The analyze of the use of software programming in the design process of a information visualization weather data poster

Luis Carli
FAU USP
info@luiscarli.com

Carlos Zibel Costa
FAU USP
czibel@globo.com

ABSTRACT
We approached a infovis design study from a graphical design point of view, focusing on the rationale and iterative nature of the process. We show how the layout solutions were developed and we present some code abstractions, methods and objects, that were created for helping representing layouts on code and iterating them. These abstractions can in the future be developed into a full toolkit focused on the layout aspects of the infovis graphic design process.

KEYWORDS: Information Visualization, Graphic Design, Layout, Design Process.

Introduction

Moere (MOERE; PURCHASE, 2011) says that information visualization is a design discipline, meaning that all aspects of a visualization are designed, explicitly or implicitly, consciously or unintentionally. He puts that becoming aware of the rationale behind design choices is not only interesting, but can benefit the information visualization community as a whole. Here we briefly show the graphic design rationale and process from a weather infovis.

From our background in architecture and design we analyzed the design process of a information visualization, focusing on the iterative layout development. During the process we used and created some abstractions that helped code and test layout solutions. These methods and objects reflect some of the necessities that we had for developing code that contributed to the reflective conversation (SCHON, 1992) of our design process. Target problems and design requirements (Fig.01).

The objective of the studied visualization (figure 1) is to help a reader see how the temperature change in his city or in a city that he plans to visit, during the course of the day and along the year, allowing him to check what to expect in a certain moment of the day/year according to weather data from the last years. Showing when the temperature normally drops or rises, how is the variation from year to year, when the difference of temperature is more constant from year to year.

In a graphic design point of view we wanted to follow Tufte (TUFTE, 2001) data-ink ratio, where most of the ink on the graphic is for presenting new information. To do that we searched layout solutions with fewer labels as possible, while still communicating enough information for a new reader to understand the visualization. To better focus on the encoded data, we also wanted a balanced relation between the space occupied by the graphs and the white space around it, while keeping the graphs on a logical grid, so they could be easier to focus on and read (MULLER-BROCKMANN, 1996).

As the visualization is intended for an average reader, we decided that its size should fit inside a 1000x500 pixels frame, with this we have an image that can be fully readable in most computers without scrolling or zooming, and that can be easily shared and printed. The minimum and medium font size is determined by

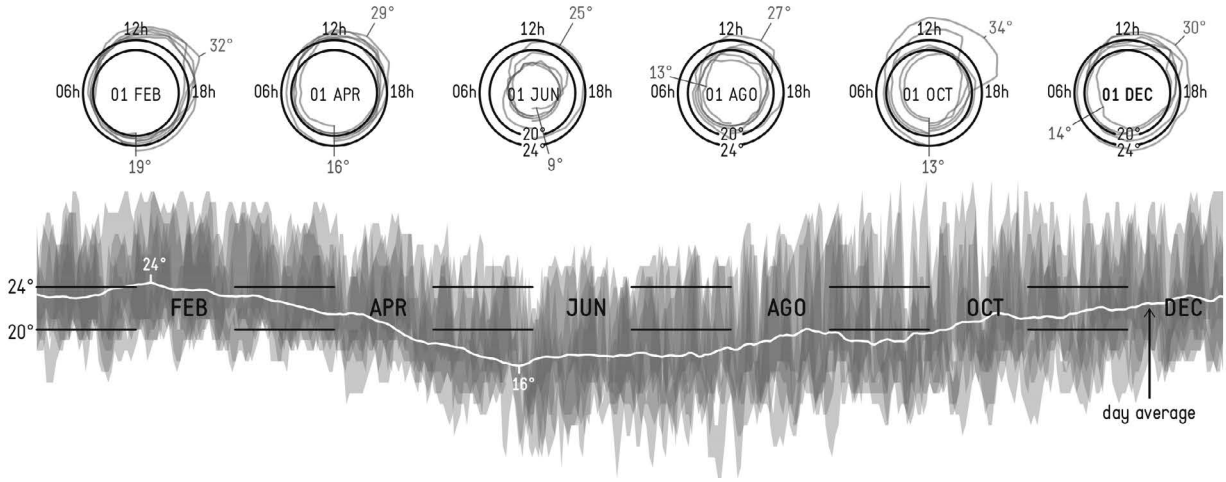

Fig. 1. Visualization of the weather temperature on São Paulo from 2007-2011.

the size, support and average reading distance of where the visualization is going to be displayed. The minimum font size limit the amount of information displayed that is comfortable to read.

