AnGraf: Creating Custom Animated Data Graphics

AnGraf
Creating Custom Animated Data Graphics

Daniel A. Dias

TR-05-99

Table of Contents

1. Introduction

The graphing of data sets can often provide a great deal of insight beyond numerical methods. Consider the data in Table 1. Both data sets are statistically similar, but by looking at the graphs of the data sets in Figure 1 it becomes immediately clear that they are unique. The first data set is quasi-linear while the second data set is a curve. This difference, while not immediately recognizable in the table, becomes clear in the graph.

Table 1: Data sets with identical statistical figures.[1]

Data Set I			Data Set II		Statistics
x	*y*		*x*	*y*	
10.0	8.04		10.0	9.14	N = 11
8.0	6.95		8.0	8.14	mean of X's = 9.0
13.0	7.58		13.0	8.74	mean of Y's = 7.5
9.0	8.81		9.0	8.77	equation of regression line: Y = 3 + 0.5X
11.0	8.33		11.0	9.26	t = 4.24
14.0	9.96		14.0	8.10	Sum of squares X - $\overline{X}$ = 110.0
6.0	7.24		6.0	6.13	regression sum of squares = 27.50
4.0	4.26		4.0	3.10	residual sum of squares of Y = 13.75
12.0	10.84		12.0	9.13	correlation coefficient = .82
7.0	4.82		7.0	7.26	r^2 = .67
5.0	5.68		5.0	4.74	

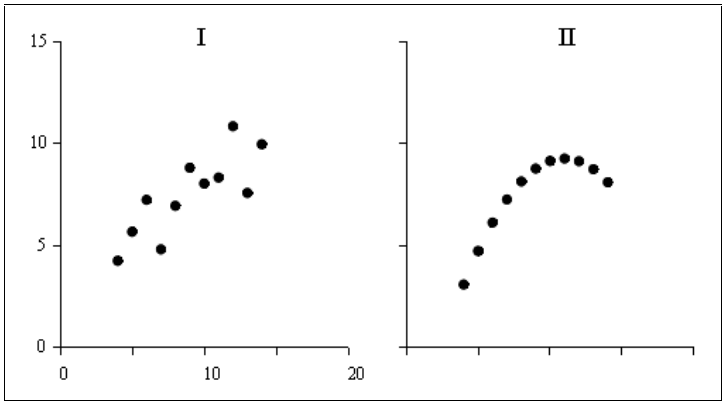


Figure 1: Graph of data sets I and II**.** ***Generated by AnGraf.***

The preceding example highlights how graphs can illuminate data by showing trends and relationships that are otherwise difficult to see. In large, multi-variate data

[1] Data taken from F.J. Anscombe, "Graphs in Statistical Analysis," *American Statistician*, 27 (February 1973), 17 – 21. Quoted in: Tufte, Edward. The Visual Display of Quantitative Information (Chesire, CT: Graphics Press, 1983), p. 13.

sets the need for data graphs becomes even more obvious. Imagine trying to study the 333 million daily transactions conducted on the New York Stock Exchange without using any graphs.[2] The problem of creating data graphics according to a user's needs is an important issue that will become even more pressing as technology inundates society with more information.

The most common tools currently used to generate data graphics are charting packages offered by spreadsheet programs such as Microsoft Excel and Lotus 1-2-3. With these packages the user may choose the graph type that will be generated for the data from a group of standard types. The packages then display the graph for limited editing. Microsoft Excel 97's Chart Wizard, for example, has more than fifty "standard" graph types which can be chosen through the dialogue based interface shown in Figure 2.

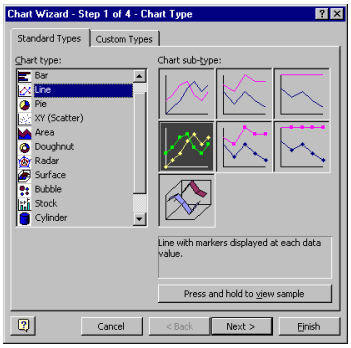


Figure 2: A dialogue window of Microsoft Excel 97's Chart Wizard.

If, however, a user wanted to create a graph with multiple sub-graphs on the same page or an animated graph, the Chart Wizard would be of little use because it does not offer either of these graph types. Although these types of graphs could be included in a future release, another example could certainly be found that was not. The fundamental problem in these graphing packages lies not in the number of graph types offered but in the underlying design. The paradigm of providing standard graphs fails to offer the flexibility users may need to design customized data graphics.

The development of the spreadsheet solved a similar problem for calculating applications. Before its introduction, different programs would be created to perform "standard" calculations such as balancing a checkbook or determining a company's inventories. Each program, albeit useful for its specific function, could only solve a single problem and nothing more. Consequently, if a user had a checkbook and inventory program but wanted to calculate a company's balance sheet, the software could not provide assistance despite the similarities in the calculations for the three problems.

The spreadsheet solved this dilemma by replacing the focus on the specific function of the calculation with a generic, editable interface for customized calculations. The spreadsheet itself could not balance a checkbook, but offered the fundamental tools to construct such an application. While constructing calculating applications may be more time consuming during setup, the flexibility to create a wide range of applications within one program more than compensates.

---

[2] Schroeder, W., Martin, K., Lorensen, B., The Visualization Toolkit (Upper Saddle River, NJ: Prentice Hall, 1998), p. 3.

A similar dilemma now exists within data graph creation. While packages like Chart Wizard are useful to illustrate particular data graphs that conform to their types, they are unable to generate graphs that may have similar underlying structure but diverge slightly from these types. Understanding how the spreadsheet's structure provided more flexibility than previous programs gives insight into how to address the limitation of current data graphing applications.

The data graphing program "AnGraf" was created to overcome these limitations. It is based on the principle that graphs can be built using primitives such as axes and points. Like a spreadsheet, it replaces the focus on the specific, final data graphs with a focus on the generic tools, or primitives, used to build customizable data graphics. In AnGraf's editing window a user adds primitives and relations between primitives to build a "frame" of the desired graph. This is then translated into OpenGL to create a final data graphic.

Take Figure 3, a graph generated by AnGraf based on an original template data graphic by Charles Joseph Minard(1781 – 1870). It depicts the severe casualties in Napoleon's army both on its march to Moscow and its return in the bitter cold winter. Four distinct variables are represented on this graph: longitudinal position (y-axis), latitudinal position (x-axis), troop size (by line width), and temperature (by color). A fifth variable (animated time) can be used to show the army's movement. Edward Tufte claims that Minard's original "may well be the best statistical graphic ever drawn."[3] Until Chart Wizard offers a "Minard" template, this sort of graph cannot easily be produced with current technology. Using AnGraf, however, it can simply be created by binding various graphical primitives with one another and associating these primitives with data.

