



Linnæus University

School of Computer Science, Physics and Mathematics

Degree project

Visualization of Weather Data

Temperature Trend Visualization



*Author: Liu Jiayi
Date: 2012-08-03
Subject: Computer Science
Level: Bachelor
Course code: 2DV00E*

Abstract

Weather data are huge. Traditional visualization techniques are limited to show temperature trends. Pixel-based approaches could be used to visualize the huge amount of weather data and in process show the temperature trends.

A prototype using this approach is built to make temperature data more understandable in changing trends. It is implemented using a 2D representation and many popular interaction techniques. It is a lightweight and reusable tool to visualize temperatures.

Keywords: information visualization, pixel-based approach, temperature, weather

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Goal and criteria	2
1.3	Outline	2
2	Background and Related Work	3
2.1	Background	3
2.2	Related work	8
3	Prototype Development	12
3.1	Conceptual Design	12
3.1.1	Representation Demension Issues	12
3.1.2	Graphical User Interface	13
3.1.3	Pixel-based Block	13
3.1.4	Visual View in Limited Size	14
3.1.5	Selection	16
3.1.6	Zooming	16
3.1.7	Line Chart	19
3.1.8	Granularity Issues	19
3.1.9	Smooth Transition with Morphing	23
3.2	Implementation	24
3.2.1	Programming Language and Framework	24
3.2.2	Data Parser	25
3.2.3	Color Coding for Temperature	27
3.2.4	Buffered Image Cache	29
3.2.5	Rectangle Manager	29
3.2.6	Animation for Smooth Transition	30
4	Discussion and Conclusion	31
4.1	Discussion	31
4.2	Conclusion	32
4.3	Future work	34
	References	36
	Appendix A How to download the dataset files	40

1 Introduction

Every day there are enormous data produced. Newspapers are big information producers. As technology developed, machines generate more and more data with various sensors and networks. People play games with joysticks as input devices; suffering on the Internet, people post messages through social communities; experiments yield much information to researchers and many other examples show that data accumulate, which leads to big information bang [1] [2] [3].

At first, infographics makes information more understandable [4]. However, it is a manual approach to draw something with specified pattern into a picture which has a nice face and is clear to provide knowledge [5] [6] [7]. But it is lacking in interaction, which means that people cannot adjust data view dynamically [8]. Furthermore, the picture is hard to reuse for fresh data. When new data comes, it will be a new picture representing them [8]. Moreover, data in large scale makes problems to infographics. A picture with fixed size does not contain all data. They are huge so that many details cannot be covered [8]. This approach is not adequate to meet current needs in the age of information explosion.

Information visualization is invented to deal with new challenges to explain dynamic large scale information and to include interaction to make visualization more usable [9]. It is a subject of computer science, which visualizes abstract, non-spatial data to transfer knowledge and help people to discover their internal structure and deeply meanings [10]. It has specified algorithms to display data automatically, makes drawings regenerating easily and shows more relationships between rich data [8]. Now it is widely used in many fields, such as biology, physics, meteorology, engineering and so forth [11] [12] [13] [14].

1.1 Motivation

Weather is a classic and popular topic. Rainy days annoy people while they are happy on sunny days. Weather forecasts are given on daily bars. Then we know when to put on our coats and when to embrace the sea with swimsuits. Moreover, the data that we will visualize are stored for later analysis.

Everything is prepared to realize the visualization for weather: data sources are provided. Due to advances in technology, there are large volumes of weather data online and open to public. European Climate Assessment & Dataset [16] is a place to download weather dataset (see also appendix A). In this project, we visualize mean temperatures which are a kind of weather data. Typically temperature dataset includes point pairs of (date, temperature value). When viewing the point pairs as (date, value), we could visualize various kinds of weather data like humidity and atmosphere pressure.

Many ideas and theories are available and useful and beautiful tools in-

spire us [15]. Many people have been devoting themselves into visualization study because of the explosion of information. Information visualization theory is built and improved. We have discussed them in the Sections 1.3 Background and Section 1.4 Related work. Computer science, as a tool science, has been becoming stronger and stronger since programming languages are invented [17]. People communicate with computers in those languages to enjoy the high-speed calculating. Before computer was invented, people figured out a function plot with several points. The points are picked from the collection of all ordered pairs $(x, f(x))$ and they are connected via lines directly in turn to form an approximate graph. Nowadays, we can tell computer to collect thousands of points. Then we get a really nice graph, which shows details much more than by manual method.

1.2 Goal and criteria

A prototype to visualize temperature data should be developed to help people, especially weather analyst, to understand temperature trends for one specified place conveniently. Firstly, an adequate approach should be found. Conceptual design is the product. The design includes how to display temperature data and what interactive actions are supported. Secondly, the design should be realized. The final product is a prototype of computer application to visualize temperature data.

1.3 Outline

This report is divided into 4 chapters, summerized here:

Chapter 1 gives a brief introduction to information exploration and information visualization. It also relates the motivation that we do the project of temperature visualization and sets the final aims. Chapter 2 is about what knowledge we apply. Some interesting idea and research work related to temperature visualization are contained. Chapter 3 discusses how our prototype is designed and how to implement the design. It is the main text of this report. In Chapter 4 you will learn about our result. The prototype has some drawbacks and some ideas is provided for future work.

2 Background and Related Work

In this chapter, related knowledge is researched. There are several ideas to visualize weather data inspiring this project. Then some theories for information visualization are discussed. At the end of this chapter, a few related works are described.

2.1 Background

There are several cases from our lives for weather visualization. Mitchell Whitelaw [18] published an idea on his blog. He designed weather bracelet, a kind of jewelry representing 3D printed temperature data. The bracelet consists of many house-shaped slices. The slice height is based on maximum and minimum temperature data from a single day. They lap over each other and make a circle.

Fernanda Viegas and Martin Wattenberg [19] show us the colors of a year also in a circle. They processed lots of photos and use a specified algorithm to compress a world into color streams. The visualization says that spring is the most colorful season and we can also get the color trends from it.

There is another nice case for weather visualization in design. It is the Dexia Tower in Brussels, which represents temperature changing with lights in different colors (see Figure 2.1). The idea is from Space Canon [20]. The light color depends on tomorrow's temperature compared to the monthly average. Warm color means the temperature is higher than the average. In contrary, cold color points out it lower than the average.

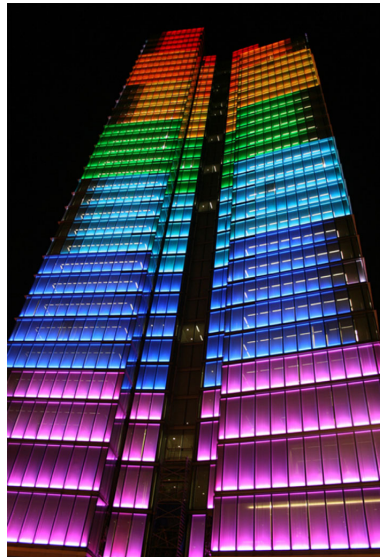


Figure 2.1: Weather tower: the Dexia tower in Brussels [7]

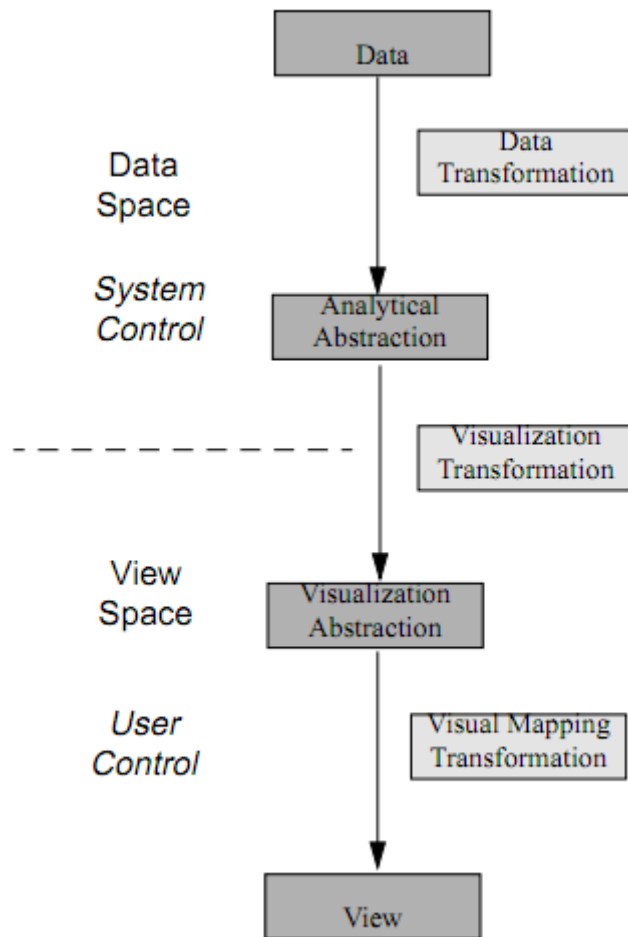


Figure 2.2: Information visualization pipeline model [21]

There are two parts of back end and front end. The back end receives raw data and takes data transformation via a special data structure to form analytical abstract data. View through user control does visual mapping transformation from visualization abstraction to present data. Visualization transformation is to connect back and front end. It is important step for information visualization.

Basic theories for information visualization are necessary. The information visualization reference model is one of the most successful theories, which is also called the data state model developed by Ed Chi [21]. He discussed the visualization pipeline of Stuart Card [22] and improved the model with a network (see Figure 2.2). The network has nodes and edges. Each node refers to a data state while a directed edge between nodes indicates that there is an operator to transform data from a state to the next. The procedure from raw data to view is clear in the network.

There are several approaches to visualize information. Icon-based, axis-based and pixel-based approaches are classical [10]. The icon-based approach is to use simple shapes to represent discrete data and get them together to form a picture. Chernoff face [23] is a wonderful example to explain icon-based approach. Eyes in different sizes can represent different values; face color can also show different states of an attribute.

The axis-based approach is to map attribute values into space points that measured by axes. Chart, which contains symbols like strips, lines or slices to render data, is popular in this approach [24].

The pixel-based approach is to map values into recognized pixel blocks. It can be viewed as a simplified icon-based approach because a pixel is a shape of dot. But they are different, for shapes represent values in icon-based approach while colors represent values in pixel-based approach. Heat map, as an instance, is a graphical representation of data with a matrix containing individual values in specified color-coding [25].

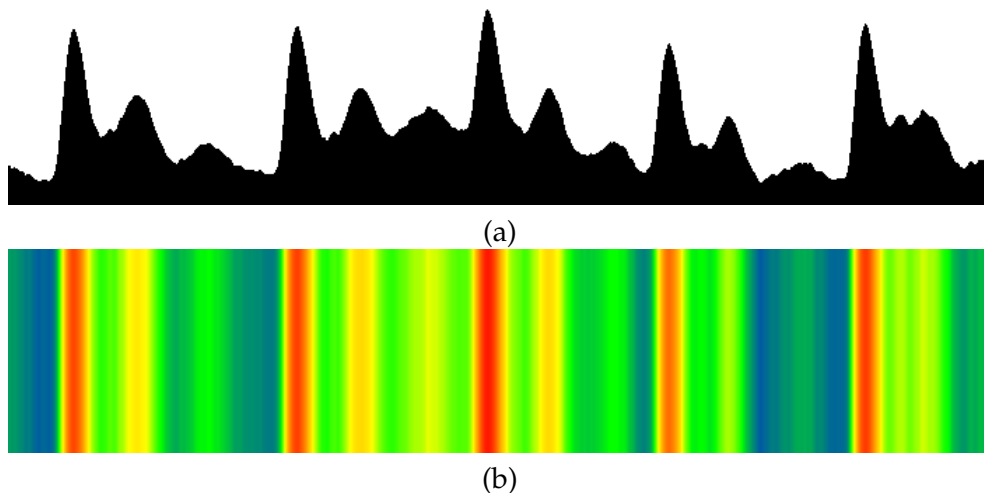


Figure 2.3: The same data are represented through two different approaches. (a) pixel-based approach and (b) axis-based approach.

In addition, there is an example to compare axis-based and pixel-based approach (see Figure 2.3). In (a), we can easily find out the function graph is

periodic and every cycle contains three peaks roughly. However, if searching for the lowest areas of the graph, we need to compare an area to another. Therefore, it takes a little time. In (b) we just, by contrast, check the blue area to get the lowest parts. But it is not easy to grasp the periodicity in (b).

Information visualization is a disciplinary subject including human-computer interaction. There are two principles of overview-plus-detail and context-plus-focus to guide designing for visualization transformation in the reference model [10]. Overview-plus-detail principle states a tool should provide both overview and detail. Overview presents a high-level view of data and detail is a dynamic view where user decides a displaying range from the overview [26]. In contrast, context-plus-focus principle states that there is only one global view and some focused regions which are selected by user in greater detail are embedded in the global view [10]. In a limited view size, distortion is often used to highlight the focuses in context [27]. It is applied in fisheye view, which magnifies interesting point with a continuous decreasing in magnification toward the edges [28].

