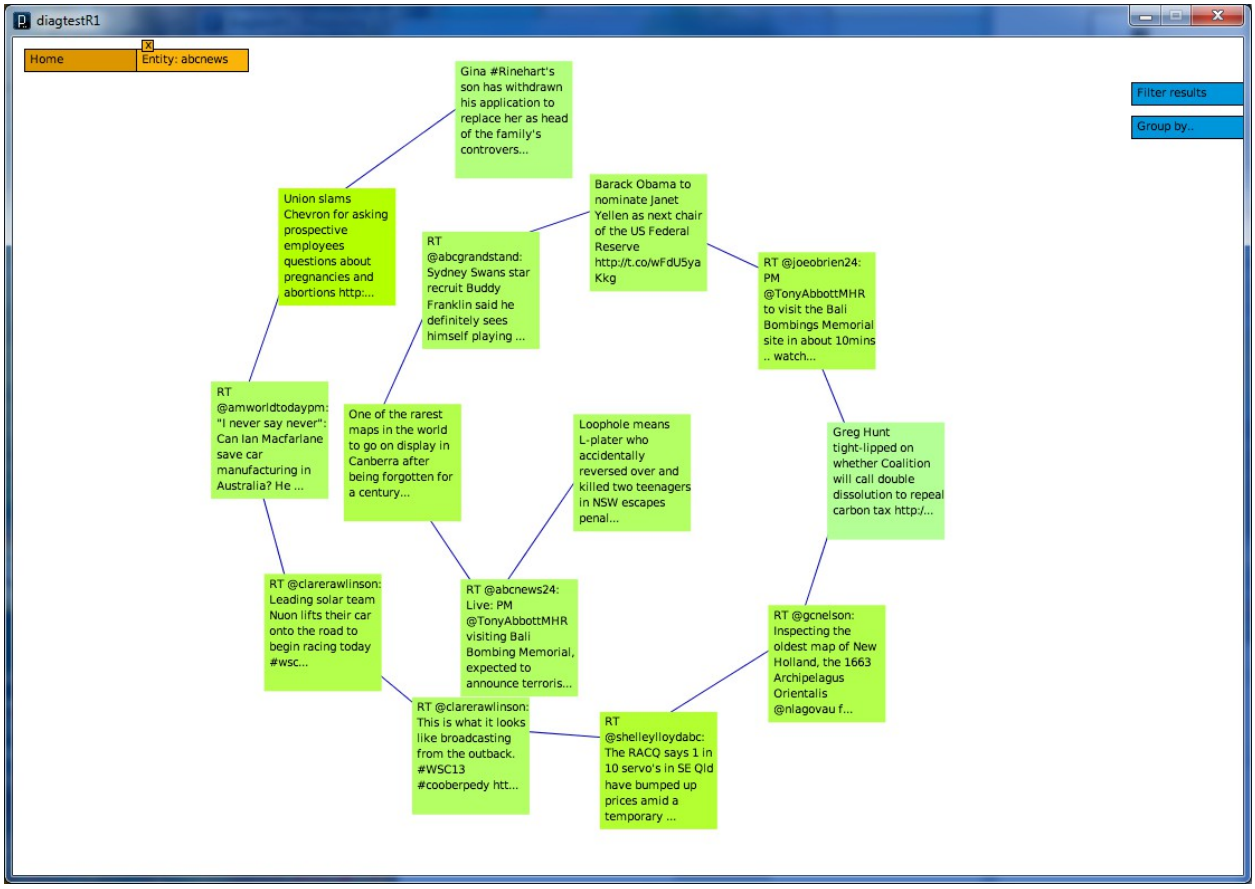


Figure 22: After filtering for an entity, the number of items can be greatly reduced

The user can select as many subsequent filters as she needs in order to get results as specific as she needs to. This can be done by repeatedly clicking on related entities; each one will create a tab and apply the corresponding effect (Figure 23).