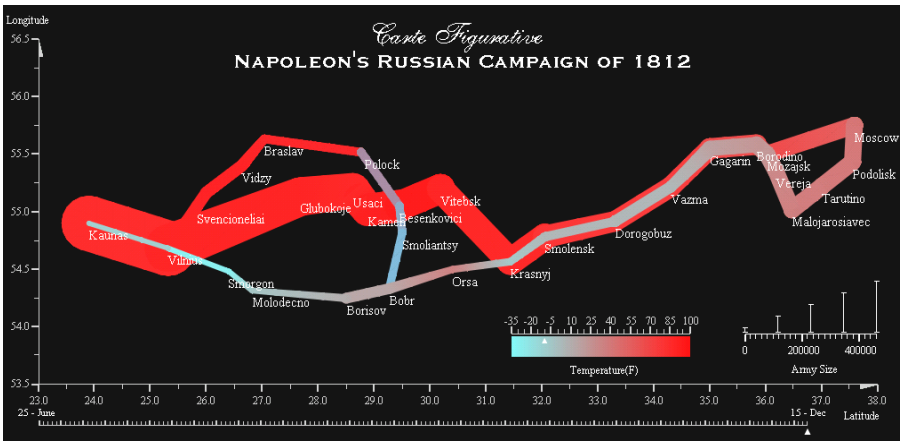


Figure 3: This is a replication of the classic graph by Charles Joseph Minard[4] generated by AnGraf.

[3] Tufte, Edward. The Visual Display of Quantitative Information (Chesire, CT: Graphics Press, 1983), p. 40.

[4] E. J. Marey, *La Méthode Graphique* (Paris, 1885), p. 73. Data obtained from Mark Derthick in the SAGE project.

Flexibility and variation will continue to become more important in the visual representation of data. AnGraf, unlike any current program, responds to this need. As the spreadsheet developed to provide adaptability in calculations, AnGraph offers an efficient, generic interface which enables the user to create customized static and animated graphs.

2. Previous Work

2.1. Visualization Systems

Scientific visualization systems such as AVS from Stardent, IBM's Visualization Data Explorer, and the Visualization Toolkit[5] can create custom graphics to fit almost any need. These coding libraries are a common alternative for users who want to create a graphic not available from standard graphing tools. Myers, Goldstein, and Goldberg, however, make the following critique of this solution[6]:

1) Users generally have to write code to "specify the desired graphic." This requires some programming knowledge.
2) Using a charting routine from one of the libraries generally requires "specifying lots of complex parameters."
3) "It is not possible to directly manipulate the generated pictures to change the display …"

Consequently, the programming tools still do not allow the average user "easily" to create customized graphs.

2.2. Gold Project

Brad Myer's Gold project,[7] Graphs and Output Laid-out by Demonstration, investigates the use of demonstrational techniques to create data graphs. In Gold a user is able to "free sketch" the desired graph design, and the system attempts to determine an appropriate chart to display. Having the system interpret the sketch, however, restricts the true customizability of graphs and possibly prevents the user from generating the preferred graph.

2.3. SAGE Project

Steven Roth's SAGE[8] project, which stands for System for Automated Graphics and Explanation, is a knowledge-based system for automatically designing data graphs.

---

[5] Schroeder, W., Martin, K., Lorensen, B.. The Visualization Toolkit. Upper Saddle River, NJ: Prentice Hall, 1998.

[6] Myers, Brad A., Goldstein, Jade, and Goldberg, Matthew A.. "Creating Charts by Demonstration," *Proceedings CHI'94: Human Factors in Computing Systems.* Boston, MA, Apr. 24-28, 1994. p. 107.

[7] Ibid., p. 107.

[8] Roth, S.F. and Mattis, J., Data Characterization for Intelligent Graphics Presentation, *Proceedings of the Conference on Human Factors in Computing Systems (SIGCHI '90)*, Seattle, WA, April 1990, pp. 193-200.

For example, SAGE can distinguish between quantitative, ordinal, and nominal sets and choose an appropriate graphical representation. SAGE also has a tool called SageBrush which allows a user to "sketch or assemble graphical elements to create designs and map them to data."[9] Though this tool influenced AnGraf's design, it nevertheless has limitations. First, since SageBrush uses SAGE to generate its graphs, SAGE's knowledge-based system handles the actual layout of the graph. The user consequently has relatively limited control on the final output. While this could be convenient in many situations, it does not offer a solution when SAGE's appropriate placement is different from what the user wants. Secondly, SAGE, and consequently SageBrush, do not support animated graphs.

3. Approach

3.1. Primitives, Data Placement, and Graph Design

AnGraf is based on the idea that data graphs should be built from small generic blocks for maximum flexibility. AnGraf calls these blocks "graph primitives." There are two basic types of graph primitives. First, there are markers which represent data. There are three marker primitives: graph points, lines, and labels. Second, there are ranges which define the data's range. There are four range primitives: graph axes (positional ranges), color ranges, size ranges, and text ranges. In AnGraf the distinction between markers and ranges is significant because only ranges can be directly associated with data sets.

The logic behind this separation is that markers themselves do not directly convey data. Consider the relationship between a data point and a graph axis by looking at a simple scatter plot of a single data point without axes in Figure 4. Almost no information is conveyed by this graph because the graph point itself carries no data. It merely represents a point in two dimensional space. It is a point's position to a relative range, such as a graph axis or another appropriately placed graph point, that enables it to represent data.

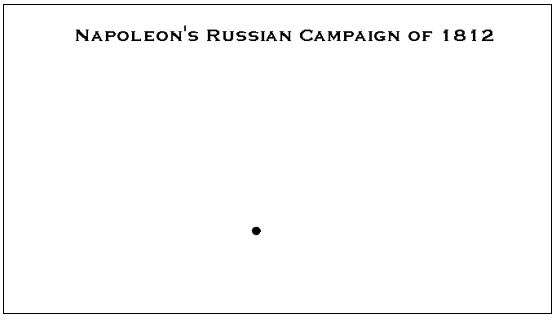


Figure 4: Single data point without axes. *Generated by AnGraf.*

---

[9] Roth, S.F., Kolojejchick, J., Mattis, J., and Goldstein, J. Interactive Graphic Design Using Automatic Presentation Knowledge. *Proceedings of the Conference on Human Factors in Computing Systems (SIGCHI '94)*, Boston, MA, April 1994, pp. 112-117.

Figure 5 shows the data point from Figure 4 with associated axes. Now one can clearly see the data value of the point. The graph axes, as a positional ranges, allow for the point to represent a distinct data value. Consequently the range primitives, which in this case are graph axes, could be seen to carry the data while the marker primitive, in this case a graph point, merely represents it.

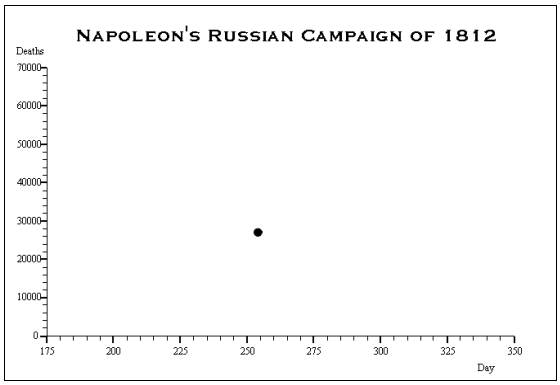


Figure 5: A single data point with appropriate axes. *Generated by AnGraf.*

Adding more data points to the scatter plot allows a marker's relative position to another marker to convey the data's relative value. Again, however, this positioning can be seen to depend on the range over which the data is defined, not on the marks themselves. Figure 6 shows the full scatter plot of Figure 5 with two different horizontal and vertical positional ranges, demonstrating that the significance of relative spacing between markers depends on the range not the markers.