Many interaction techniques, such as reconfiguring, encoding, filtering, abstracting, elaborating, connecting, exploring, selecting, smooth transition and so on, play important roles in tools to make information visualization more effective [29].

Reconfiguring is to give user a different arrangement to represent data for user (see Figure 2.4 (a)) [29]. For instance, some points, at first drawn in a line, are displayed in "L" shape. The layout alters from a line to "L" shape.

Encoding is to offer a different representation to user [29]. There is a common example to show encoding. A number can be encoded in different ways. Arabic numerals are often used to represent a number, which belongs to icon-based approach. In pixel-based approach, a dot can represent one and dots as many as a number value is can visualize the number.

Items are shown conditionally to user via filtering [29]. This interaction is to reduce visual objects in a view to make user's searching for interesting points clearer [30]. Google search engine is good at filtering. There are countless web pages on the Internet and they make a messy colossal network. However, Google collects data and invents Page rank [31] to sort target contents. Then it just provides an input box for user to filter data by some keywords to pick important information.

Abstracting is to show users less detail while elaborating is to increase more detail (see Figure 2.4 (b)) [29]. Semantic zooming includes those two interactions [32]. When user zooms out on a view, there will be an overview to display things in small size. Zooming in on the view, user will see things get bigger and their semantic details are also shown.

Connecting is to highlight related items to user (see Figure 2.4 (c)) [29]. There are three scenarios here. One is that there are some perspectives to represent same dataset. If user operates on some interesting point in one

perspective, the points in the other perspectives referring to the operated point should be highlighted. In another scenario, a visualization application runs on two or more computers which connected via networks, especially for meetings. User's operating on a point on one computer should be reflected on the other computer screens. The previous scenarios are for same item in different views. And the last scenario is for related items in a same view. For example, there is a dog, a cat, a tiger, a monkey and a fish drawn on screen. When user selects cat for highlighting, the tiger can be highlighted also because it is a cat-like animal so that the tiger is related to the cat.

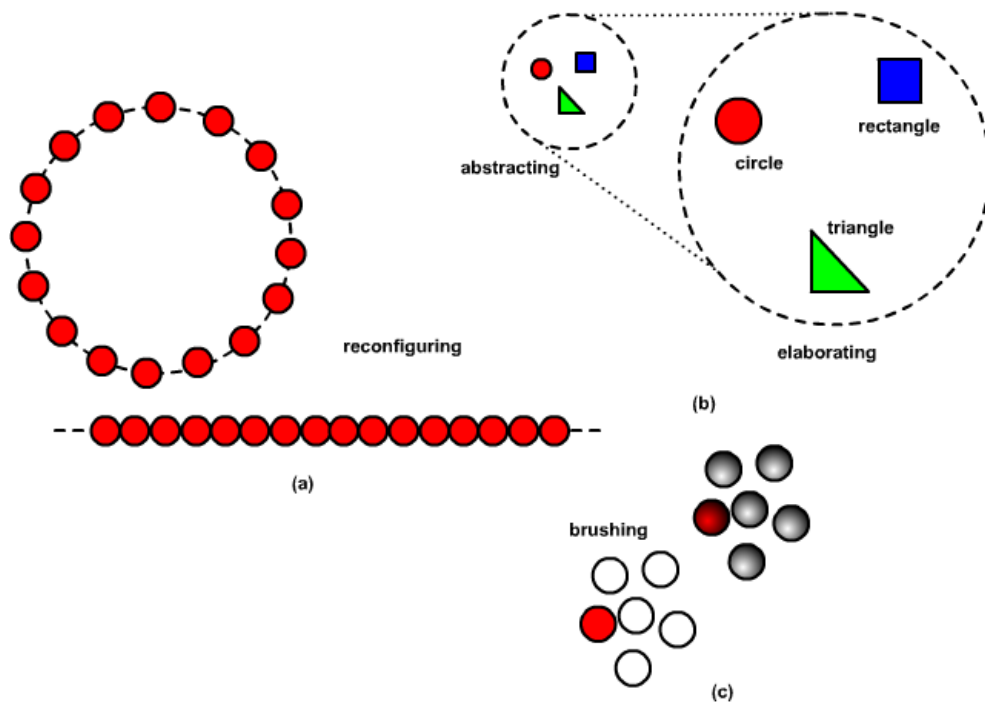


Figure 2.4: Reconfiguring, abstracting, elaborating and connecting examples. 16 elements changes their arrangement from in circle to in line, which shows reconfiguring in (a); (b) shows semantic zooming concerned abstracting and elaborating; there are two views of perspective and front view in (c) and brushing, a technique of connecting, is here to highlight the red circle in the front view meanwhile the red sphere is selected in the perspective view.

Exploring is to support discovering something else [29]. Navigation is the main technique to explore data [10]. It includes panning and zooming. Google Maps [33] is taken for an example. World map is too big to be displayed in detail through a window with limited size. Panning is realized by dragging mouse to move a view window in the global view wall. Thus user can explore everywhere in the data. When user wants to find two places adja-

cent to each other, they may be shown in a window at the same point because of the lack of resolution ratio. Then zooming is needed to distinguish the two neighborhood places via magnifying the map.

Selecting is to mark something as interesting [29]. User can select single visual object or multiple objects by mouse clicking and key stroked. And there is often a bounding box to highlight the selected objects.

Smooth transition is to make a view's changing not sharp [34]. User will lose his focus on some interesting point if there is a sudden moving or re-configuring. Therefore, the changing should be smooth. Morphing, which is used from the film *Willow* in 1988 and from *Hunger* in 1974 with computer, is one of techniques to realize smooth transition [35]. For instance, a circle will replace a square. To do smooth transition, there can be five frames to form a morph. Square, hexagon, octagon, dodecagon, circle are in each frame in turn. And setting an interval to switch frames, user can watch an animation with smooth transition where a square changes to a circle.

Visual data are presented on screen through rendering [36]. There are two popular kinds of representations, 2D and 3D representation [8]. 2D representation is most suitable for computer screen because screen can be a plane and the representation generate 2D image drawn also in a plane. 3D representation can simulate the real world. It is useful to provide graphical environments to facilitate data analyzing and help other subjects like physics, biology and manufactory being improved. Data-tip pops up when user points at a piece of data and legend may be somewhere to explain a picture or map.

2.2 Related work

Current weather visualization tools are various. Many of them are map-based. Weather data are often presented on a specified map with a gradient color table. Thus those tools are developed in pixel-based or icon-based approach for visualization. And they show data at a time point. Wind Map (see Figure 2.5) [37] is an application to present wind conditions on US land at a latest time. Its visualization based on US map is to draw white lines whose density means wind speed. And slope of white line and animation of drawing the line indicate wind blowing direction at a place.

Since map-based tools can only present data at one time point, developers use animation to display whole history of weather condition on specified map. An application showing wind condition history is US Wind Patterns (see Figure 2.6) [38]. It is also based on US map. And wind condition at a weather station is represented by circle. The size of circle refers to the speed of wind. There is a line inside circle pointing out the direction of wind. In addition, the color of circle shows current temperature. It is really nice to visualize 72 hours history of wind motion in an animation. However, overlapping makes the visualization messy.

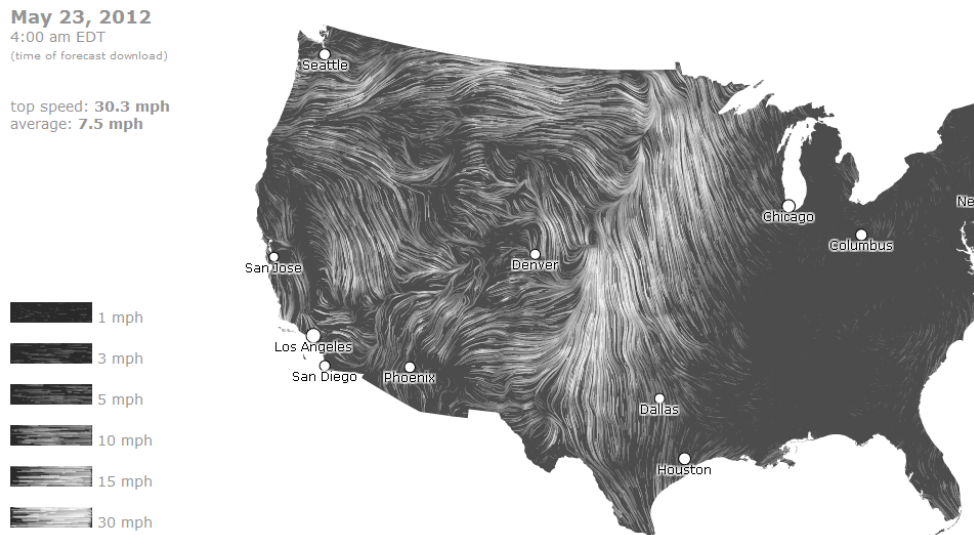


Figure 2.5: Wind Map. The screenshot of Wind Map shows wind conditions in U.S.A

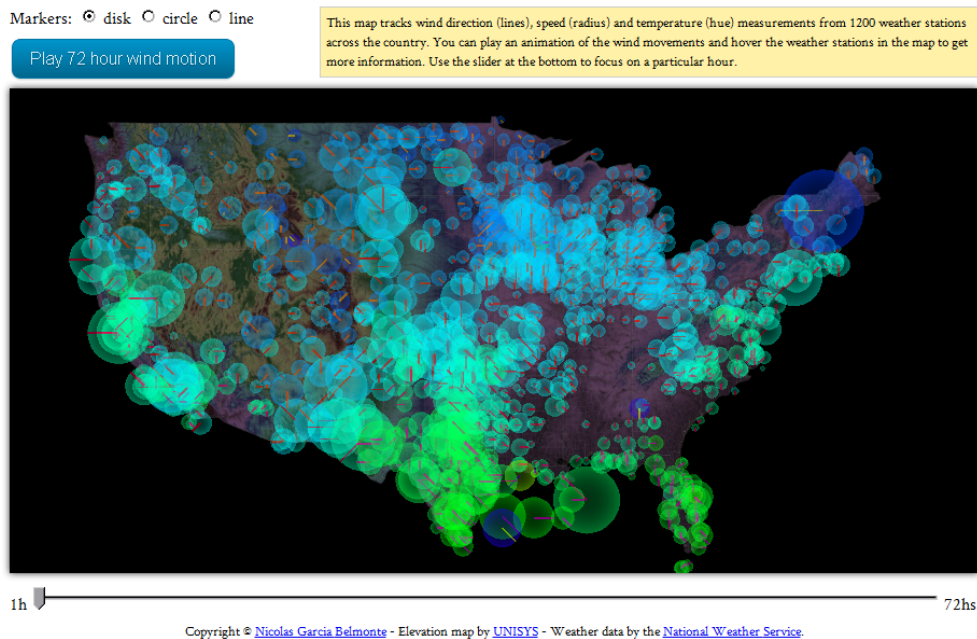


Figure 2.6: US Wind Patterns. The screenshot shows wind condition history in U.S.A at first frame.

Weather forecast is a hot topic. In common life, people use Google search and type "weather" and a place name. Then Google return weather forecast on the top of results. The visualization is simple. Special icons represent specified weather conditions, such as sun icon means sunny, thunder icon means heavy raining with thunder and so forth.

Visualization in axis-based approach is popular to present weather history and give forecast. There is an example from CLEVER°FRANKE [39] to show last year's weather. There is a big circle divided into 356 arcs. A ray through two points, the center point of the big circle and the center one of an arc, can be viewed as an axis. Temperature data are mapped to the axes and then a nice graph is generated. The visualization is clear.

Last but not least, Weather Spark (see Figure 2.7) [40] is a very powerful tool to trace into everyday weather information and provide forecast for one specified place. It mainly use line chart to display weather data. The weather information is rich, which includes temperature, humidity, pressure, wind condition and others.