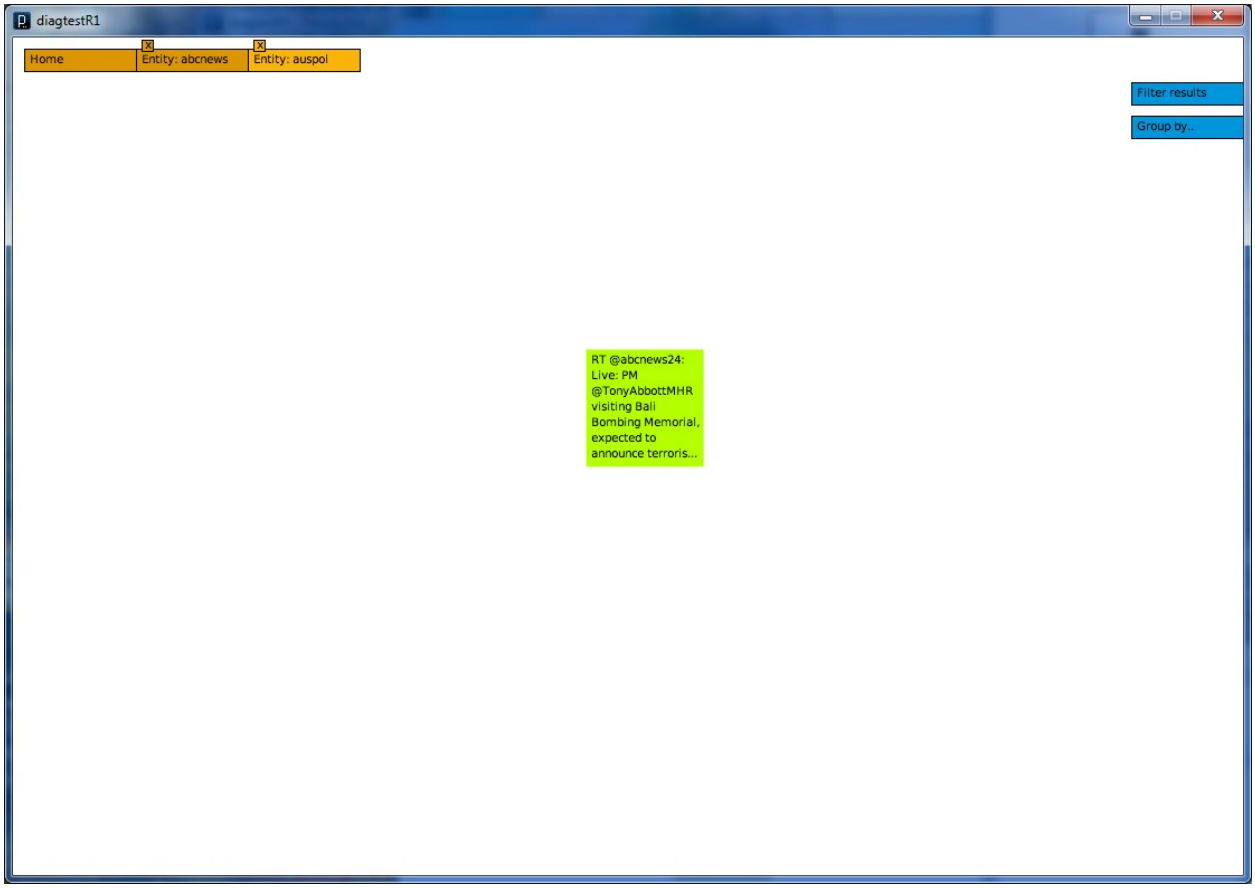


Figure 23: Further filtering allows the user to more precisely locate what she is looking for

By clicking on each tab it is possible to go back and forth a succession of filters. Each tab has a "close" button that the user can click to remove a filter.

9 Feedback

9.1 Presentation at Recorded Future

I displayed the project in a brief presentation to the Recorded Future staff in August 2010.
The process as well as the decisions leading to the final prototype (all that is described in the present report) were presented in about 20 minutes and then the Recorded Future staff asked questions and gave me suggestions.

One such suggestion was a different interaction possibility: instead of panning the spiral, "zooming" the spiral in and out, which would be akin to displacing oneself inside a 3D helix. As an analogy, it could be like an elevator in a glass tube that is the center of a spiral staircase, moving upwards and downwards when the user chooses to.

9.2 Assessment by informal testing

Informal testing was conducted with a small number of users for qualitative feedback: 10 out of which 4 responded. Due to time constraints, no further efforts were made to contact additional participants.