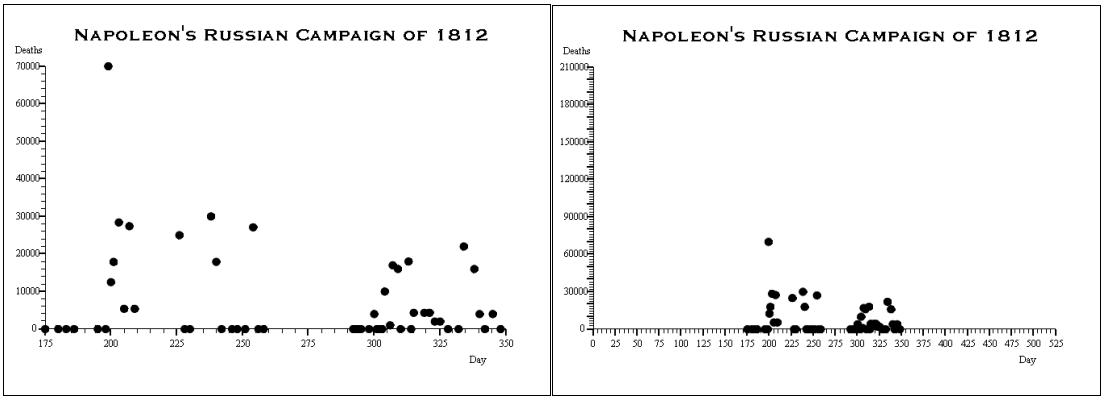


Figure 6: Full scatter plots of the death vs. time in Napoleon's campaign with two different ranges.

To account for this, AnGraf only associates data with ranges. In order for a marker to represent the data over a range, it must be bound with that range. In Figure 5 and Figure 6, for example, a graph point is being used to represent data over vertical and horizontal axes. Therefore in AnGraf's structure the marker, or graph point, should be bound with the ranges, or graph axes, to create the scatter plot. Figure 7 graphically illustrates the bindings between primitives used to create Figure 5 and Figure 6.

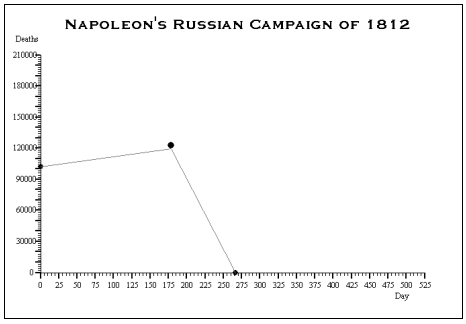


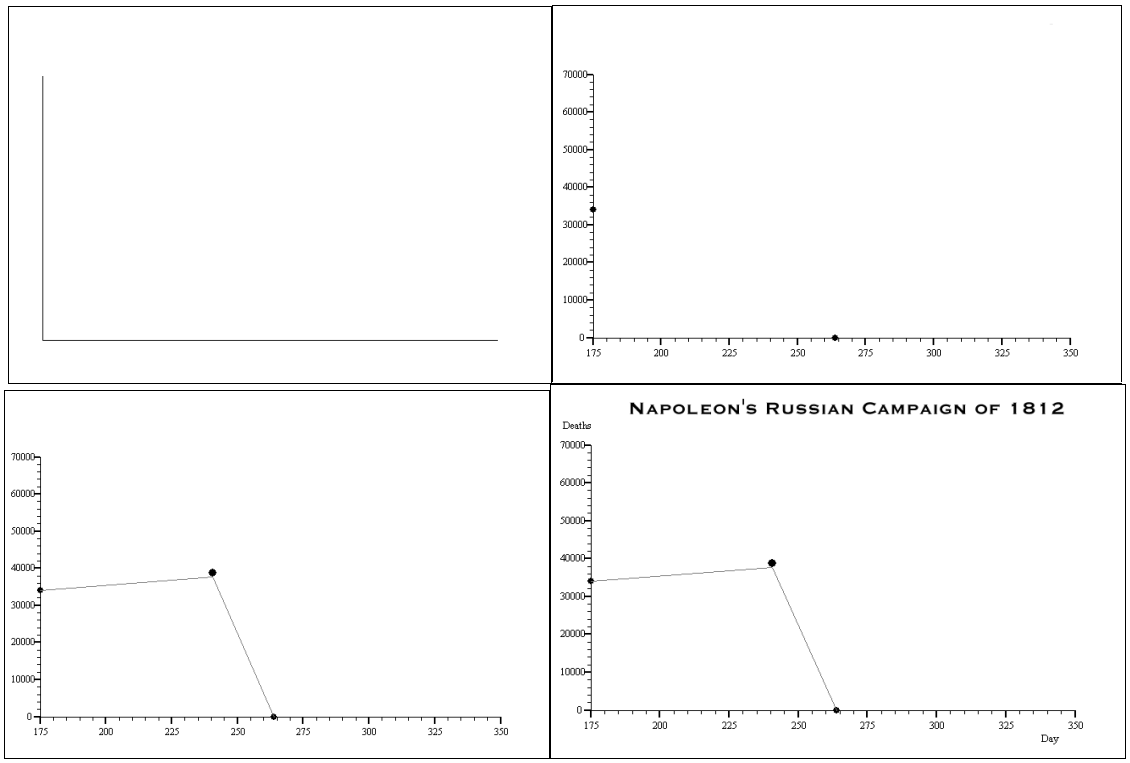
Figure 7: Bindings between a graph point and graph axes to create Figure 5 and Figure 6**.**

AnGraf's base structure for an axis is not a two dimensional axis but a single generic graph axis. The separation of the horizontal and vertical axis in AnGraf's framework is significant because it allows the user to design data graphs that have points positioned only vertically or horizontally. Moreover, a two dimensional axis can be created by positioning two axes together, and each axis would position bound primitives in its respective dimension. This approach allows for a great deal of flexibility in graph design by providing the user individual control over each of the two dimensional axes. Furthermore, if the ability for an axis to be positioned in a depth plane were added a three-dimensional axis would merely be composed of three appropriately positioned graph axes that handled each dimension.

While the system of binding primitives to build graphs provides flexibility, it follows certain rules within a graphing system to prevent the occurrence of impossible graph schemes like a graph point bound to more than one vertical axis. To address this, AnGraf prevents such an infeasible connection from occurring by limiting the binding process. The stipulations for appropriateness are not very complex or restrictive, and each primitive is queried before it is bound as to whether the particular binding is appropriate.

3.2. Creating a Scatter Plot in AnGraf's Structure

Here is a brief example of how AnGraf's model would represent the design of the two dimensional scatter plot seen in Figure 5. Two graph axes, a horizontal and a vertical, are positioned together to form a two dimensional axis. Time data and casualty data are then associated with the respective horizontal and vertical axes. This creates a two dimensional graphing surface. A graph point is then bound with the time data set on the horizontal axis and the casualty data set on the vertical axis to "represent" the data over the positional range. Graph labels without bindings are then added to create a title and axes' labels. Finally the data is applied to this design to create the final graphic. A storyboard of this process is seen in Figure 8.



Now applying data to the design creates the final graphic.