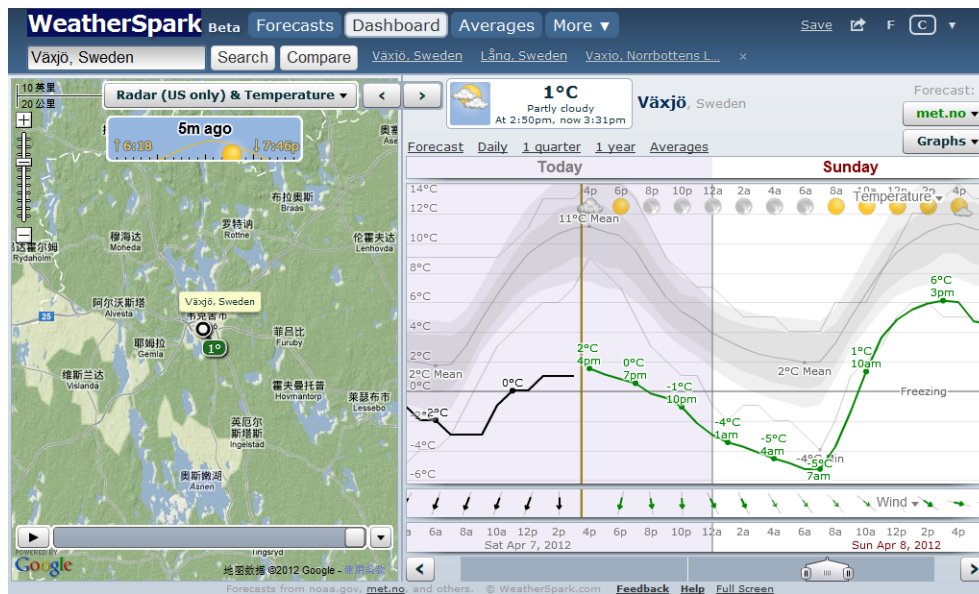


Figure 2.7: Weather Spark. The screenshot shows hourly temperature in Växjö on some day.

The present tools for weather visualization are nice. However, some problems still exist. Map-based applications present much information of different places and offer animations to figure out weather changes but they are hard to show a long-term weather history. People have limited memories in brain to record animation and analyze changes. As for line chart, it cannot provide comparison of data between many years. Groups of a year's data are

drawn as ploy lines in same coordinate. If there are many years to show, the lines overlap each other, which will confuse user.

3 Prototype Development

Our final goal is to build a prototype for temperature visualization for one place. It should offer an overview of temperature trends. Then information of date, temperature and trend needs to be displayed in one view. The development process is divided into two steps. First, the prototype need to be designed. What the prototype looks like and how it works are both discussed. Secondly, we talk about the implementation on the perspective of software engineering to discuss how to make prototyype run efficiently.

3.1 *Conceptual Design*

In this subsection, the main topic is what will be realized.

3.1.1 *Representation Demension Issues*

1D representation is considered. A number axis can hold required information. Dates can be encoded as a number, such as 0 to 1st January, 1970; 1 to 2nd January, 1970; 35 to 5th February, 1970; 7000 to 3rd March, 1989 and so on. The only dimension is occupied by date and other information should be contained. Each number is drawn on number axis as a dot. The default color of a dot is black and it can be colored to represent temperature, i.e. gradient colors can be calculated to be mapped to different temperatures and all black dots are replaced by colorful dots. Obviously temperature trend is shown on number axis by dots which form a line of rainbow. The line implies the temperature trend by day.

Line chart for temperature visualization is common [40] [41] [42]. Temperature information can be drawn in rectangular coordinates. The x-axis represents date and y-axis temperature. This visualization approach uses poly line, which show temperature trend by day. It is a kind of 2D representation.

Another kind of 2D representation is to increase temperature trend information in one view. It is an approach combining the ways above. Temperatures are visualized by colorful dots and x-axis represents date in a year. Instead, y-axis represents year. And then matrix of colorful dots creates a network. A row of dots form a rainbow line to imply daily temperature trend while one from a column of dots refers to temperature trend by year.

Extended from the first 2D representation approach which uses line chart for temperature visualization, one of 3D representation is to add z-axis to represent year. Temperatures of a year are drawn in a flat line chart. All line charts are ordered by year in z-axis. Daily and annual temperature trend could be shown but annual temperature trend is not clear because screen is a flat and a point below another one can refer to a lower temperature or belong to a different year in a limited viewport.

Comparing each approach and considering our final goal, we prefer a 2D representation to show temperatures with their daily and annual trends. Al-

though temperature data break into two parts at 31st December and 1st January of next year, visualization through this approach will clearly show most information.

3.1.2 Graphical User Interface

The graphical user interface allows users to interact with a computer with images instead of text commands [43]. It supports an effective operation environment to control of a computer. Users perform actions through direct manipulation of graphical elements [44] and get feedbacks to make operational decisions.

To visualize temperatures in 2D representation, a view showing them is necessary, and also interactions. When facing a view of temperatures, users can explore data and find their interesting things. Then they can make some marks to highlight the things and read more detail information. As it turns out visual view, control and information panel should be realized. Visual view panel plays the most important role and data will be displayed on it. Control panel has some buttons to help users search for interesting points and make different marks on the points. Information panel collects detail information and presents it for users.

The layout for the three panels is simple (see Figure 3.1). Visual view panel should occupy most space for its significance of visualization. Control panel and information panel are arranged on the right side. Control panel is on top-right while information panel is on bottom-right. In addition, a menu bar getting all commands together will be on the top.

3.1.3 Pixel-based Block

At beginning of this chapter, we discussed some representation approaches and decided to use 2D representation. And y-axis represents year, meanwhile x-axis represents day in a year. Color coding is applied for visualizing temperature.

We define a block called day block. It is a rectangle containing many small rectangles which we call nodes (see Figure 3.2). A colored node represents a day in a year. Its color shows what temperature the day is. To add more information, season block and year block are defined also. There are four seasons of spring (1st March - 31st May), summer (1st June - 31st August), autumn (1st September - 30th November) and winter (1st December - the last day of February in next year). A season rectangle, which shows temperature of a season in average, is to combine nodes referring to days in same season. A season block has five segments as five nodes because the range of winter should be split into two periods as 1st January - the last day of February and 1st December - 31st December. And year block has only a node to represent temperature of a year in average.

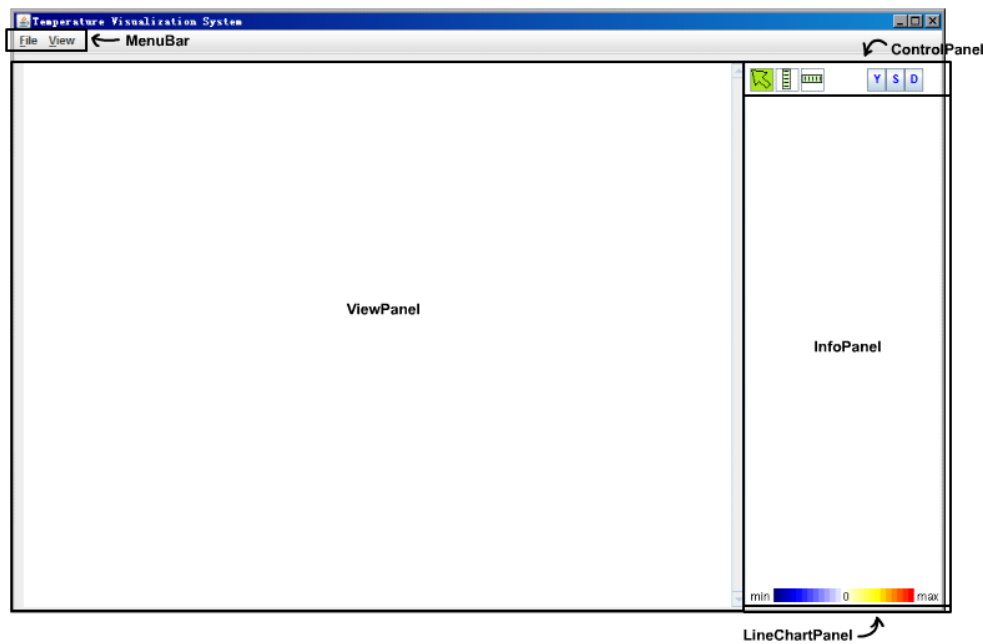


Figure 3.1: Graphical User Interface

Temperature should be mapped to color (see Figure 3.3). Bright colors make people feel warm and in contrary, dark colors make people feel cold. Then we define red represents temperature above zero while dark blue below zero. The temperature of zero is white.

Day, season and year block are all pixel-based block, since we map values into colored small rectangles as recognized pixel blocks. They are basic visual objects in our project.

3.1.4 Visual View in Limited Size

The visual view panel in limited size is a problem. It can hold pixel-based blocks but not too many. We find two solutions: **semantic zooming** and **scrolling**.

Semantic zooming is mentioned in section 1.3. Pixel-based blocks can be grouped by five years, ten years or more. New pixel-based blocks, which represent the average of temperature for a group of blocks, are generated. Pixel-based blocks representing a group can also be grouped and they form a hierarchy. Finally, the number of displayed blocks should be adapted to the volume ability of visual view panel. When user clicks on a group or several group, semantic zooming would be performed. Visual view should remove other visual objects and redraw the specified group in detail.

Scrolling is a classic interaction technique to enable user navigating a windowed view [44]. It is a special kind of panning. Panning is described in

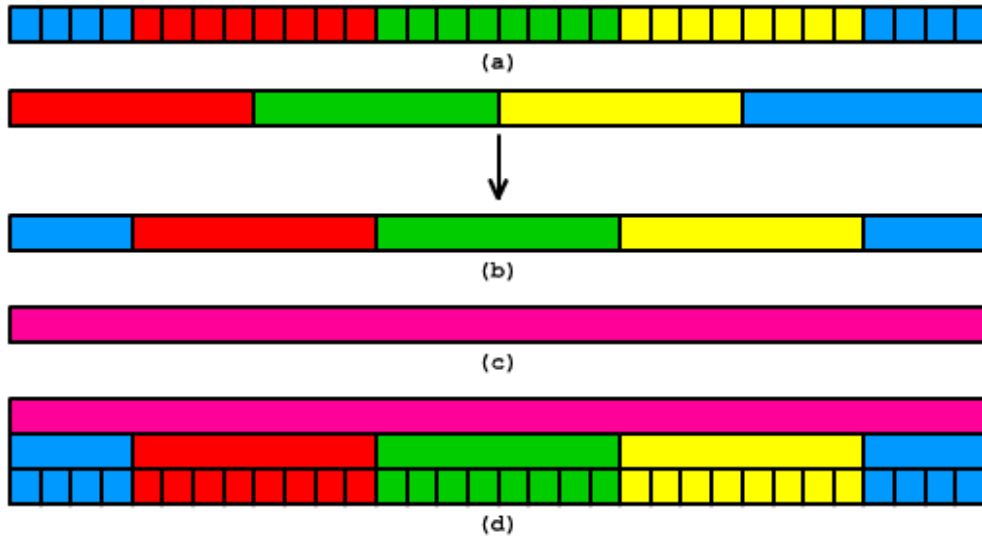


Figure 3.2: Kinds of blocks. (a) day block (b) season block (c) year block (d) combined block. The rectangle in (a) is divided into 366 segments because of leap year; there are 4 seasons but spring starts from March. Thus in (b) season block finally contains 5 segments. Both ends represent winter while spring, summer and autumn refer to the other parts in turn; a combined block is displayed in (d). It remarkably shows the hierarchy information about mean temperature.

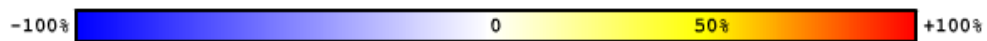


Figure 3.3: Color-coding for nodes. All value should be converted into percentage with positive or negative sign; a value of zero is mapped to white color; blue indicates value of negative number; yellow and red represents a positive value.

section 1.3. It provides a windowed view which can be moved in any direction. Scrolling merely supports moving a windowed view in vertical or horizontal way. When there are many visual objects to draw, a view in limited size cannot hold all of them. Therefore, a window is demanded to give the view a range to display part of objects. To control of scrolling, scroll bar is the right component (see Figure 3.4).

Both techniques can solve the problem that visual view is in limited size. The former one can make an overview about all temperature data in one view. The latter one can provide continuous trend information including every year. They match each other in strength. But scrolling is easy to realize. Considering time cost, scrolling is more priority than semantic zooming.

3.1.5 Selection

When interested in something on screen, user would like to mark it sometimes. Single selecting, a kind of selecting technique for interaction, is employed to achieve this aim. There define three modes (see Figure 3.5):

1. Picking one mode is a mode to enable user marking single node.
2. Picking column mode is for user to select a single column of nodes.
3. Picking row mode provides user with the ability to pick up all nodes which belong to a same year.

To make selected objects easier to locate, two ideas are discussed here. The first idea is to blur the others to highlight selected objects. For example, user selects a node. There will be a not completely transparent white mask to cover on the entire view except a hole for the selected node. Then the other nodes are blurred and the selected one stands out. The other idea is applying bounding box technique. After user selects objects, a black frame or frames would be drawn surrounding them (see Figure 3.6). We prefer the second. Blurring other objects will alter color of nodes a little because there is a not completely transparent white mask. It will affect reading more information from others after selecting something.

3.1.6 Zooming

A display has limited width with a specified resolution. We assume that user uses a screen of 1024 pixels by 768 pixels. A day block has about 350 nodes. Each of nodes can occupy only 2 pixels because it should be 1050 pixels greater than 1024 pixels if each node uses 3 pixels. Thus a node is very small. When intending to point at a day node, user will be confused whether the node is hit correctly or not.