The first participant is a student at a Computer Science program in another university; logically he has a good knowledge of Social Media and experience using computers. This participant uses Twitter often, posting on average 4 messages per day.

Two of the participants are fellow students from the Interaction Design Master's Programme at Chalmers, so they have good knowledge of Social Media and experience with using computers. They are however not avid users of Social Media: they possess Twitter accounts but rarely post messages (less than 10 messages a month).
They are also able to criticise the layout from the Interaction Design theory's point of view.

The last participant is a scientist from an unrelated field. He has a good knowledge of using computers, has heard about Social Media, but is not a user himself (he has no experience with Twitter nor other Social Media networks).

The participants were sent the program executable, which connected to a Twitter timeline consisting of mostly tweets from news and science outlets (CNN, BBC, New York Times, NASA and the European Space Agency among others) and instructed to keep the program running in the background for some time until many tweets had been gathered - approximately 10 minutes.

After the screen had been populated, they were told to follow the project's vision of a typical Twitter use case involving browsing without a specific goal in mind (just looking for something that caught their interest) and searching for all the information regarding a specific topic. Afterwards they were asked to respond to a questionnaire about the interaction and general design of the program as well as reporting any technical issues they may have found (bugs).

The questionnaire as well as the use case can be found in the appendix of this document.

9.2.1 Feedback from informal testing

Overall, the feedback was positive: the testers found the spiral shape interesting and appealing, the interface self-explanatory. They mentioned that the project was responsive even when showing a great number of items and found it convenient that no matter where they were looking, they would always notice when there were new messages arriving because of the animation that occurs when rearranging these new messages.
They noted that after many spiral loops, the separation between two of them could occasionally be insufficient. However, after the arrival of new items, the problem would generally be corrected. The very first participant made a suggestion that was adopted for the subsequent tests: obscuring the background of the application when an item is selected. This permits the user to better concentrate when selecting items.

One important criticism was that navigating by panning around could become tiresome in the outer loops of the spiral. A solution to this problem could be implementing a control to automatically jump from one item to the next one. It was considered, but due to time constraints, it wasn't possible to include it.

10 Discussion

While it is mentioned in the background, that the concept of Visualisation extends beyond graphical representations to other kinds of representation (auditory, tactile, etc.), it was decided early on that this project should consist of a graphical representation.
The novelty of the Event-Relationship data was a challenge in the beginning, and adding an extra challenge of designing a non-graphical representation would probably cause the project to extend beyond the available time, because the author is entirely unfamiliar with non-graphical representations (e.g. for the visually impaired).

In general, the methods, methodologies and framework chosen worked well. A description of the performance of each of them follows.

10.1 Brainstorming

Brainstorming was used mainly in two instances - proposing ideas for the "general" visualisation technique, and proposing ideas for the visualisation technique "tailored" to the data.

It was not so useful for the "general" visualisation technique because, since it is supposed to be a general model, existing techniques are favoured over new, unproven ones. In this case perhaps it is faster to simply choose from the ones available in visualisation literature rather than brainstorm (there is not much space for creativity in this step)

However, brainstorming proved very useful for proposing ideas for the visualisation technique tailored to the data (the last step in Benjamin Fry's methodology), because especially when one is unfamiliar with the data, many good ideas are generated this way, even if not all are applicable.

10.2 Benjamin Fry's methodology for creating interactive visualisations

Benjamin Fry mentions when proposing the methodology, that it should not be taken as a strict methodology but rather one should apply only the steps that are relevant to one's work.

In the case of this project, perhaps due to the novelty of the data type, almost all the steps proved to be necessary and were successfully followed. One deviation from this method would be that the mining step is not applied. The pattern detection is up to the human to perform.

In general I found this methodology to be quite comprehensive and flexible at the same time. I will very likely apply it to any visualisations I design in the future.

10.3 Processing

Processing has many features which are especially convenient when making prototypes of visualisations: it compiles rapidly, its syntax (Java) is similar to other programming languages and its "quick and dirty" philosophy as well as the variety of functions that can be immediately used encourages experimentation.