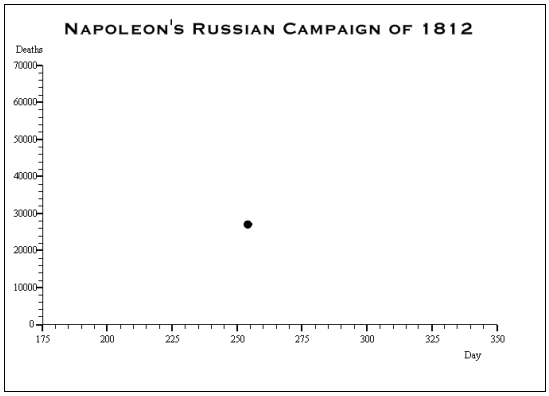


Figure 8: Storyboard for creating the design for the final graphic shown in Figure 5.

4. Implementation

4.1. AnGraf's User Interface

The creation of customizable graphical representations of data is an issue of both design and application. In the design stage the user must decide which aspects need to be highlighted and what displays should be used to illuminate the data. The application of this design involves accurately representing the data in the manner chosen. AnGraf consequently has a two window design. One window is for designing the data graphs which is accomplished by editing the layout and properties of the graph's primitives. The

other window is for application of this design to the data and displaying the desired graph.

4.1.1. The Editing Window

The bare editing window could be considered the "canvas" for the graph. The actual creation of a graphic begins by adding primitives to the canvas through a menu based system. Once a primitive is on the canvas, it can be adjusted to fit a user's needs. It can be dragged and dropped to different locations, and its size and color can be customized. Changes made in the editing window are directly translated to the graph's final display. This WYSIWYG designing process gives the user ultimate control of the final graph. In the cases where some consistent positioning between primitives is desired, AnGraf's editing window incorporates tools to align and size graph primitives by other primitives. An example of this feature is seen in Figure 1 on page 1 where two graph axis are identically sized and vertically aligned.

One difficulty in editing animated data graphs is that two dimensional graphs become three-dimensional over animated "time." To place animated graph editing back into the two dimensional realm, the layout of primitives in AnGraf's editing window represents a generic "frame" from which the final frames of the animation will take their form. A frame is a single graph generated during an animated graphical sequence; it is the equivalent of a single picture in a movie. For instance, in a scatter plot animation of a single variable there would be only one graph point per frame. The full animation would successively show each point in the graph, or "frame", to create the full animation, or "movie," of the graph. So the design in the editing window would only show the graph axis with a single graph point primitive in this case.

A dialogue allows a user to select a data column as an "animating" context for the graph. Graphs can be animated over any data column. After a user selects a column, all the rows of the loaded data are sorted so that the selected data is in ascending order. This allows the animation graphing process to merely entail looping through all the rows and displaying the corresponding graph frames of the data.

If no data column is selected for animation, AnGraf will create a static graph of the data which is simply all of the graph frames combined into a single graph. Using AnGraf's static graphing capabilities, graphs such as scatter plots can be generated by simply binding graph points to graph axes without selecting animation.

A design in the editing window, consequently, will result in one of two types of graphs depending on whether animation is selected. If the graph is being animated, the output will be a frame with markers positioned appropriately from the editing frames generic model. An example of this is shown in Figure 9, where the Napoleon Campaign data is being animated by the time.

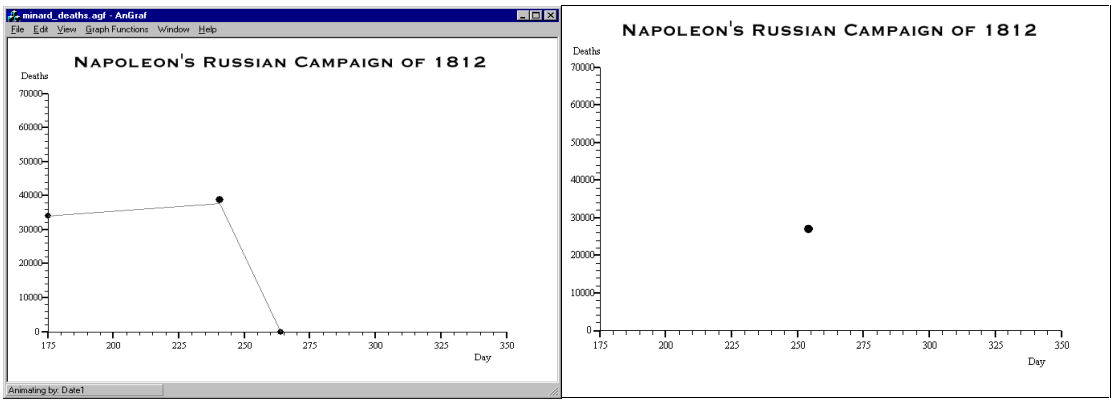


Figure 9: The editing window's frame when animating over a data set (left) represent a generic frame of the final graphic (right).

If no animation were selected however, the output from this editing window would be the scatter plot previously shown in Figure 6 on page 6. Therefore, even though only a single graph point primitive is shown in the editing window, it represents all of the final graph points for the data set in the final data graphic. To reinforce this, the editing window from Figure 9 without an animated data set and its output are shown in Figure 10. The lower left side of the status bar in the editing window displays whether the graph is being animated by a data set.

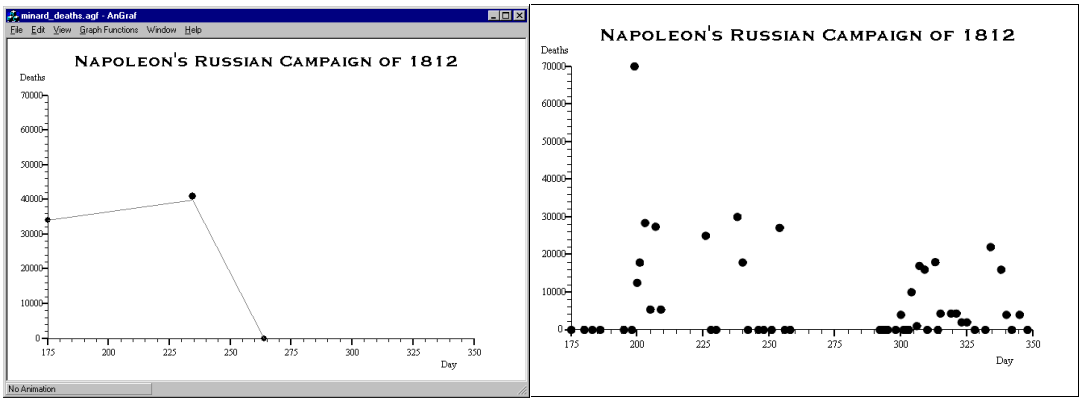


Figure 10: The editing window's frame when not animating over a data set (left) represents a generic point of all the points to be generated in a scatter plot (right).

The editing window uses context menus to associate data with range primitives by having the user select data columns through either a menu or a dialogue. The process varies because each primitive handles its own context menu, and will provide data selection options according to its needs. A color range, for instance, needs to be provided with start and end colors in addition to a data column, so it uses its own dialogue. A graph axis, on the other hand, just needs to associate data with the axis, and it uses a menu-based approach. The difference between these two methods is seen in Figure 11.