To fulfill the defined target problems we needed to:

- Show year and day temperature information at the same time
- Keep the visualization inside a frame limit of 1000x500 pixels.
- Use a font size for labels that range from 15px to 20px.
- Show the minimum amount of labels while keeping the vis understandable for a new reader.
- Have a good balance between empty spaces and graphs.
- Allow the reading of how constant the temperature variation was from year to year on a same date/time.

Visual enconding and code abstractions

To fulfill these requirements, we first decided that we needed two parts on the visualization. One for how the temperature changed along the day, showing some days from the year; and one for how the temperature changed along the year. We iteratively developed several alternatives for the label placement, color, font and stroke size for the visualization, while we tried to satisfy the presented requirements.

We started by doing some hand sketches for the overall composition, then continue to develop the design on code using real data. We developed the visualization on Objective-c App Kit, with some small modifications of the outputted pdf done on the Adobe Illustrator Software, like the application of the gradients (Fig. 02).

To help with the iterative layout development we used and created some code abstractions. We worked with nested views objects that held the drawing code of a specific part of the visualization, each view had their own independent coordinate origin and all the drawing that they did was in relation to its origin and to its size, so manipulating and resizing the views didn't distorted its geometry, figure 2 shows how the layout was divided in views. Some of the drawing code was also related to the computed size of the text labels, so if the size of the font type changed, the geometry which depended on it also accommodated and redrawn accordingly.

We made a shared dictionary for holding global variables, whenever a color or a number was added to the dictionary on the code, the executed software automatically generated an user interface for controlling this variable. These global variables were used to iteratively test variations on, for example, the font and stroke size, as the perception of the visualization

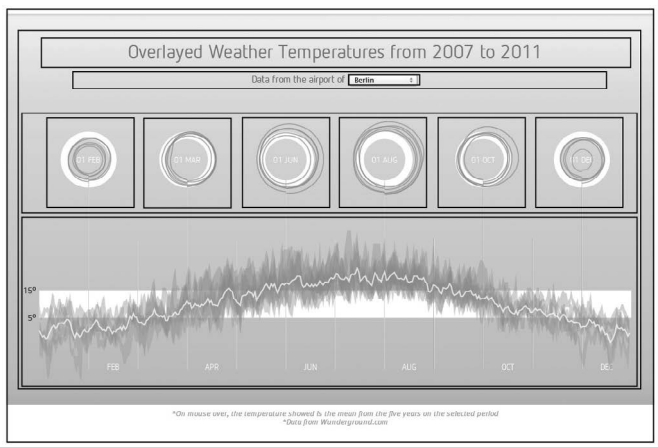

Fig. 2. Visualization with its layout views overlayed.

changed a lot with small transformations on some of its graphic properties.

For positing the views, instead of directly setting their x, y, width and height we used methods that compute these values based on layout relations. For example, a method that position the top of view 'J' 15 pixels from the bottom of view 'K', a method that create a border space around a view that all the other methods will use to calculate new position and size values, a method that says that the right side of the view 'K' will expand until it is 20 pixels from its superview right side. By using methods that abstracted layout relations we were able to do fast iterations and represent how the views were composed more easily in code.

When the visualization was rendered the software executed two different passes through the nested views tree. In the first one it calculated the size and position of the views based on its siblings and superviews values, using the methods described in the last paragraph. In the second pass, the drawing code of the view was executed in relation with the view size and the font size. By using these two passes we could assign some variables from the layout relations and the drawing code to the shared dictionary, and test their variation iteratively using an automatically generated UI (Fig. 03).

In figure 3 there's some of the layout iterations for the day temperature view, besides the data it needed to show the reader which day the information was from and how the time and temperature was mapped. We plotted the temperature radially using five years of data, the line has some opacity so the different years can stack. Instead of drawing labels for each increment of temperature we marked only two ranges, that define a pleasant temperature in the visualized city. It's then easy to see when the temperature stays inside or outside the comfort zone for that city.

The maximum and minimum temperature for the day is displayed with a line like a tick on a clock, which help to see when on the day were those temperatures. The hour labels were placed on the sides and on the top, leaving