On the other hand, the same “quick and dirty” philosophy can encourage people to write code that is quite hard to maintain and expand, when a prototype turns out to be something that is “almost” good enough to become something more “serious”.
One important disadvantage, at least in the standard Processing libraries, is the lack of functionality to create interfaces that are closer to native applications, which is easily done in other languages. For example, in Processing there are easy ways to detect general clicking and dragging actions. But if one desires to be able to enter text into a standard text field, it is necessary to create something that looks and reacts like a standard text field. This is much more work than appears at first glance, because text fields are very common in programs and people interact with them in different ways and will likely expect it to react as a normal text-field. For instance, some people might expect to be able to drag text from another program into a pseudo-text field within a program made with Processing and expect it to appear within the text-field. This seemingly simple interaction involves many operations which would have to be implemented by the visualisation’s programmer. When using other programming environments, common objects like text fields are already fully implemented and ready to be used. The same can be said of most other common interface components.

I recommend using Processing for prototyping, especially when the usage of common user interface controls such as the one referred to in the previous paragraph do not exist or do not play a very significant role.

11 Future work

There are possibilities to explore in 4 aspects:

- Enhancements to the existing system

Due to time constraints, some interesting features were not considered for implementation:

- Filtering by a precise time interval.
Currently it is possible to group events time-wise by year and month; this resolution is not precise enough for Twitter and it may also be insufficient depending on the data received from Recorded Future. Being able to filter more precisely would help the user discover what she is looking for.

- Filtering by custom, user-set categories.
Giving the user the possibility to mark interesting events in different categories would increase the visualisation's usefulness when using it as a tool to do research. After categorising several events, the user would be able to see them separately (using the category as a filter) and infer relationships between them.

- Using the spiral as a collaborative brainstorming tool.
Ideas would be added to the spiral by the participants. After the proposal period is up, the ideas would be reordered by feasibility and categorised. These categories would act as filters, and supplemental data (addenda to specific ideas) would be possible as well.

- Displaying further relationships between events

In the real world there are events which can be identified as being the cause of another event - in other words, a relationship between events exists. While this data could not be gathered from the Recorded Future database, in the case of Twitter an example would be conversations that can occur between two or more participants.
How would this link be best displayed in the Spiral Entity-Relationship Diagram?
One proposal would be to use arrows or lines connecting two places in the spiral.

One disadvantage would be that the diagram could become cluttered if there are too many occurrences of this kind.

Also, particular to the case of Twitter, conversations can exist between a participant in one's timeline and other participants external to one's timeline. This implies that an event represented in the spiral might have a link to an event that is external to it. As this is one way of gathering novel information for the hypothetical user of the Spiral Diagram, representing these links would be useful for the user.
A proposal for one such representation would be to create a smaller spiral, sprouting from the event in the main spiral in the diagram (Figure 24). If the sub-spiral is only shown when the item is focused, cluttering would be unlikely. It is convenient however, to convey the user that a sub-spiral exists for an event even when the item is not focused.