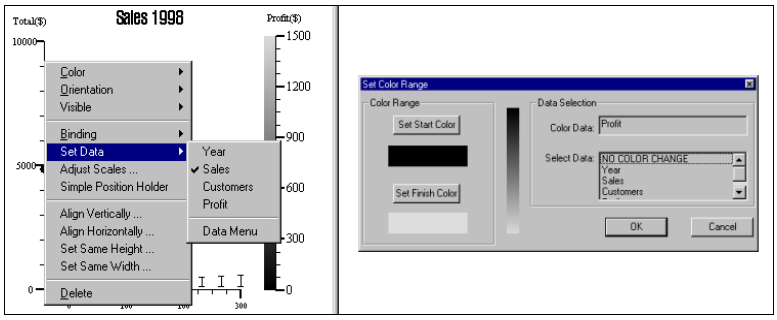


Figure 11: Example of the customized approaches to data selection. The left pane is an example of a graph axis' data selection. The right pane is the color ranges data dialogue.

The addition of a data column to a range primitive creates a scale for the data. The scale serves two purposes. Internally, it allows each data column to have an independent scale over the range, and externally, it enables the user to customize displayed scales independently. Figure 12 shows a graph axis with two data sets and two associated scales on the vertical axis.

All graph scales are adjustable through a dialogue that can be accessed from a primitive's context menu. The dialogue allows the user to change both the internal representation and the visual presentation of the scale for a particular data set. The only significant values for internal scale representation in AnGraf are the maximum and minimum values. These variables define what range of the data the axis will represent, and consequently where data will be placed. The dialogue also allows the user to change almost all the properties in the visual display of a scale's tick marks and annotation. Furthermore, when more than one data set is associated with a primitive, it provides options for the user to lay out multiple visible scales or hide scales so that format meets the user's needs.

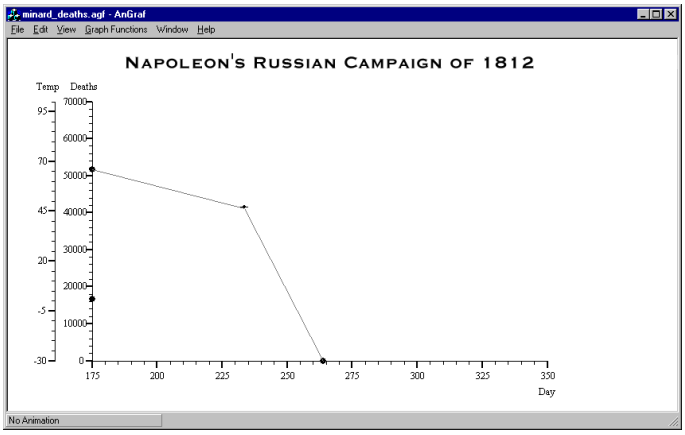


Figure 12: A graph axis with two data sets and two associated scales on the vertical axis**.**

All data primitives in AnGraf, except for a graph axis, can have a maximum of one data set associated with them. A graph axis can have as many associated data sets as a user desires. This allows for the same two dimensional axis to handle any number of distinct data points. The addition of a data set to a graph axis causes a change in the

editing environment. A node is created on the axis for the new data set so that other graph primitives, especially points, can specifically bind with the new data. As mentioned earlier, a new graph scale is also added to the axis for the data set. Consequently, an axis with two data members will have two scales and two binding nodes as seen in Figure 12.

When a user wants to display a marker without a corresponding range in the final graph, the WYSIWYG implementation could conflict with AnGraf's structure of only being able to associate data with range primitives. An example of this would be in a color range where the user would like to have a single marker reflect a particular color's data value. An example of this is shown in Figure 14. To create this effect the marker is being positioned by a single horizontal axis primitive within the color range that has the identical data and scale to this range. A view of this in the editing window would look like Figure 13. This primitive is required by AnGraf's structure, but

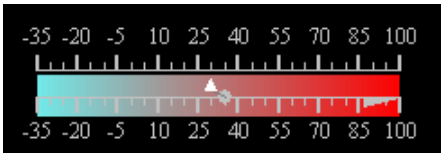


Figure 13: Editing windows view of an axis used to create an inlaid marker.

Figure 14: Color range with an inlaid marker.

including this in the final graph does not create the desired effect. To address this and other such situations, all primitives have a visibility option which allows a user to select whether a particular primitive is displayed. So to create the inlaid marker in Figure 14, the graph axis would exist to provide the graph marker with horizontal positioning information, but the axis' visibility property would not be set and consequently would not be displayed.

In addition to handling the layout of primitives, the editing window provides an interface for binding primitives together. By selecting a primitive's context menu function to add a binding, a line is drawn between the primitive and the mouse cursor until the user clicks on the desired primitive with which to add a binding. If the binding selected is legitimate, a line will be drawn between the two primitives in the editing window to signify the binding. If it is illegitimate like an attempted binding of a point to two vertical axis, no binding will be formed and no line will be drawn.

Since data is associated with ranges, the binding process is consistent for all ranges. Binding a marker to a color range works in the same manner regardless of whether the marker is already bound to an axis. No data set information needs to be passed from the marker to the range because the range handles its own data set. If data were associated with markers, however, in order to have a marker represent both a position and color range each data set would need to be chosen to be bound with the appropriate range.

Binding objects within the editing interface accomplishes two things. First, it establishes a relationship between graphical primitives within AnGraf's framework of ranges and marks. Second, it allows bound objects to move as a logical group. When a user creates a two dimensional axis by binding a horizontal and vertical axis together, AnGraf automatically places the two primitives together. Since they are considered a logical group by their binding, moving one primitive causes the other primitive to move. In this way a two dimensional axis will not visibly split by moving one axis. Likewise, if a user created a two dimensional axis bound with a graph point, since all the primitives are bound together, the entire graph could be moved by moving a single primitive. While this feature may be useful in some situations, it can be a limitation in others. To compensate for this, a keyboard override can be pressed to have a primitive move independent of its bound members.

Since information such as data sets associated with primitives are not actively displayed in the canvas, AnGraf's status bar attempts to provide information on primitives and the state of the graph to the user in an unobtrusive manner. Figure 15 shows the editing frame for Figure 12 as the mouse is placed over a graph axis' data node.

AnGraf's editing interface is flexible for design, but using it to layout graphs can be tedious. One of the most obvious areas for improvement in the program would be to improve the user interface to reduce the time it takes to generate a graph. It is worth mentioning, however, that the program does have a mechanism that would allow a user to avoid repeating design work. Since data can be associated with primitives at any time and the loading of data into the editor is not required to be an initial step, AnGraf has the ability to create "templates." The layout of graph axes, labels, and points can be done without loading or associating data, and this file can be saved to be used later with different data sets. Clearly this is not a perfect solution, but it at least allows different types of graphs to be specified so that only data association would need to be done after importing a data file.