We also find two solutions. Zooming is a fast answer. User can zoom in on a view to see nodes bigger and clearer. However, it will pay for the cost of a

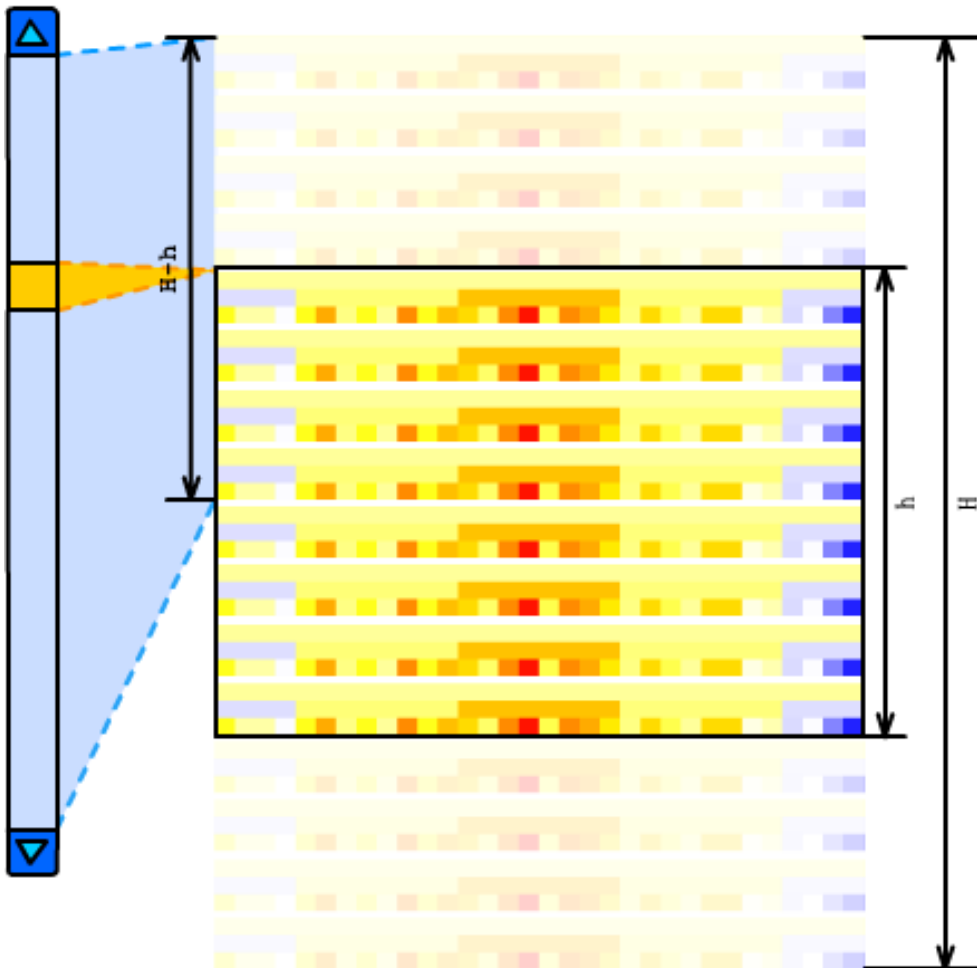


Figure 3.4: Scrolling with scroll bar. The height of entire view is H ; a windowed view's height is h ; the slot of scroll bar has the minimum value as the top of entire view; and it has the maximum value at H minus h to ensure the windowed view can display every combined block at least once, when traverse all values of the scroll bar; the slider block position represents the top of the windowed view.

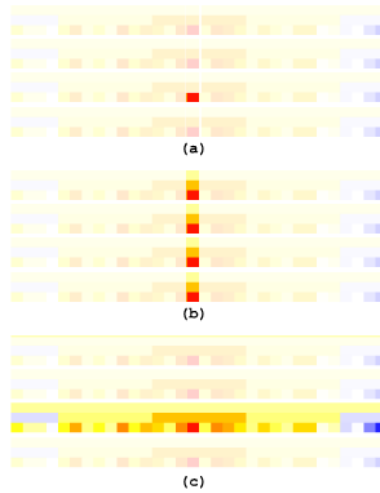


Figure 3.5: Three modes of single selection. Picking one mode is in (a); (b) is an example of picking column; (c) shows picking row mode; In addition, user can pick up a day, a season or a year node in picking one mode; also in picking column mode, a column of a day, a season or all year nodes can be selected.

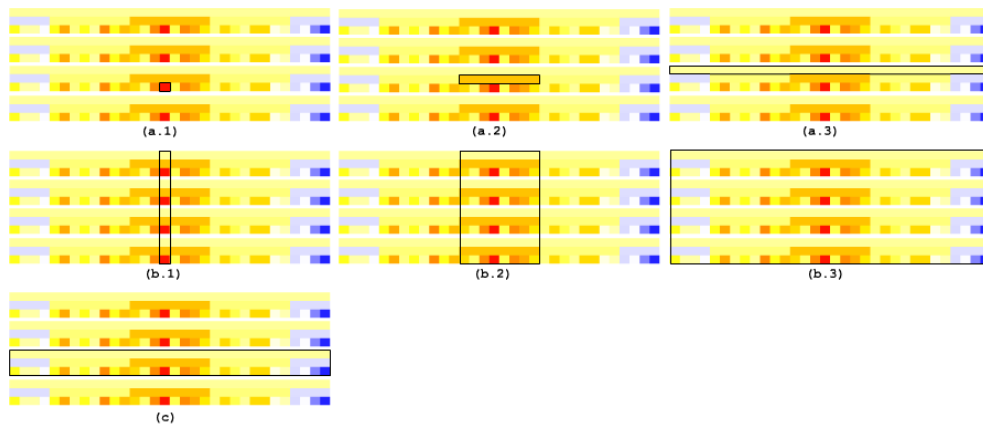


Figure 3.6: Bounding box for all conditions. (a.1) shows a bounding box for single day selected; (a.2) for single season; (a.3) for single year; (b.1) for a day column; (b.2) for a season column; (b.3) for year column; (c) for a row of a single year

little trend information lost when user zoom in. To keep trend information, the other solution is better. Fisheye view is a kind of technique in focus-plus-context visualization described in section 1.3. When mouse hovers on a day node, the node should be magnified. Nearby nodes will be magnified also while far nodes will be compressed in a same day block (see Figure 3.7). Therefore, a distorted view of fisheye could be useful for selecting. It pay for the less cost of trend information loss in vertical direction.

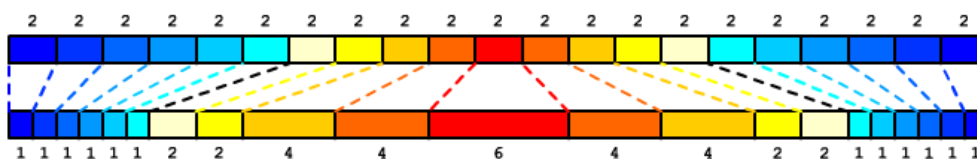


Figure 3.7: The fisheye view for day block within 21 elements. The number labeling above / below an element is how many pixel the related element occupies; the original view (top) contains 21 elements and each element is 2-pixel; the fisheye view (bottom) has 21 elements and each element is in different pixels.

3.1.7 Line Chart

Some people feel dizzy after watching on screen with objects in colors for a long time. Some people are not sensitive to colors, and some people also have difficulties in identifying two similar colors. They cannot get accurate temperature values from gradient colors immediately. Hence, chart is a good complement to visualize temperature.

A line chart could be used to enforce visualization of desired instances of data, such as visualize daily temperature of one year. Additionally, visual cues could be embedded to show intense details like the maximum and minimum temperature of the selected data.

A small line chart will be drawn on information panel, when user selects a column or a row of nodes (see Figure 3.8). It is in a limited space so that scrolling technique could be used to make reading all selected nodes possible. To get accurate value of temperature node, data tip, which shows timestamp and exact temperature value, will pop up if mouse hovers on a node point in the chart. In addition, light red / light blue bar implies the maximum / minimum temperature of the selected data. (see Figure 3.9)

3.1.8 Granularity Issues

We have defined three types of blocks. The three kinds of basic blocks, day block, season block and year block, can be displayed simultaneously. They get together to form a combined block. There are four kinds of combined block:

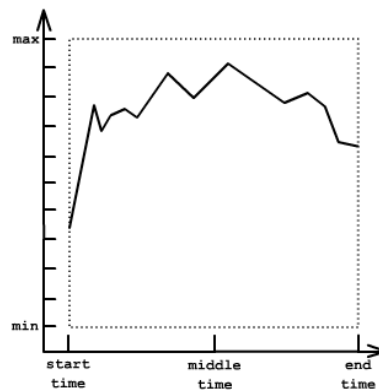


Figure 3.8: Line chart for this project. The chart has two parts. One is the main view to visualize temperature data; the other one is made of the x- and y-axis. There are 10 labels beside y-axis as calibrations and 3 labels below x-axis refer to time. The main view is limited to display part of picked data.

- full combined block of day, season and year block
- year-day combined block
- year-season block
- season-day block

However, combined blocks lose annual temperature trend, even though they contains more temperature information.

Changing layout is a good idea to keep annual temperature trend. The default layout uses one column of basic blocks. And it can be two columns of basic blocks. The left column includes basic blocks which are in same type and the other blocks are put into right column. At least one kind of basic block is not separated by the other blocks in different type.

Another way is applying filtering. Filtering is used to reduce complexity on clutter in a view, which means visual objects that a user is not interested in could be hidden. It is useful when a user want to look for and focus on something in specified range. For instance, user would like to know what the annual trend of a day is. Then he / she just needs to filter out season blocks and year blocks. Leaving only the day blocks will make it easier to understand the trend (see Figure 3.10).

The latter is selected. The reason behind it is that day node is really small and two columns of basic blocks will make day node smaller. Recognizing and picking a day become more difficult.

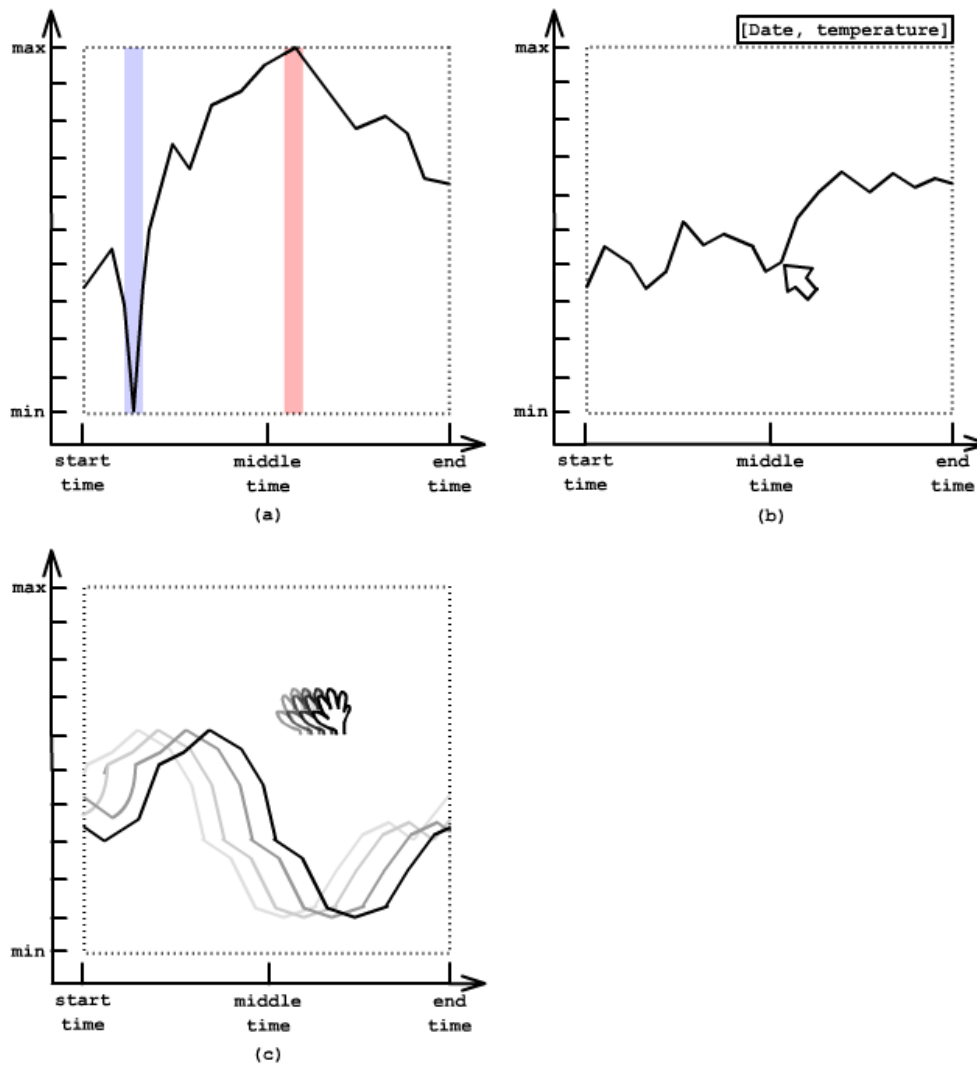


Figure 3.9: Interactions for the line chart. (a) shows the design of that light blue / light red strip highlights the data with minimum / maximum value; (b) figure out a solution to provide user with an interaction way to browse data; (c) displays an idea to view all data in a fixed window with dragging to scrolling technique.