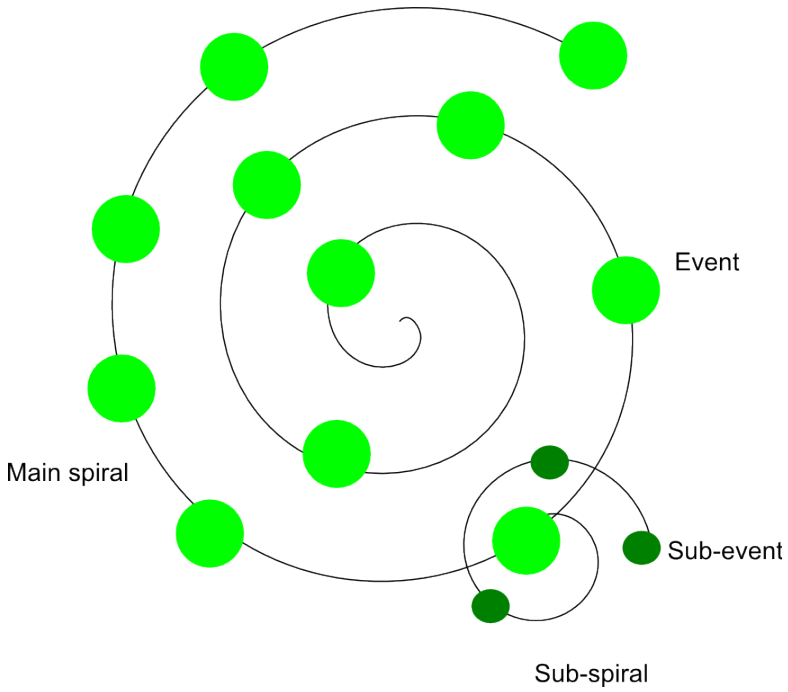


Figure 24: The sub-spiral proposal

- Representation of other kinds of data in the Spiral Diagram

It would be interesting to try to represent other kinds of data - for example a catalog - in this way, and see if it has any advantage over current representations. Borrowing the "sub-spiral" proposal from the previous bullet point, a hierarchy could be established by presenting categories as the "root" items of the sub-spirals and having the items in each category appear in their corresponding sub-spiral (Figure 25). The most popular selections from the catalog could be easily and instantly visible by representing it as "relevancy" in the Spiral Diagram (i.e. a brighter hue depending on how relevant an item).

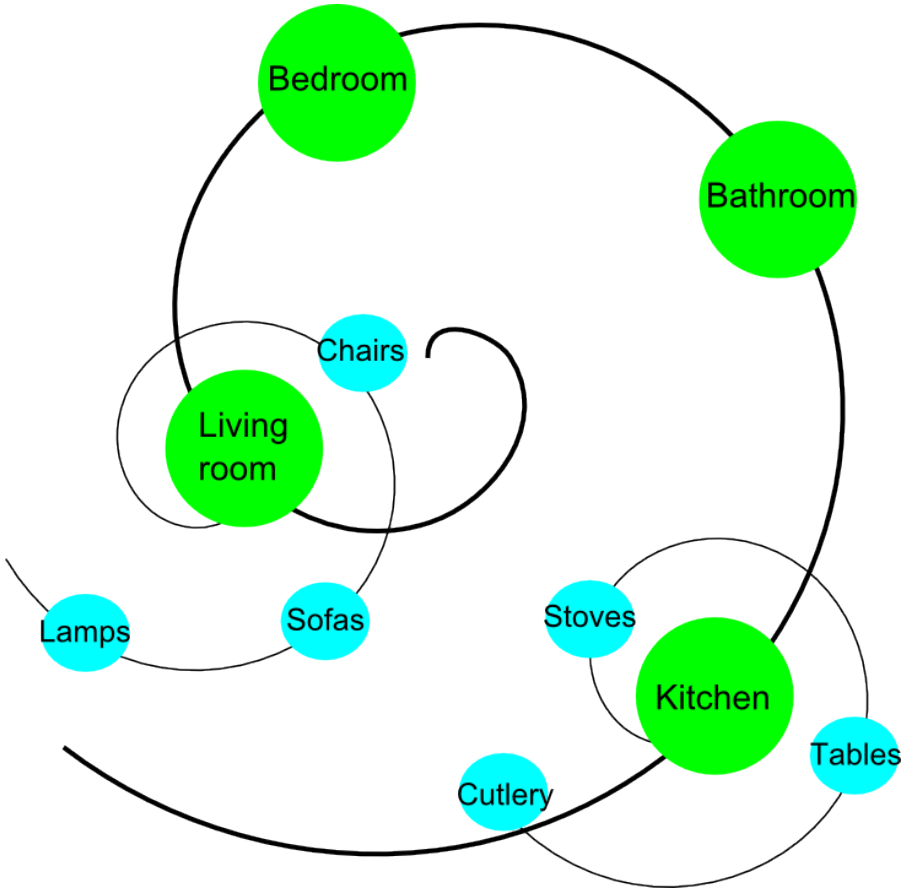


Figure 25: The Spiral Layout applied to a catalog

- Mobile platforms

Mobile platforms pose additional challenges due to their reduced screen estate and their transient usage pattern (the user is generally on the move, making continuous interaction over long periods unlikely). Without considering the disadvantages that a graphics-rich representation has with respect to battery use, would this representation have advantages over a more traditional one (such as the default Twitter interface)? And considering that touchscreen interaction can be richer than a traditional mouse-and-keyboard interaction with the usage of multi-touch screens, how can this best be best utilised?