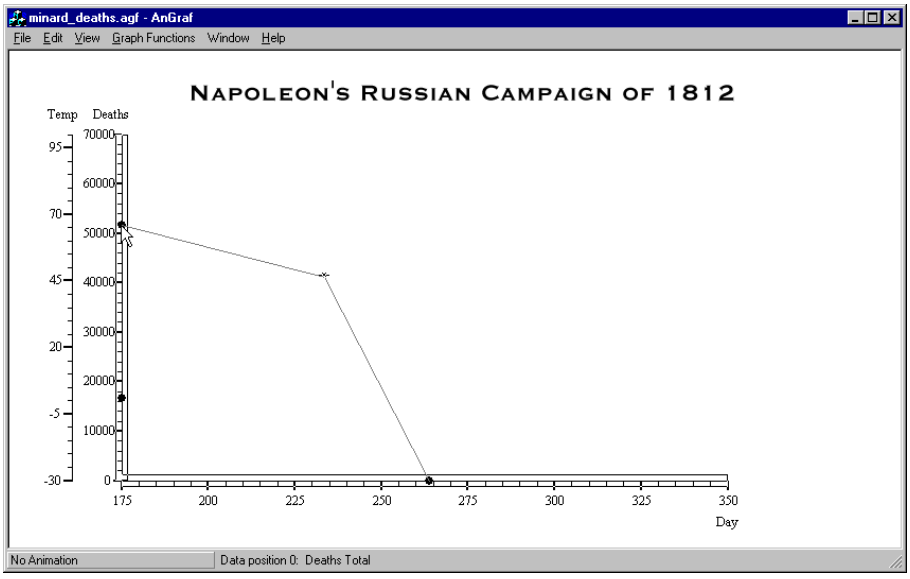


Figure 15: The editing frame for Figure 12 as the mouse is placed over a graph axis' data node.

4.1.2. The Graphing Window

When a user decides to display the graph designed in the editor, the representation of the data in the designed graphic is done in a separate window using OpenGL. OpenGL was chosen to display the animated graphs for three reasons. First, it has many features to create quality graphical output with high precision. Secondly, it is very expandable. As mentioned before the creation of three-dimensional graphs could theoretically be done by allowing graph axis to point in the depth plane of the page and then binding this type of axis with a normal two-dimensional axis. OpenGL, with its native support for rendering three-dimensional scenes, would easily facilitate this expansion. Finally, OpenGL, unlike Microsoft Windows' native windows, supports double buffering, a necessity for graphing animation.

The graphing window can either create static graphs or interactively display animated graphs. As mentioned before, its functional state is determined by whether a user selected a data column to animate over. In both cases, however, the actual graphing method is the same. Ranges, such as color ranges, size ranges and axes, are simply drawn to the screen just as the editing window displays them, but markers, such as graph points, are displayed in a three-step process. First the marker, which will be assumed to be a graph point for the example, checks for ranges associated with it. For each range, the point queries the primitive for information in the current graph frame. For example, if it is a graph axis, it will query for position information in the graph frame, or if it is a color range, it will query for color information. The second phase of the drawing process occurs when ranges handle these queries.

Since ranges are directly associated with the data, returning appropriate size or color information involves looking up the data value at the given frame, and appropriately scaling it. If markers were associated with the data, however, a range would have to query the marker for a data set before being able to scale it. By placing data with ranges, however, AnGraf avoids this intermediate step. For instance in a color range's case, since the primitive knows the data set, the start and finish colors, and minimum and maximum data values, the color range itself can linearly interpret the appropriate color for the current data value without querying any other primitives.

Once the ranges have returned their information, the third and final phase of drawing the appropriately positioned, sized, and colored point can take place. In the cases where ranges do not provide the necessary information for drawing, the settings in the editing window are used. Therefore, if no ranges are associated with a graph point, it will be drawn identically as seen in the editing window.

The graphing window also provides some tools to the user. When rendering an animated data graph a window that is similar to a CD or tape player's control panel is displayed. It allows the user to control the playback of the animation, from simple options such as starting, stopping, and pausing the animation to more complex options such as adjusting the playback speed. It also shows the animated variable's value for a given frame.

Another option offered by the graph window is to make an animated residual graph. This means that rather than showing just an individual frame's data, it shows all graph frame's data up to and including that frame. This can be very useful when

constructing an animated line graph because individual frames do not show the full line segment's data.

5. Examples

AnGraf allows a user to create custom static and animated graphics that were previously unattainable. The following examples highlight the flexibility AnGraf's structure of building graphs from primitives and the broad variety of graphs it can create.

5.1. Napoleon Campaign Data

Previous examples, like Figure 6 on page 6, have been constructed using data from Napoleon's Russian Campaign of 1812. As mentioned earlier, Charles Minard created a famous data graph of this campaign, and a final rendering of an AnGraf interpretation of it was shown in Figure 3 on page 3. Yet by tracing a possible evolution of this graphic from a simple deaths vs. time scatter plot to a final animated graphic, one can see how graphical primitives can be used in AnGraf's framework to create flexible, robust data graphics.

5.1.1. Creating a Multi-line Graph with an Overlaid Sub-Graph

The simple scatter plot of deaths vs. time that has been used in previous figures does not convey all the information about Napoleon's campaign. One distinct flaw is that it fails to provide any insight into troop movement or relative losses. A solution to this would be to redesign the graph such that the two dimensional axis represents longitude and latitude of troops positioning and then have a line graph show each of Napoleon's two main divisions paths. A third axis on the bottom, right of this two-dimensional positional graph will be added to create an overlaid line graph of troop size vs. latitudinal position. The decline of this line will show troop losses. To distinguish between the two divisions in each graph one will be colored gray and the other black. The resulting graph would look like Figure 16.

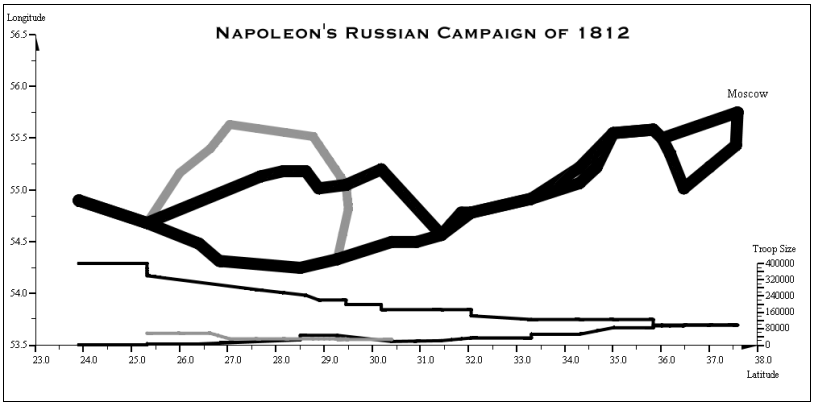


Figure 16: Simple positional and troop size line graphs for the two divisions in Napoleon's Campaign

The actual construction of this graph within AnGraf's editing window can be viewed as a ten-step process.

1) A horizontal and vertical graph axis are positioned together to form the two-dimensional data axis.
2) The vertical axis is associated with both divisions' longitudinal data. This creates two scales that are set to have identical maximum and minimum values, but only one graphical representation of the scales is displayed on the vertical graph axis.
3) The horizontal axis is associated with both divisions' latitudinal data. Again the scales for the data are set to have identical maximum and minimum values, and only one graphical representation of the scale is displayed.
4) A black and a gray graph point are added to the canvas and bound with their appropriate division's longitudinal data set on the vertical axis and latitudinal data set on the horizontal axis. Graphing this design would create a scatter plot.
5) To create a multi-line graph, the line option is set on each graph point which causes a line to be drawn between each pair of adjacent displayed points. Since graph points control the line widths that interconnect them in AnGraf, the circular graph points smooth non-aligned line edges.
6) A second vertical graph axis is then added and associated with both divisions' troop sizes. The two scales are handled in an identical manner to the previous axes.
7) This new axis is then positioned on the right hand side of the graph.
8) Two more black and gray graph points are added to be bound with their appropriate division's latitudinal data set on the horizontal axis and their troop size data set on the right vertical axis.
9) The line option is also set on these new graph points, but they are made slightly smaller than the other two in order to create a thinner line graph. In AnGraf, graph point size determines the line width of interconnecting lines.
10) Finally, graph labels are added to mark the different axes and title the graph.