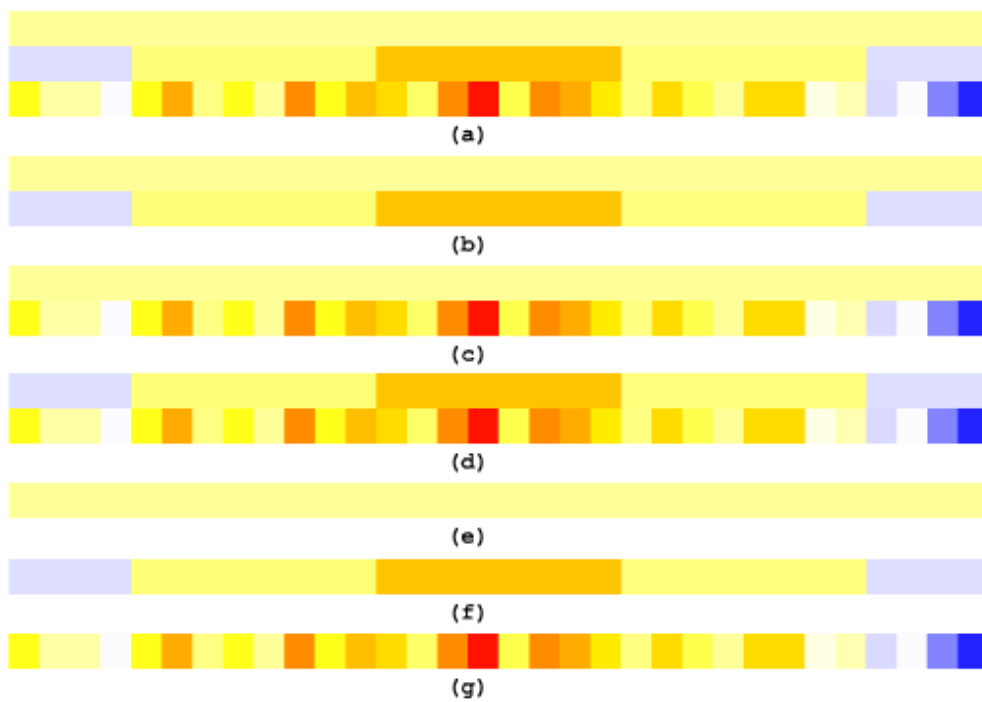


Figure 3.10: 4 kinds of combined blocks and 3 basic blocks. (a) full combined block with year-season-day block; (b) with year-season block; (c) year-day block; (d) season-day block; (e) year basic block; (f) season basic block; (g) day basic block.

3.1.9 Smooth Transition with Morphing

When the user filters out or in something, the program should redraw the entire view. A user may lose his attention to an interesting point after there is a sudden redrawing. To avoid this issue, we need to use smooth transition. Before completely redrawing, the program should create some frames and perform an animation to smooth the change (see Figure 3.11).

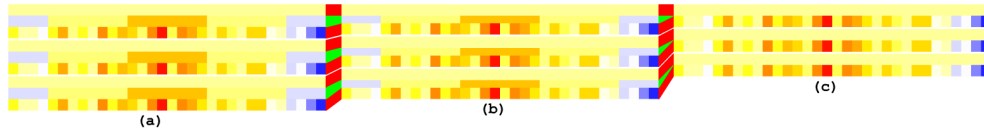


Figure 3.11: Three frames for morphing when combined block is changed. The combined block with year-season-day block in (a) is prepared for morphing; the year-day block in (c) is the final frame for morphing; the middle frame shows season block folded gradually in (b); the red-green areas between frames indicates season block folded and combined block position changed.

3.2 Implementation

In this section, a prototype will be developed.

3.2.1 Programming Language and Framework

Nowadays programming languages are various like C, C++, Java, C#, Python. And every year there are several languages born and some dies. The top 5 popular programming languages are C, Java, C++, Objective-C and C# [44]. C is father of the others. It is powerful and flexible for coding because people can use it to manipulate machine directly. However, different companies have different libraries to implement graphical drawing in C. Therefore program in C for visualization is not satisfactory in portability. C++ extends C and is an object-oriented language. But it is hard to study since it makes programming messy that a class can inherit from several classes and developers need to manage memory themselves. Objective-C is for Apple devices so that the portability of applications written in Objective-C is bad. Java and C# both are powerful. And they are also easy to study. Garbage collecting is a perfect technique so that people do not care about complex memory management [46]. C# is latest. However, it is developed by Microsoft and it is confined in Windows platform, although it can be port to Linux platform via Mono [47] or Wine [48]. It is new and the libraries from third parties for C# are not many.

Java is a popular and powerful programming language [49] [50]. Many people study it because of its portability and many libraries from third parties. There are numerous open source libraries supporting various tasks in Java. Thus we choose Java.

There are many nice libraries for visualization in Java. The library Processing [51] is powerful one. It defines a framework to visualize things but it is not stable. Sometimes application crashes due to it but the fault cannot be fixed. JFreeChart [52] is a library that makes chart implementation more convenient. But it slows down program's running speed. To escape from the limitations of third-party library, we decide to use none of them. In this project, Java Graphics 2D is only used to realize visualization.

Model-view-controller (MVC) [53] is the proper structure to match information visualization pipeline model. The pipeline model, which is described in Section 1.3, includes the elements of raw data, data transformation, data table, visual mapping, visual data, visual transformation, view. Models can refer to data states. They stores data in the same space but different attributes and values Controllers are the components to do data and visual transformation. They undertake the task to bring data from the old state to a new one. And views have regions to display specified data in different states. Therefore, MVC is suitable for the implementing of the pipeline model.

Adopting MVC architecture, we make a sketch of the prototype (see Fig-

ure 3.12, 3.13). Parser plays a role to fetch raw data and make data table which can be handled easily for computer. "Viewer", which does not refer to a person, is an instance of view to present visual objects. Image caches transform data table into graphical drawing and they are controllers. There are some components for interaction which are also controllers. After parser passes data table to controller to generate image caches, "viewer" fetches pictures in image caches to form a complete view. All segments are integrated together to build the final prototype.

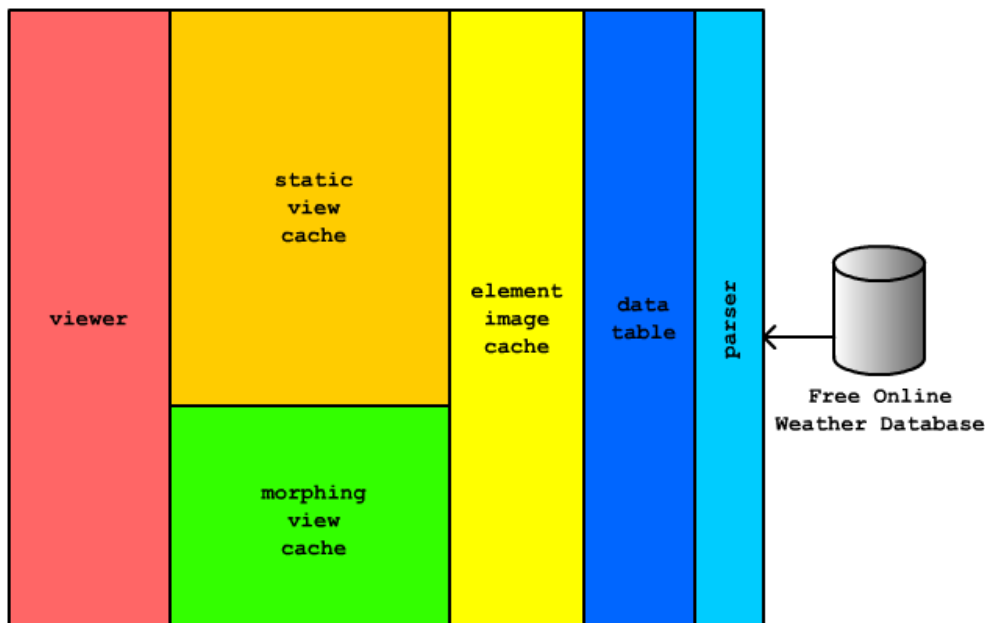


Figure 3.12: The architecture of the prototype in this project. The parts of parser and data table form model; controller is composed of 3 kinds of caches; there is a "viewer"; user downloads a dataset from free online weather database and launches the product of this project to load it; the parser converts pure text data into data objects which builds a data table; the cache controllers maintain buffered images of data; every piece of datum has an element image; interacting with user, controllers select what to draw and put visual data to viewer (there are 2 kinds of "what" to draw: static and morphing); "viewer" is a painter to present the visual data.

3.2.2 Data Parser

Our goal is to visualize temperature data. The mean temperature datasets from ECA&D are used in this project. There are four columns of SQUID, DATE, TG and Q_TG in each dataset table. SQUID is source identifier. It is reserved to use. DATE is the identifier of a record. It is a timestamp. TG is

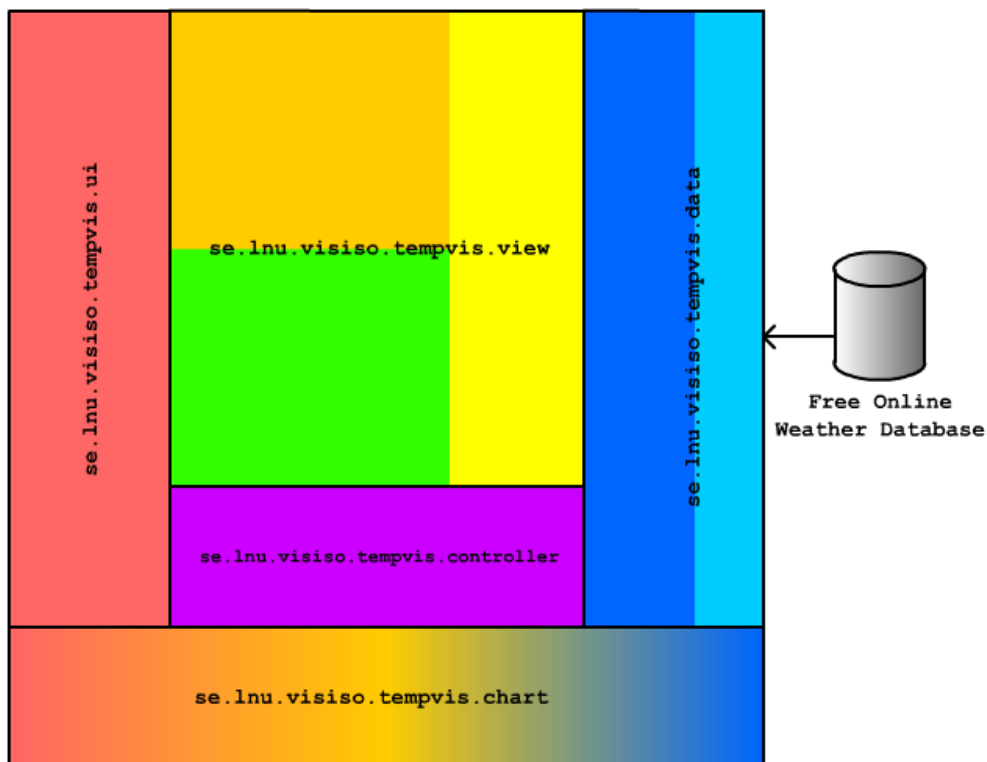


Figure 3.13: Package diagram of this project. Data package contains parser and data table; view package is made of buffered image caches; data selecting and color-coding is on the duty of controller package; all things presenting to user are from ui package; chart package is an individual package connected with data, controller and ui package.

mean temperature value in 0.1 degrees centigrade while its quality is stated by Q_TG. Moreover, the state of TG can be valid, suspected and missing. And DATE is split into three attributes of year, month and day to make them convenient to be processed for computer. In addition, when month is 0, 1, 2 or 3 and day is -1, the record refers to a season node. Month and day both are -1, which means the record refers to a year node.

Furthermore, temperature is a kind of weather information. If user would like to visualize other weather data, developer just needs to write a new parser to generate data table in specific format. Then the application can be extended to visualize all kinds of weather information.

Data parsing is the duty of data package. There are 8 classes to parse a text file into data table and do statistics to get a list of nodes with maximum temperature and minimum temperature. To handle recognition exceptions, city node has a list of "bad guys" (records without necessary information or with invalid one) to reinforce application error tolerance.