12 Conclusion

In this project, a situation is first observed: there is too much data about current events being generated on the Internet even if one only uses a couple of information sources. In the project's case, the objective is to design something that will allow the user to make the most of the query data available from Recorded Future; in other words, the research question is "**How can the Recorded Future entity-relationship data be represented conveniently?**".
Recorded Future is a service that aggregates, analyses, organises and classifies news articles. Twitter, a popular social media network oriented to writing small messages, is also considered due to the similarity of the data format between both services. Using automated means to quickly discard information that is not useful is complicated. Moreover, there is also the possibility that the user does not have a specific objective in mind and desires to know in general what is happening.
In the beginning, neither the methodology nor the platform necessary to develop this project had been decided. What had been decided is that the solution would be a graphical visualisation, and that it would be a prototype of exploratory nature, which are the project's limitations.

Literature search was conducted continuously to find how to best deal with problems and situations encountered and to find inspiration as well.

This project pertains the branch of Visualisation known as Information Visualisation since the data being represented is generated by news articles around the world as well as messages in certain social media networks. For this reason, techniques from the Information Visualisation will be used to work with the data.

The visualisation structure chosen for the project is a graph; specifically a network graph. A network graph is comprised of vertices and edges. Vertices are an abstract representation of objects; edges are a representation of the relationships between the objects.

Following Benjamin Fry's guide for creating engaging visualisations, several layouts were proposed and considered. Several possibilities were proposed although many of them were found to have significant drawbacks in readability when populated with data from Recorded Future. Finally, one proposal in the shape of a spiral was determined to be adequate.
The motivations for implementing the spiral model are a good use of the screen's "real estate", especially when contemplating using a "details on demand" interaction for the data, smooth reordering of items when new data arrives after the existing data has been laid out, and aesthetic appeal.

After testing several frameworks for implementation, Processing was chosen due to its visualisation-oriented nature.

The project was implemented iteratively and in its final form is able to query the Recorded Future or Twitter services, receive results and display them on-screen.
These results are organised in a spiral layout, where the most recent result is placed in the center of the spiral and older items are placed as the spiral unwinds, in chronological order.

The items represent relationships between entities. Items considered more relevant relative to the rest have a more lively hue as a strategy to make them stand out. Since the number of items usually exceeds the on-screen space, the user is able to pan the view to inspect any item she desires.
Initially only relationships are displayed on-screen but the user is able to click on any to display its details, including any participating entities. She is also able to filter by a keyword as well as by clicking on an entity. The spiral rearranges existing items when new ones arrive smoothly using animation.

The implementation was shown to a limited number of people which responded with positive comments, bug reports (which were fixed) and improvement suggestions. It was also displayed to the Recorded Future staff, who came up with an extra layout suggestion (which was not possible to implement due to time constraints).

Finally as future work, it is proposed that the visualisation be extended with features, to other data sources and tested, and that the visualisation could be ported to mobile platforms since the interaction is suitable for touch screens, although it is speculated that some adaptations may be necessary.

13 Appendix

13.1 Informal test questionnaire

The project's executable was sent to 4 participants of different backgrounds and knowledge of social media, so that they could run it on their own computers along with the following instructions:

The program should be left running for about 10 minutes until it gathers many Twitter messages. Once the program is ready, please follow the user case written below and describe what you think of the program, in terms of:
- usefulness
- responsiveness
- presentation (aesthetics)
- ease of use
- suggestions for improvement / errors you notice

Twitter use case

1. *Open the program to monitor the twitter timeline (without necessarily having an objective in mind)*
2. *Leave it running for a number of minutes, letting news results accumulate*
3. *Navigate the query results: pan, click, move around.*
4. *Locate the items with the most momentum ("relevance") - either visually or by using the grouping tool.*
5. *Navigate the sub-results: pan, click, move around.*
6. *Refine the query results if the information desired has not been located: through the search box or even another group operation.*
7. *Locate the desired datum*
8. *See article on web browser*
9. *Return to program.*
10. *Decide to see another article*
11. *Decide to see a less relevant article: click a previous state in the breadcrumb trail*
12. *Repeat from step 5*
13. *When done, return to step 2*

14 References

- Card, Stuart K., Jock D. Mackinlay, and Ben Shneiderman. 1999. Readings in Information Visualization: Using Vision to Think (Interactive Technologies). Morgan Kaufmann. http://www.amazon.com/Readings-Information-Visualization-Interactive-Technologies/dp/1558605339.