5.1.2. Using Size Ranges to Create a Variably Sized Line Graph

This graph could still be made cleaner. By adding size range primitives associated with troop size and binding the positional graph points with these ranges, the point size of the troops' position, and consequently line width, could represent the number of troops instead of using the overlaid sub-graph. The result of this is seen in Figure 17.

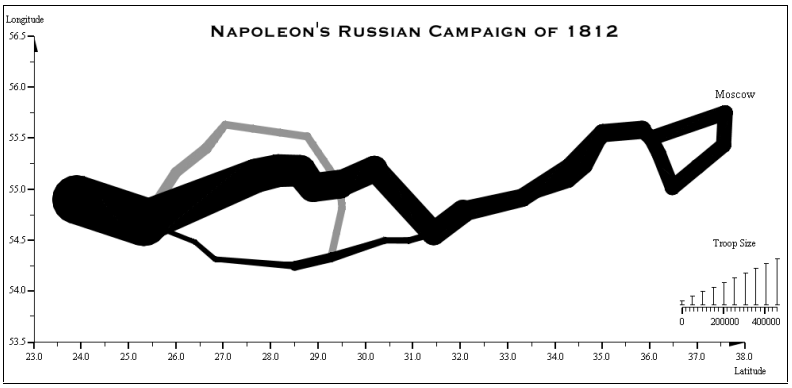


Figure 17: Napoleon's Campaign data using size ranges to have point size, and therefore line size, represent troop size.

Two size range primitives must be added since each of these range primitive handles only one data set. The graph points are bound with the range that is associated with the appropriate data set. As with the axes' scales, the maximum and minimum values are set to be identical and only one of the graphical representations of the ranges is left visible in the final graphic.

5.1.3. Adding Color Ranges to Create a Variably Colored Line Graph

Further improvements could still be made to this graph. The retreat phase of the campaign becomes relatively hidden as the attacking and retreating line of the graphic overlap. Moreover, no information is given as to outdoor temperature, which has a significant impact on casualties in the retreat. To address this color range primitives associated with temperature can be added to allow the temperature at each division's location to be represented by the graph point's color. The result of this change is seen in Figure 18. As with the size ranges, two color ranges are added to handle each division's data set. Since their scales are set to be identical, only one graphical representation of the range is displayed. A new title with a different font is also added to the graph.

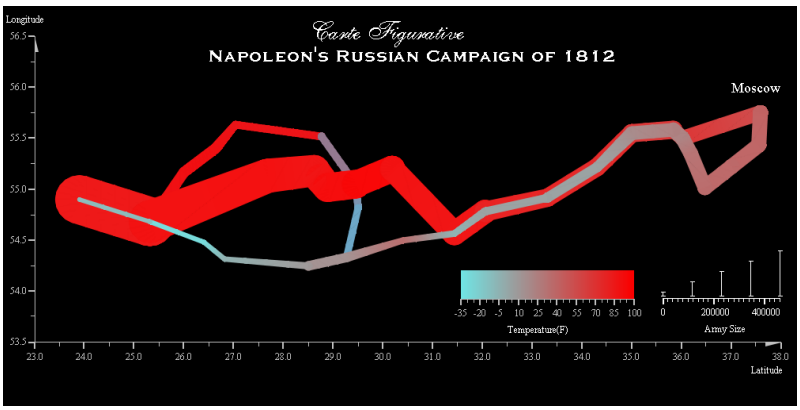


Figure 18: Napoleon data showing positional data, army size (by point size), and outdoor temperature (by point color).

5.1.4. Adding Text Range Primitives to Create Variable Labels

This graph provides a similar amount of information to the Minard original, but it does not give a sense of the physical cities that the armies are moving through. By adding a text range associated with the town names of the divisions' positions, graph labels can be bound with these ranges to display the town names in the graph. To have the labels positioned appropriately they are also bound with their appropriate divisions' horizontal and vertical positions. Figure 19 shows the results of this addition.

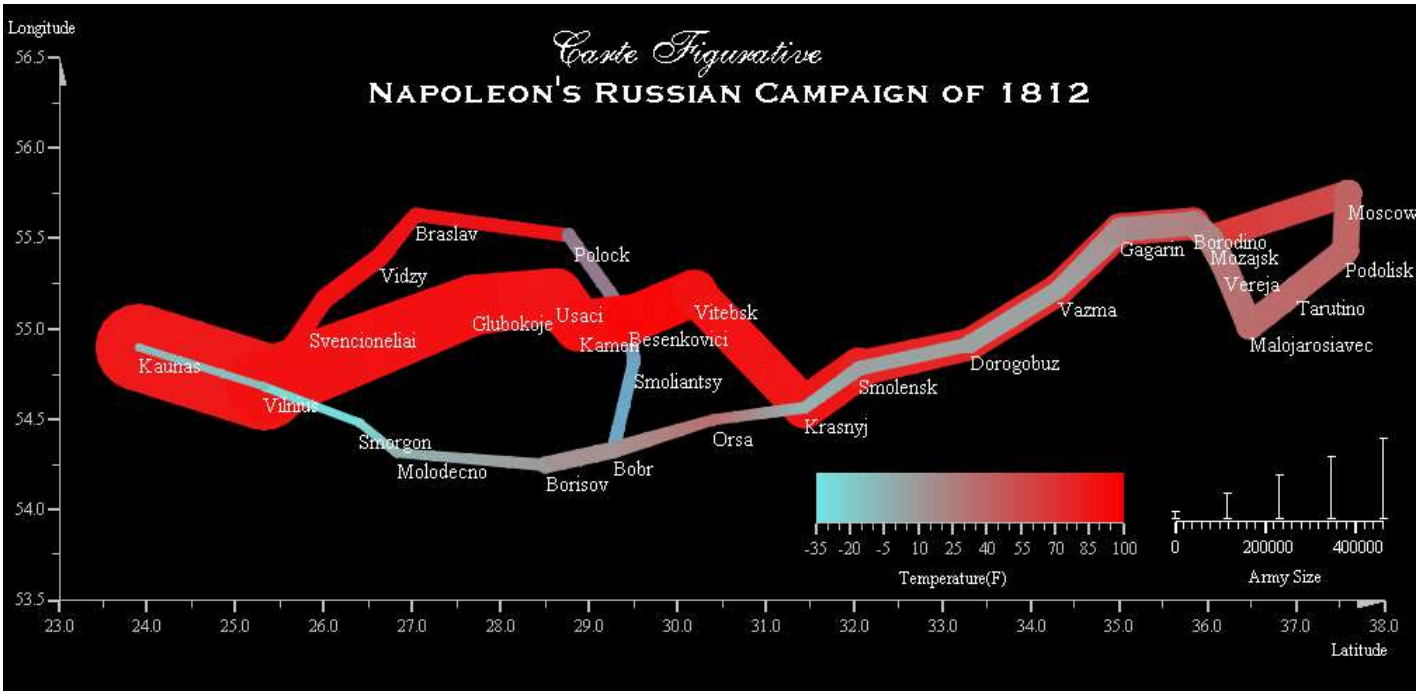


Figure 19: Napoleon Campaign Data with town names added.