3.2.3 Color Coding for Temperature

Color is a very important attribute. There are two kinds of color, cold color and warm color. And the cold one is just for low temperature while the warm one is for high. Color gradient will be defined to code a range of temperature values. RGB is a famous color model [54]. Every color is calculated by a linear relation within red, green and blue as

$$value(color) = R \cdot value(red) + G \cdot value(green) + B \cdot value(blue)$$

The coefficients of R, G and B are mapped into (0, 1) in some mathematical way. The vector (R, G, B) can be treated as a point in space. Then all colors are confined in a cube space [30]. And now it is easy to create a gradient color table. Assume P(R, G, B, c) refers to a color (R, G, B); c is constant to specified a value precisely to map to the color. Two points $P_1(R_1, G_1, B_1, c_1)$ and $P_2(R_2, G_2, B_2, c_2)$ builds a line L. Here comes a value c_0 between c_1 and c_2 . What color is it mapped to?

First, L should be presented via a parameter t

$$L : \begin{cases} R = (R_2 - R_1)t + R_1 \\ G = (G_2 - G_1)t + G_1 \\ B = (B_2 - B_1)t + B_1 \\ c = (c_2 - c_1)t + c_1 \end{cases}$$

Second, c_0 is mapped to a color (R_0, G_0, B_0) . Thus (R_0, G_0, B_0, c_0) is on the line L. Then,

$$c_0 = (c_2 - c_1)t + c_1$$

Therefore,

$$t = \frac{c_0 - c_1}{c_2 - c_1}$$

And the color mapped is

$$\left(\frac{(R_2 - R_1)c_0 - R_2c_1 + R_1c_1}{c_2 - c_1}, \frac{(G_2 - G_1)c_0 - G_1c_1 + G_1c_1}{c_2 - c_1}, \frac{(B_2 - B_1)c_0 - G_1c_1 + G_1c_1}{c_2 - c_1} \right)$$

Third, pick more than two points to make a polyline to get a completed gradient color table. A program just searches for two adjacent points to let a specified value in range. And use the result above to get color of the value.

There 3 classes of SpacePoint, SpaceLine and ColorGradient in controller package undertake the work to calculate color gradient. SpacePoint is to hold a vector (R, G, B, c); SpaceLine works out line equation between two points; ColorGradient contains a list of points as a polyline and implements color mapping algorithm with the formula above (see Figure 3.14).

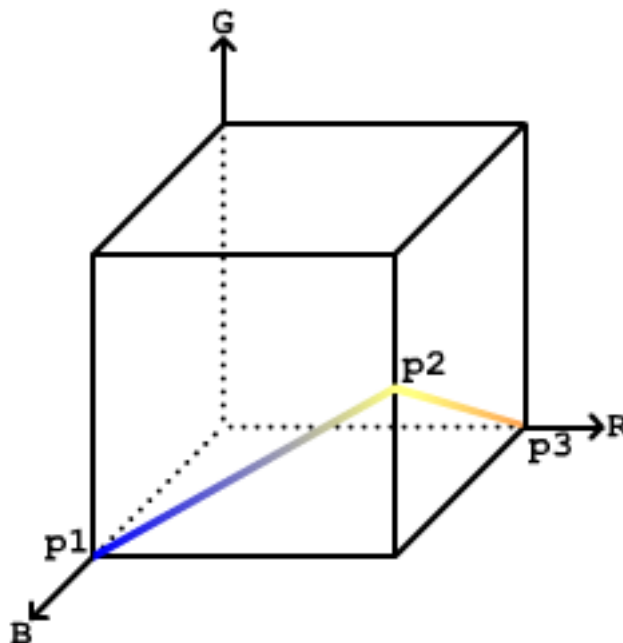


Figure 3.14: A polyline for gradient color table. It's an example to use a polyline to define a gradient color table. There are 3 points p1(0.00, 0.00, 1.00), p2(1.00, 0.50, 1.00) and p3(1.00, 0.00, 0.00). And 3 static values are c1 = -2.00, c2 = 1.00, c3 = 3.00; p1 and p2 make a sub gradient color table meanwhile p2 and p3 make another one. If there is a value of -1.00, it is between c1 and c2. Its color referring to p1 and p2 is (0.33, 0.17, 1.00), a kind of blue. If a value of 2.00, its color should be (1.00, 0.25, 0.50), a kind of red.

3.2.4 Buffered Image Cache

Buffer is significant. It is a container to hold the most frequently used objects in a large scale, especially when it will take much time to recalculate them. In another word, buffer saves middle results of reckoning low level data to accelerate more important other calculating. Thanking for the enhanced memory today, people can use buffer technique freely and not need to think much about how to manage memory efficiently.

A dataset is made of about ten thousand records. It is huge. And the data are static. Every year has a year node, 4 season nodes and 366 day nodes of totally 371 nodes. Assuming a view can display data of 20 years, there will be 7420 nodes to draw. It could work to draw them directly but will be stuck if user wants drawing as if it is in real time. In another word, it is very slow for a direct drawing. However, buffer is helpful to draw faster. 371 nodes in a combined block are drawn into an image stored in a buffer. Then a view shows 20 years and the program just reads the buffer and draws 20 buffered images. It is much quicker than to draw 7420 nodes. Therefore in a best way, buffer technique, i.e. cache, is utilized. In the view package, YearImage is a class to draw a full combined block in a cache as designed in section 2.2. Considering filtering configurations, ViewImage picks up buffered images in the YearImage cache and selects right blocks of year block, season block or day block to draw in a new buffered image.

In parallel, AnimationElementImage is based on YearImage and builds a buffered image of morphing view with AnimationImage and AnimateThread. They will be discussed in section 3.5.

In addition, YearLogicFisheye is a class to hold the buffered image of logic fisheye view. It gets 21 adjacent nodes and draws them with distortion as designed in section 2.5.

3.2.5 Rectangle Manager

ViewPanel is a class in UI package and ViewScroll in controller package. After loading a temperature dataset, the program will draw combined blocks on ViewPanel. At first, every combined block should be registered with a rectangle in a ViewScroll instance. The rectangle refers to the position of a combined block in a global view. Because of the large size of the global view, a window in ViweScroll could select part of them to present to user. The window is a rectangle, which picks the registered rectangles. Each picked rectangle is contained in or crossing with the window rectangle. Then ViewPanel fetches what to show from ViewScroll's window and draw combined blocks whose mapped rectangles are picked.

PickOne is significant class to support single selecting and fisheye view. When mouse move event is fired in ViewPanel, a number pair (X, Y) is pushed to PickOne. PickOne will search for a rectangle containing this point in all

registered ones of ViewPanel. Then PickOne grasps which node in the found rectangle pointed at. The node is a year node, season node or day node. Now the single selection is realized. It is also on the duty of PickOne to provide a rectangle for the selected node. Through that node, ViewPanel draws a bounding box to highlight it, a column which it belongs to or a row for its year.

Fisheye view is based on the result of PickOne's selected node. YearLogicFisheye receives a selected node from PickOne and does nothing if the node is not a day node. And in the same year, 10 day nodes forward the node, 10 ones backward it and it will be collected. YearLogicFisheye prepares a buffered image to store pixels showing the 21 rectangles with fisheye distortion. Finally, the prepared image is revealed to cover the original 21 rectangles.

Filtering works in ViewPanel. Once user sends a command to change filtering configuration, ViewPanel records the top block in the window. And then it tells ViewScroll to change registered rectangle. The last thing is to set the window's top align to the recorded rectangle.

3.2.6 Animation for Smooth Transition

Thread should be used to finish animation, which runs apparently in parallel with each other threads. Animation has two parts working together. One part is preparing an image frame. The other part is outputting the frame. AnimateImage prepares frame to ViewPanel to present. And the prepared frame consists of AnimationElementImage. It should be figured out in real time. AnimateThread takes the responsibility to finish a frame.

AnimateThread tells AnimationImage which frame to draw. AnimationImage commands ViewScroll to alter all registered rectangles in height and position gradually. Meanwhile, AnimationElementImage is responsible to draw a block under changing. The procedure to act animation is simple. AnimateThread as a timer thread receives frame number to draw the specified frame into buffer. AnimateImage fetches a frame from the buffer.

4 Discussion and Conclusion

The implementation yields a prototype application which visualizes temperature for one specified place (see Figure 4.1). A row refers to temperatures in a year and different years' are displayed in column. Thus it shows two kinds of trends, daily trend and annual trend, with gradient colors in matrix. And it uses 2D representation where daily temperatures in a year are in x-axis and history temperatures of different years are in y-axis. Color is mapped to temperature. It contains three cardinal pieces of information, daily trend, annual trend and temperature. They are defined as t-t-T (time-time-temperature) information. This visualization is named as matrix-based visualization.

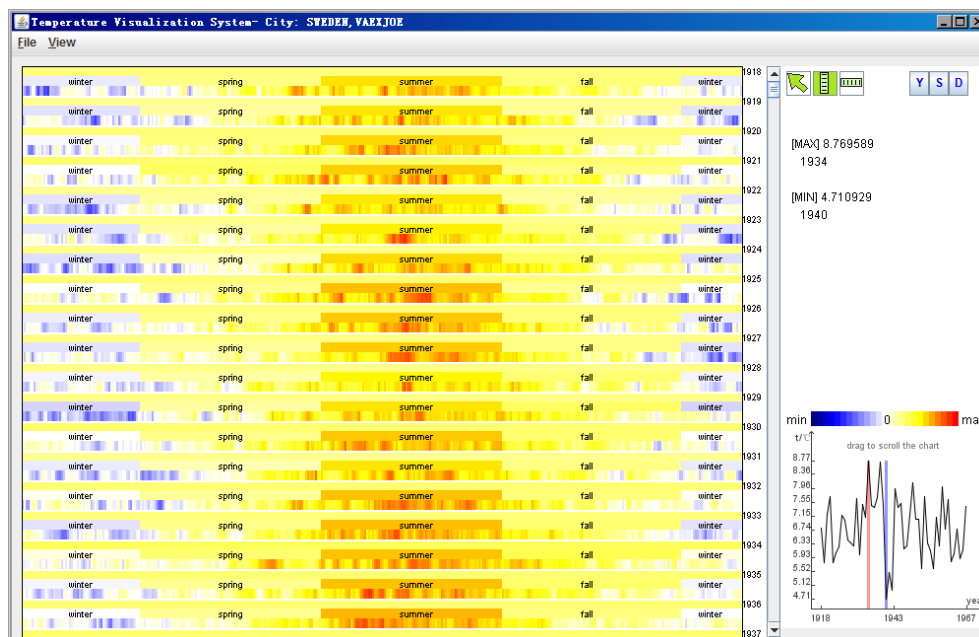


Figure 4.1: Final result of the prototype for temperature visualization. The dataset is for Vaxjo, Sweden. The selecting mode is for a column and all year nodes are picked while they are plotted in the line chart. InfoPanel shows the year with max and min temperature.

4.1 Discussion

Claude Shannon [55] established information theory in 1948. He defined a concept called information entropy to describe bandwidth of binary information. As is known to almost all, it is a truth that the sun rises from the east. It, as a message, gives people little information and thus the entropy of that truth is nearly zero. The information contains high entropy where people think it is impossible but maybe true. Visualization is any technique for

making images, diagrams, or animations to provide a message for human's eyes.

Therefore, we recognize that the essence of information visualization is to highlight things in high entropy and avoid information containing low entropy in a graphical view. There should be specified standard one to describe entropy is high or low. And it is applied in the comparison between temperature visualizations to show which one is better.

We built a prototype to visualize temperature in 2D. Our matrix-based visualization offers t-t-T information for one place. And we described several visualizations in 2D representation in the beginning of Chapter 2. Line chart visualization can provide t-T (time in x-axis, temperature in y-axis) information for one place while map-based visualization can provide position-T (position as a point in x-y coordinates, temperature in color) information. In addition, toward trend, which means temperature of a place has an influence on temperatures in nearby places, can be shown in map-based visualization.

Our goal is to develop a prototype to help people understand temperature trends for one specified place. Line chart visualization can show daily temperature trend. It is a common and good way to visualize temperature. In map-based visualization, temperatures in different places are displayed and toward trend can be understood for places not a specified place. It is not a suitable way to visualize temperature for one place.

And matrix-based visualization shows not only daily temperature trend but also annual trend conditionally (see Figure 4.2). When two or more kinds of nodes, year node, season node and day node, are displayed, there will be no annual trend. The reason behind it is that the trend between nodes in the same type is broken by other kinds of nodes. Annual trend appears only if the view contains one kind of node. Therefore, two or more kinds of nodes displayed make matrix-based visualization regress to line chart visualization, which shows daily temperature trend. But it is better than line chart visualization because regressing matrix-based visualization provides more temperature data in hierarchy. Only one kind of node displayed shows less information about temperature values. However, annual trend can be understood to make up for the loss. Thus it is also better than line chart visualization which merely shows daily trend.