- Di Battista, Giuseppe, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. 1994. "Algorithms for Drawing Graphs: An Annotated Bibliography." Computational Geometry 4 (5) (October): 235–282. doi:10.1016/0925-7721(94)00014-X. http://linkinghub.elsevier.com/retrieve/pii/092577219400014X.

- Fry, Ben. 2008. Visualizing Data: Exploring and Explaining Data with the Processing Environment. O'Reilly Media. http://www.amazon.com/Visualizing-Data-Explaining-Processing-Environment/dp/0596514557.

- Gretarsson, Brynjar, Svetlin Bostandjiev, and John O'Donovan. 2010. "WiGis: A Framework for Scalable Web-Based Interactive Graph Visualizations." Graph Drawing: 119–134. http://www.springerlink.com/index/9768Q60438977178.pdf.

- Holden, C. (2013). Pattern of Life and Temporal Signatures of Hacker Organizations. *analysisintelligence.com*. Retrieved November 09, 2013, from http://analysisintelligence.com/cyber-defense/temporal-signatures-of-hacker-organizations/

- Jackson, Tom, Sarah Wheaton, Edward Martinet, and Aaron Druck. 2010. "Tracking Twitter Traffic About the 2010 Midterm Elections." The New York Times. http://www.nytimes.com/interactive/us/politics/2010-twitter-candidates.html.

- Krikorian, R. (2013). New Tweets per second record, and how! *blog.twitter.com*. Retrieved November 09, 2013, from https://blog.twitter.com/2013/new-tweets-per-second-record-and-how

- Mackinlay, Jock D., George G. Robertson, and Robert DeLine. 1994a. "Developing Calendar Visualizers for the Information Visualizer." In Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology - UIST '94, 109–118. New York, New York, USA: ACM Press. doi:10.1145/192426.192470. http://dl.acm.org/citation.cfm?id=192426.192470.

- Mulder, Ingrid, Henk de Poot, Carla Verwij, Ruud Janssen, and Marcel Bijlsma. 2006. "An Information Overload Study." In Proceedings of the 20th Conference of the Computer-Human Interaction Special Interest Group (CHISIG) of Australia on Computer-Human Interaction: Design: Activities, Artefacts and Environments - OZCHI '06, 245. New York, New York, USA: ACM Press. doi:10.1145/1228175.1228218. http://dl.acm.org/citation.cfm?id=1228175.1228218.

- Purchase, Helen C., Robert F. Cohen, and Murray James. 1995. "Validating Graph Drawing Aesthetics" (September 20): 435–446. http://dl.acm.org/citation.cfm?id=647547.728624.

- Spence, Robert. 2007. Information Visualization: Design for Interaction (2nd Edition). Prentice Hall. http://www.amazon.com/Information-Visualization-Design-Interaction-2nd/dp/0132065509.

- Tidwell, Jenifer. 2005. Designing Interfaces: Patterns for Effective Interaction Design. O'Reilly Media. http://www.amazon.com/Designing-Interfaces-Patterns-Effective-Interaction/dp/0596008031.

- Tufte, Edward R. 2001. The Visual Display of Quantitative Information. Graphics Pr. http://www.amazon.com/The-Visual-Display-Quantitative-Information/dp/0961392142.

- Weisstein, Eric W. 2013. "Archimedean Spiral -- from Wolfram MathWorld". Wolfram Research, Inc. http://mathworld.wolfram.com/ArchimedeanSpiral.html.

- Weisstein, Eric W. 2013. "Logarithmic Spiral -- from Wolfram MathWorld". Wolfram Research, Inc. http://mathworld.wolfram.com/LogarithmicSpiral.html.

- Wickre, K. (2013). Celebrating #Twitter7. *blog.twitter.com*. Retrieved November 09, 2013, from https://blog.twitter.com/2013/celebrating-twitter7