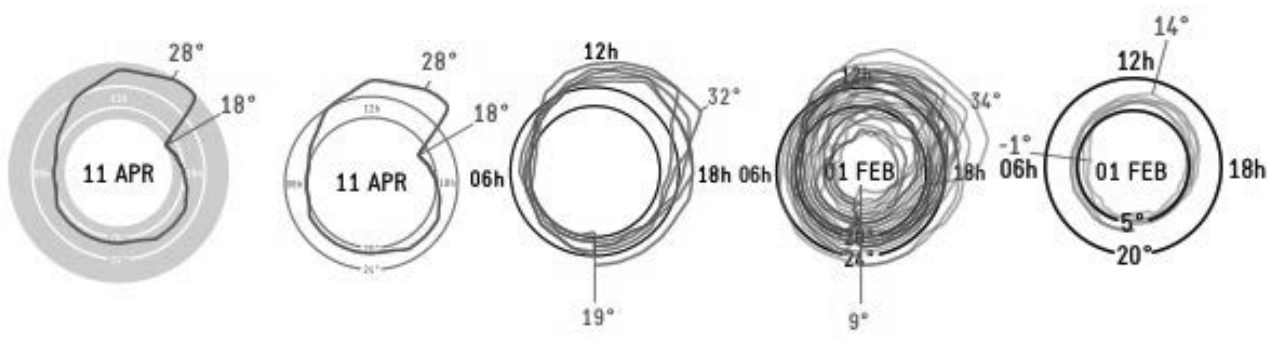

Fig. 3. Layout iterations on the day-temperature view.

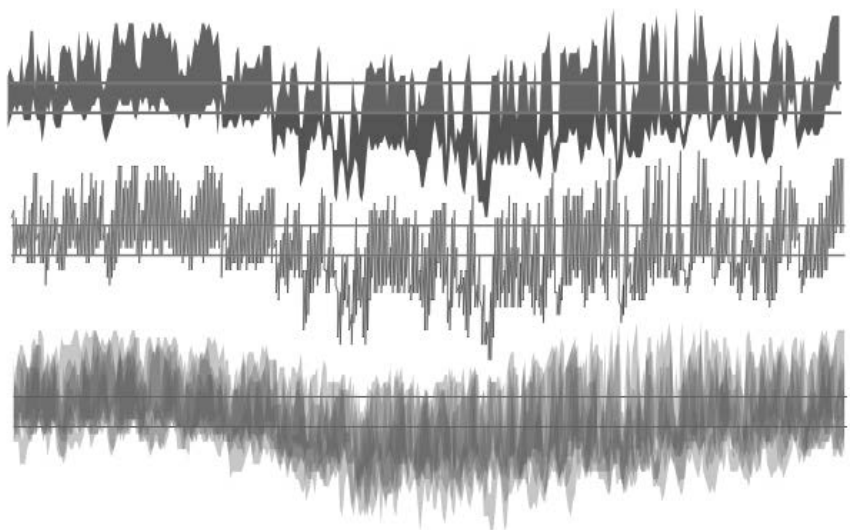

Fig. 4. Iterations on the year view.

the bottom to the temperatures labels. As the view is used six times, when the minimum temperature label overlay the range temperature label we opted to hide the range label, leaving the image with less clutter while the info can still be deduced from the siblings views (Fig.04).

Figure 4 shows some iterations of the year view, we end up showing the five years of data using layers with opacity on top of each other. The same temperature range labels from the daily views are used here, but for achieving most info with less labels we dashed the line and wrote the months info on the created spaces. On the lines and areas that represent temperatures we applied red and blue colors, forming a gradient which works as a label and also as a aesthetic refinement.

Conclusion

This paper showed a graphic design approach to a weather temperature visualization, were the focus was to find a good layout with a big data-ink ratio. We ended up developing some code abstractions to help iterate quickly, they also helped to reduce the translation between the mental model that the designers had about the visualization and its code representation.

The main code abstractions were:

- Nested views tree for the visualization main geometries groups, with two render passes, one for the layout and one for the drawing.
- Methods for setting the x, y, width and height properties of the views through the creation of relations between views inset size, and sides (top, bottom, left, right) distance to others views and superviews.
- Automatically generated UI based on a variable type (color, font, number, etc) for tweaking the drawing and seeing results in real time.

These code abstraction can, in a future work, be developed into a full toolkit to help in the graphic design and layout process of information visualizations. By making it faster to visualize and iterate through layout solutions, while using abstractions that make the code instructions more similar to the models that a graphic designer use, we think this toolkit could assist in creating and visualize spaces of layout solutions.

Also we think that bringing together graphic design process and information visualization, can help in the development of new layout solutions and in the scientific evaluation and testing of these solutions.

Acknowledgments

We thank FAPESP for funding this research

Bibliography

MOERE, A. V.; PURCHASE, H. On the role of design in information visualization. Information Visualization, v. 10, n. 4, p. 356–371, 14 out. 2011.

MULLER-BROCKMANN, J. Grid Systems in Graphic Design. Traducao. [S.l: s.n.].

SCHON, D. A. Designing as reflective conversation with the materials of a design situation. Research in Engineering Design, v. 3, n. 3, p. 131–147, 1992.

TUFTE, E. R. The visual display of quantitative information. Traducao. [S.l.] Graphics Pr, 2001.