In summary, matrix-based visualization shows more information about temperature trend or temperature values. It is an improvement of temperature visualization compared to axis-based visualization.

4.2 Conclusion

A prototype is invented with matrix-based visualization to make temperature trend stand out. It can show daily temperature trends and conditionally annual trends.

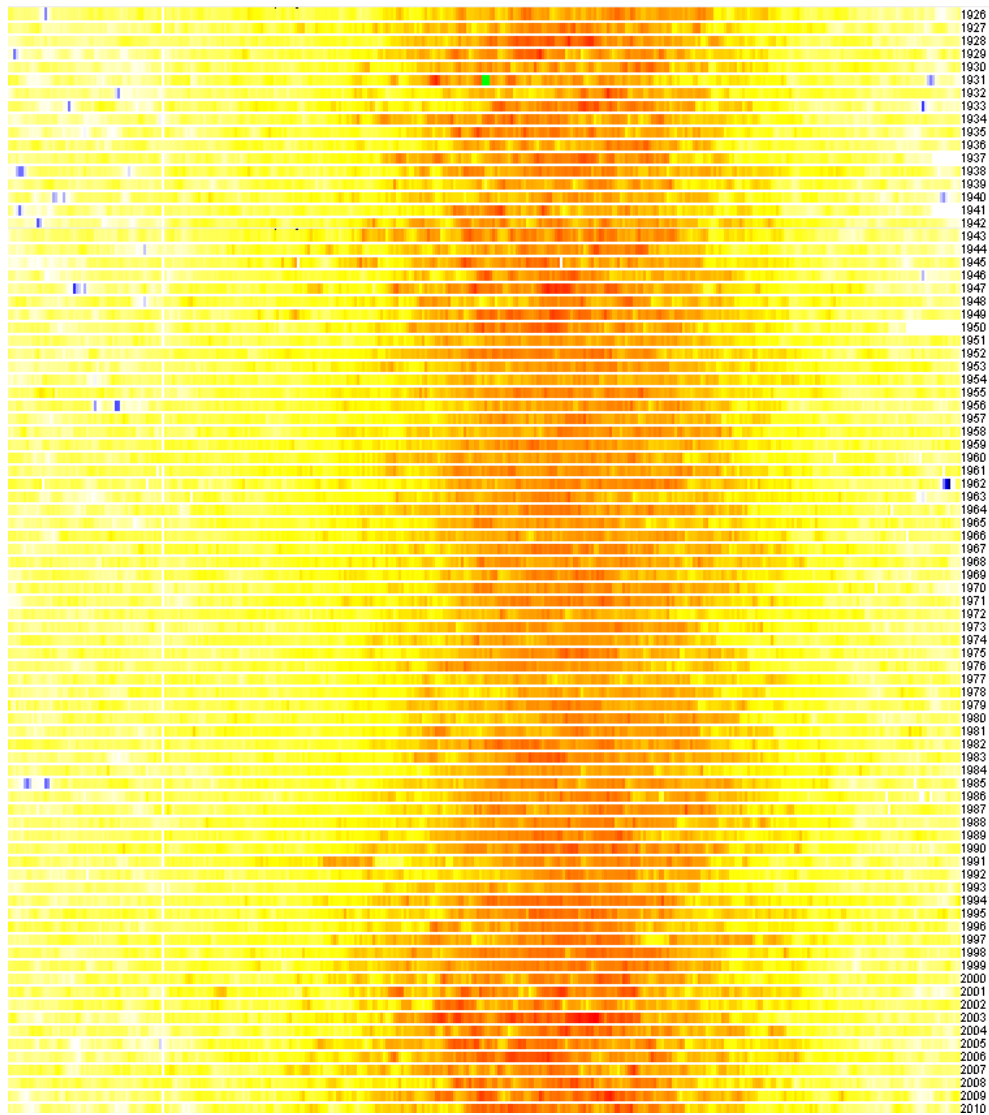


Figure 4.2: Matrix-based approach for temperature visualization. The dataset is of Barcelona, Spain; in horizontal, user can know annual trend of temperature while entire trend is shown in vertical; in general the daily mean temperature values of Barcelona are almost above zero and hottest days are apparently in summer; in vertical, Barcelona's climate becomes warmer in 50 years but does a cycling in 100 years.

In conceptual design, we used matrix to hold temperature. A cell for a piece of temperature data in matrix is mapped to color. In addition, line chart will be drawn to filter out annual trend information to help people focus on temperature values. And some interactions are used: scrolling is used to show temperature data in a view with limited size; filter makes it easier to search for interesting points easier; sudden redrawing entire view is avoided via using morphing techniques.

Following the design, we implemented the application carefully and made it convenient to be extended. The task to build the target prototype is fulfilled. It is portable and stable.

4.3 Future work

Although the prototype we build works well on temperature visualization, there are many things to improve. Before discussing future work, we talk about the drawbacks of our application.

1. Single selecting is limited. User cannot analyze specified range of daily temperatures. For example, the maximum temperature between 1st May and 31st May in 1990s cannot be read directly. And temperature comparison between columns or rows is not available, such as comparing the differences between temperatures in 2000 and 2010.
2. Missing data breaks continuous trend. Missing data are mapped to green. User can find missing data in a view immediately. However, green missing data affect user's understanding of temperature trend. When users go through a row or column and meet a missing node of temperature, they would split the row or column into two parts by the node. The node will become a breakpoint of trend.
3. Daily trend between years is interrupted. 31st December and 1st January of next year are separated in two day blocks. When users would like to grasp the temperature trend from 1st July to 30th June of next year, they will treat the range as two segments, where one is from 1st July to 31st December and the other is from 1st January to 30th June.

Multiple selecting is a feature to pick some elements. It will be used to select temperature nodes in specified range. In visualization view, a selecting rectangle is painted by mouse drag and nodes in the rectangle are selected. Then application can use them to do statistical calculating or some special comparing, even do clustering when selected temperature data are huge.

Flexible hierarchy is a good idea to grasp more information for data in range. Temperature nodes will be grouped not only by season and year but also by month, 10 days and others. And the group setting is under user's control. The application will be more customized.

To avoid green nodes of missing data, the prototype will be more intelligent so that it can predict missing temperatures. It collects temperatures adjacent to missing one, gives the missing one a value adapt to daily and annual trend in its row and column and draws the predicted one in not completely transparent color instead of the missing one in green.

Daily nodes will be continuous in date to take away break between two years. Spiral coordinate is helpful. It can inherit all functions from matrix-based visualization and make nodes continuous in date. It uses polar coordinate to locate every node and latest temperatures are shown in outside and bigger nodes. It will be a wonderful visualization with spiral coordinate.

Moreover, plug-in technique can be used to open a door for developers. We can provide a set of application programming interfaces to attract more people make contributions to temperature visualization based on our work. And this prototype can be easily extended to visualize kinds of data such as atmosphere pressure, humidity, wind condition, rainfall and so forth.

References

- [1] Lars Heide. Punched-Card Systems and the Early Information Explosion, 1880-1945 (Studies in Industry and Society). The Johns Hopkins University Press, 2009
- [2] Jack Fuller. What Is Happening to News: The Information Explosion and the Crisis in Journalism. The University of Chicago Press, 2010
- [3] Latanya Sweeney. Information Explosion, last accessed: 2012-03-18. <http://dataprivacylab.org/dataprivacy/projects/explosion/explosion2.pdf>
- [4] Information Graphics, last accessed: 2012-05-21. http://en.wikipedia.org/wiki/Information_graphics
- [5] Patrick Casey, Josh Miller. The World Reduced to Infographics: From Hollywood's Life Lessons and Doomed Cities of the U.S. to Sociopathic Cats and What Your Drink Order Says About You. Ulysses Press, 2011
- [6] Dona M. Wong. The Wall Street Journal Guide to Information Graphics: The Dos and Don'ts of Presenting Data, Facts, and Figures. W. W. Norton & Company, 2010
- [7] Edward R. Tufte. Envisioning Information. Graphics Press, 1990
- [8] Noah Iliinsky and Julie Steele. Designing Data Visualizations. O'Reilly Press, 2011
- [9] Riccardo Mazza. Introduction to Information Visualization. Springer Press, 2009
- [10] S. Card, J. Mackinlay and B. Shneiderman. Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann Press, 1999
- [11] Ilir Jusufi. Towards the Visualization of Multivariate Biochemical Networks. Linnaeus University Press, 2012.
- [12] Stuart K. Card, Jock D. Mackinlay, Ben Shneiderman. Readings in Information Visualization: Using Vision to Think. Academic Press, 1999
- [13] Lloyd A. Treinish. Creating Effective Visualizations for Operational Weather Forecasting, Proceeding of the 15 th IIPS Conference, 1999. Last accessed: 2012-05-21. <http://researchweb.watson.ibm.com/weather/AMS99/paper/IIPS16.16.pdf>

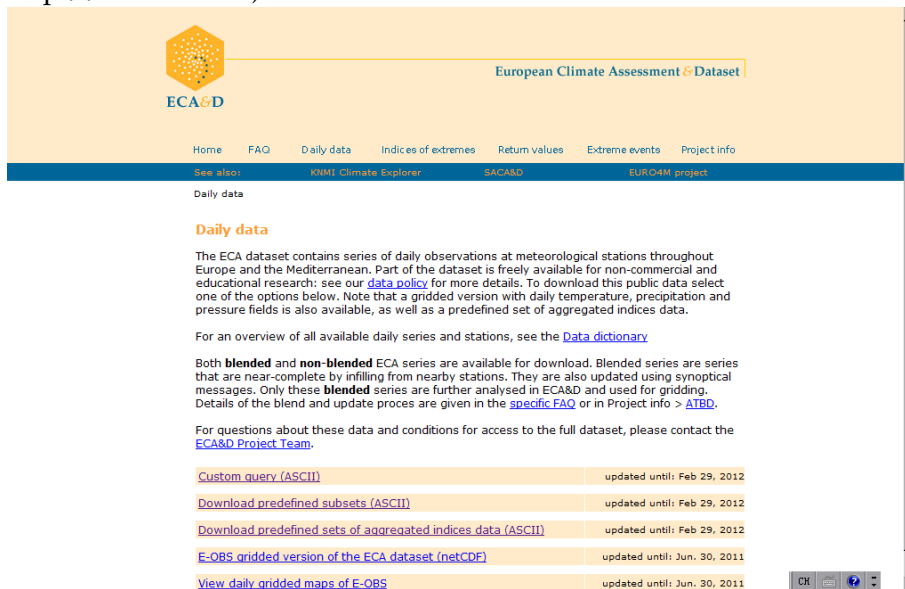
- [14] Andreas Kerren, Ilir Jusufi. 3D Kiviat Diagrams for the Interactive Analysis of Software Metric Trends, Proceedings of the 5th ACM Symposium on Software Visualization (SoftVis '10), ACM Press, 2010, p. 203-204
- [15] Julie Steele, Noah Iliinsky. Beautiful Visualization: Looking at Data through the Eyes of Experts (Theory in Practice). O'Reilly Press, 2010.
- [16] European Climate Assessment and Dataset, last accessed: 2012-05-31. <http://eca.knmi.nl/>
- [17] Thomas J. Bergin, Richard G. Gibson. History of Programming Languages, Volume 2. Addison-Wesley Press, 1996.
- [18] WEATHER BRACELET - 3D PRINTED DATA-JEWELRY, last accessed: 2012-03-19. <http://teemingvoid.blogspot.se/2009/10/weather-bracelet-3d-printed-data.html>
- [19] Flickr Flow, last accessed: 2012-03-19. <http://hint.fm/projects/flickr/>
- [20] Weather Tower, last accessed: 2012-03-19. <http://datavisualization.ch/showcases/weather-tower/>
- [21] Ed H. Chi. A Framework for Information Visualization Spreadsheets. University of Minnesota Press, 1999
- [22] Stuart K. Card, Jock Mackinlay. The structure of the information visualization design space, In Proceedings of the Symposium on Information Visualization '97, p. 92-99. IEEE Press, 1997
- [23] Chernoff face, last accessed: 2012-05-22. http://en.wikipedia.org/wiki/Chernoff_face
- [24] Chart, last accessed: 2012-03-22. <http://en.wikipedia.org/wiki/Chart>
- [25] The History of the Cluster Heat Map, last accessed: 2012-03-22. <http://www.cs.uic.edu/~wilkinson/Publications/heatmap.pdf>
- [26] Jenifer Tidwell. Designing Interfaces: Patterns for Effective Interaction Design. O'Reilly Press, 2005. Last accessed: 2012-05-22. http://designinginterfaces.com/firstedition/index.php?page=Overview_Plus_Detail
- [27] Y. K. Leung, M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques, ACM Transactions on Computer-Human Interaction (TOCHI), 1(2), ACM Press, 1994, p. 126-160
- [28] Manojit Sarkar, Marc H. Brown. Graphical fisheye views, Communications of the ACM, 37(12), ACM Press, 1994, p. 73-83

- [29] Ji Soo Yi, Youn ah Kang, John Stasko, Julie Jacko. Toward a Deeper Understanding of the Role of Interaction in Information Visualization, *IEEE Transactions on Visualization and Computer Graphics*, 13(6), IEEE Press, 2007, p. 1224-1231
- [30] Patterns: Filter, last accessed: 2012-05-22. <http://www.infovis-wiki.net/index.php?title=Patterns:Filter>
- [31] Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report. Stanford InfoLab, 1999.
- [32] Semantic Zoom, last accessed: 2012-05-22. http://www.infovis-wiki.net/index.php?title=Semantic_Zoom
- [33] Google Maps, last accessed: 2012-05-22. <http://maps.google.com/>
- [34] Patterns: Smooth Transitions, last accessed: 2012-05-22. http://www.infovis-wiki.net/index.php?title=Patterns:Smooth_Transitions
- [35] Morphing, last accessed: 2012-05-22. <http://en.wikipedia.org/wiki/Morphing>
- [36] Rendering (computer graphics), last accessed: 2012-05-22. [http://en.wikipedia.org/wiki/Rendering_\(computer_graphics\)](http://en.wikipedia.org/wiki/Rendering_(computer_graphics))
- [37] Wind Map, last accessed: 2012-05-23. http://hint.fm/wind/?utm_source=Sailthru&utm_medium=email&utm_term=Very%20Short%20List%20-%20Daily&utm_campaign=VSL%204%2F02
- [38] US Wind Patterns, last accessed: 2012-05-23. <http://www.senchalabs.org/philogl/PhiloGL/examples/winds/>
- [39] Sociale media weersentiment vs. KNMI weerdata, last accessed: 2012-05-23. <http://weatherchart.cleverfranke.com/Weathchart-CF.pdf>
- [40] Weather Spark - Beautiful Weather Graphs and Maps, last accessed: 2012-04-19. <http://weatherspark.com/>
- [41] Sample Temperature (°C) over Time (date) Line Graph, last accessed: 2012-05-23. <http://www.ciese.org/curriculum/weatherproj2/en/popup/graph2.shtml>
- [42] Temperature record of the past 1000 years, last accessed: 2012-05-23. http://en.wikipedia.org/wiki/Temperature_record_of_the_past_1000_years

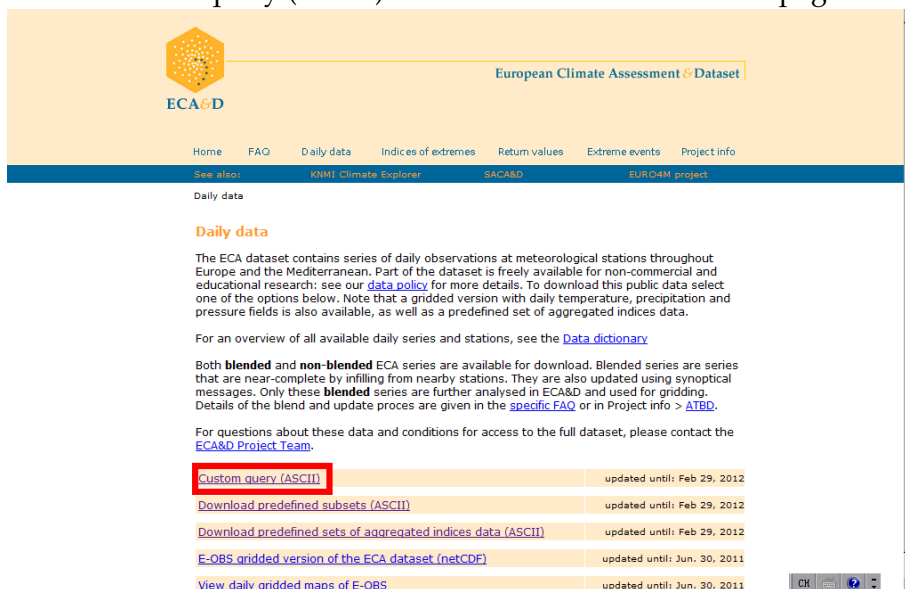
- [43] Graphical user interface, last accessed: 2012-05-23.
<http://en.wikipedia.org/wiki/GUI>
- [44] Definition of window manager, last accessed: 2012-05-24.
http://www.pcmag.com/encyclopedia_term/0,2542,t=window+manager&i=54598,00.asp
- [45] TIOBE Programming Community Index for May 2012, last accessed: 2012-06-01.
<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>
- [46] Garbage collection, last accessed: 2012-06-01.
[http://en.wikipedia.org/wiki/Garbage_collector_\(computing\)](http://en.wikipedia.org/wiki/Garbage_collector_(computing))
- [47] Mono official website, last accessed: 2012-06-01. <http://www.monoproject.com/>
- [48] WineHQ official website, last accessed: 2012-06-01.
<http://www.winehq.org/>
- [49] Cay S. Horstmann, Gary Cornell. Core Java™, Volume I - Fundamentals, 8th edition. Prentice Hall Press, 2007
- [50] James Gosling, Bill Joy, Guy Steele, Gilad Bracha. Java™ Language Specification, 3rd edition. Addison-Wesley Press, 2005.
- [51] Processing library, last accessed: 2012-06-01. <http://processing.org/>
- [52] JFreeChart library, last accessed: 2012-06-01.
<http://www.jfree.org/jfreechart/>
- [53] Elisabeth Freeman, Eric Freeman, Bert Bates, Kathy Sierra and Elisabeth Robson. Head First Design Patterns. O'Reilly Press, 2004
- [54] RGB color model, last accessed: 2012-04-04.
<http://en.wikipedia.org/wiki/RGB>
- [55] Claude E. Shannon. A Mathematical Theory of Communication. Bell System Technical Journal 27 (3), 1948, p. 379-423.

A How to download the dataset files

1. Open the website of European Climate Assessment & Dataset (ECA&D, <http://eca.knmi.nl>).



2. Click "Custom query (ASCII)" to enter the dataset selection page.



3. The options all depends on you except that the "Mean temperature" item should be chosen for the option Element. After all done, there is a "Next" button and click it to go to the download page.

European Climate Assessment & Dataset

Home FAQ Daily data Indices of extremes Return values Extreme events Project info

See also: [KIM1 Climate Explorer](#) [SAC&D](#) [EURO4M project](#)

[Daily data](#) > Custom query in ASCII

Custom query in ASCII

Select *country*, *location* and *element* to specify your query. Before that, choose whether you want your series to be **non-blended** or **blended**. Additional selection criteria are optional.

Your selection now yields less or equal than 500.000 observations. Proceed with the **Next** button.

Reset all Next

Type of series

Country
1 countries selected

Location
1 locations selected

Element
1 elements selected

Additional selection criteria

4. Click "Download" button.

European Climate Assessment & Dataset

Home FAQ Daily data Indices of extremes Return values Extreme events Project info

See also: [KIM1 Climate Explorer](#) [SAC&D](#) [EURO4M project](#)

[Daily data](#) > [Custom query in ASCII](#) > Summary of selection

Summary of selection

This page summarizes your query from the ECA dataset. Click the button to download the data. *More details* gives access to details about the series in your selection.

The exact source of each observation in the **blended** series can be traced back from the first figure of the source ID (SOUID). A source ID starting with 9 indicates synoptical data, whereas 1 indicates participant data.

No changes have been made to the source data from the participants. Only quality codes have been added. More details on the source data are available upon request from [ECA&D Project Team](#).

Country	SWEDEN
Station	Vaexjoe
Element	Mean temperature
Period	All available years
Blending	no

Estimated filesize: 1 Mb

Download

More details about the series in your selection


5. There will be a pop-up window to confirm to download the dataset file. If your browser does not allow pop-up window, please give it a chance to show.

Download

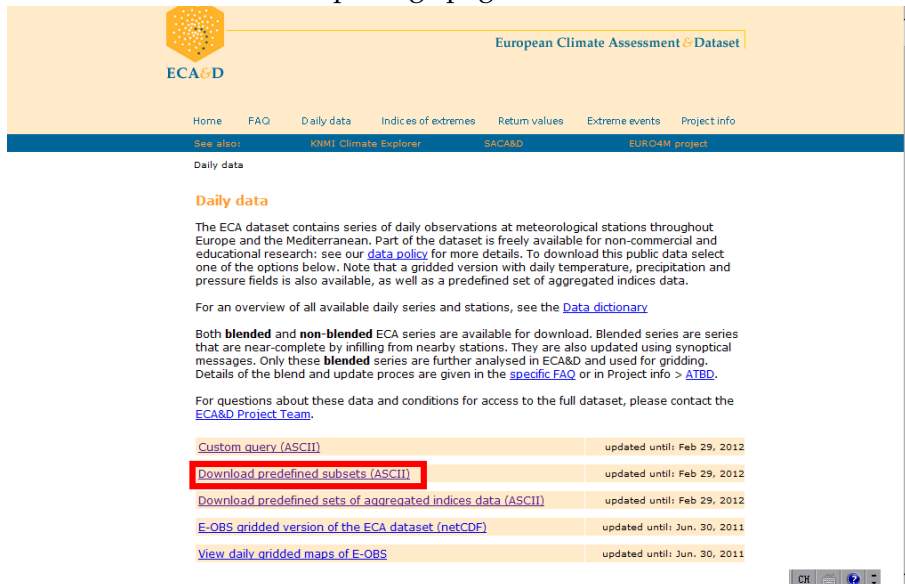
Your data file is now being created. This may take a minute; please wait until it is ready. When ready, a link for downloading your file will appear. **Do NOT close this window while your file is being created!**

[1 of 1] Processing element **Mean temperature**
(Vaexjoe) (Vaexjoe)

... Creating sources.txt ...
... Creating elements.txt ...
... Creating metadata.txt ...
... Creating ZIP-file ...

 [Download your file \(291Kb\)](#) [Close this window](#)

6. Extra: If you feel that it is boring for the procedure above, there is another to download dataset files directly. Come back to the home page of ECA&D. And choice "Download predefined subsets (ASCII)". Then it will forward to dataset package page.



The screenshot shows the ECA&D website interface. At the top, there is a navigation bar with the ECA&D logo and the text "European Climate Assessment & Dataset". Below this, there are several menu items: Home, FAQ, Daily data, Indices of extremes, Return values, Extreme events, and Project info. A secondary navigation bar includes "See also:" followed by links to "KNMI Climate Explorer", "SACA&D", and "EURO4M project".

The main content area is titled "Daily data". It contains a section for "Daily data" with a brief description of the dataset and a link to the "Data dictionary". Below this, there is a table of download options:

Custom query (ASCII)	updated until: Feb 29, 2012
Download predefined subsets (ASCII)	updated until: Feb 29, 2012
Download predefined sets of aggregated indices data (ASCII)	updated until: Feb 29, 2012
E-OBS gridded version of the ECA dataset (netCDF)	updated until: Jun. 30, 2011
View daily gridded maps of E-OBS	updated until: Jun. 30, 2011

7. Download the mean temperature datasets. The two both works.

Blended ECA dataset		
All elements (837Mb)	Sources	Stations
Daily maximum temperature TX (119Mb)	Sources	Stations
Daily minimum temperature TN (121Mb)	Sources	Stations
Daily mean temperature TG (107Mb)	Sources	Stations
Daily precipitation amount RR (215Mb)	Sources	Stations
Daily mean sea level pressure PP (35Mb)	Sources	Stations
Daily cloud cover CC (28Mb)	Sources	Stations
Daily humidity HU (31Mb)	Sources	Stations
Daily snow depth SD (81Mb)	Sources	Stations
Daily sunshine duration SS (28Mb)	Sources	Stations
Daily mean wind speed FG (30Mb)	Sources	Stations
Daily maximum wind gust FX (23Mb)	Sources	Stations
Daily wind direction DD (21Mb)	Sources	Stations
Non-blended ECA dataset		
All elements (1003Mb)	Sources	Stations
Daily maximum temperature TX (169Mb)	Sources	Stations
Daily minimum temperature TN (176Mb)	Sources	Stations
Daily mean temperature TG (162Mb)	Sources	Stations
Daily precipitation amount RR (246Mb)	Sources	Stations
Daily mean sea level pressure PP (33Mb)	Sources	Stations
Daily cloud cover CC (26Mb)	Sources	Stations
Daily humidity HU (28Mb)	Sources	Stations
Daily snow depth SD (72Mb)	Sources	Stations
Daily sunshine duration SS (26Mb)	Sources	Stations



Linnæus University

School of Computer Science, Physics and Mathematics

SE-391 82 Kalmar / SE-351 95 Växjö

Tel +46 (0)772-28 80 00

dfm@lnu.se

Lnu.se/dfm