5.1.5. Using Animation and Overlaid Primitives

As a final improvement to this graph we could use animation to show the divisions' progress over time.[10] Another horizontal axis will be added to the base of the graph and bound with a graph point to represent the particular day of the graph frame. Also a hidden overlaid graph axis with a visible marker as shown in Figure 14 will be added to make the outdoor temperature in a particular frame clearer. The results of a series of frames[11] from this animation are seen in Figure 20.

[10] To create this animation, linear interpolation was used between unknown data values.

[11] A series of graph frames are used to convey animation, but the AVI's for these animations are available at http://ddias.student.harvard.edu/angraf.html.

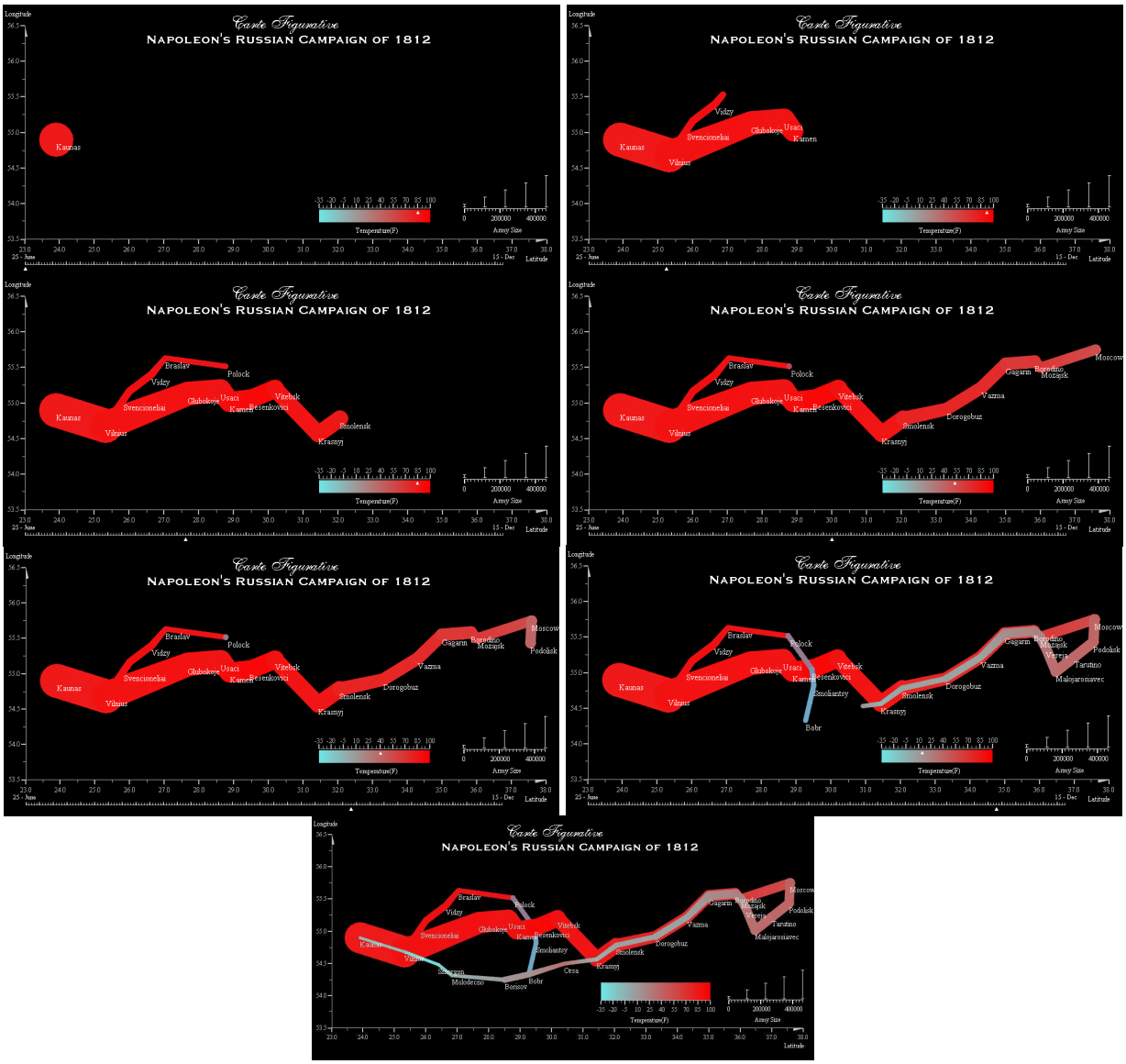


Figure 20: A series of frames from the final animated graph of Napoleon's Campaign.

5.2. AnGraf and Other Data Sets

The Minard example shows most of the functionality of AnGraf's primitives. It is important to note, however, despite the flexibility of its primitives they are not sufficient to create any two-dimensional graph. For example, AnGraf's current structure does not support creating network diagrams or pie charts. Yet, as the following examples further illustrate, the versatility that AnGraf's current graph primitives provide shows that its paradigm of building graphs is a powerful method in creating customized static and animated data graphics.

5.2.1. Seismogram Data – Approaches to Static Graphs

When seismologists investigate earthquakes, they often fit observed seismogram data to a synthetic curve. Figure 21 is an example of one such graph of the observed and

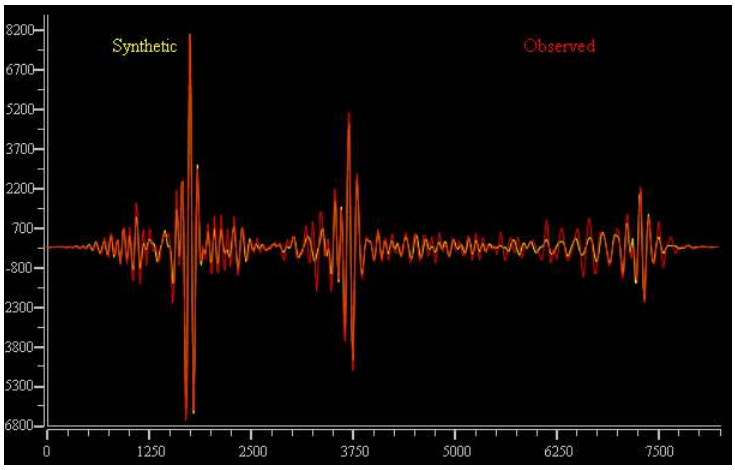


Figure 21: Graph of synthetic and observed seismological data over time

synthetic curve of seismic activity over time. This graph shows that the synthetic curve is a fairly good fit to the observed curve, but it does not provide insight into exactly where the model is making its errors. By adding another two-dimensional axis to the layout that is associated with error, a single graphic can convey the two seismograms and curve fit data. The resulting graph is shown in Figure 22. The graph shows that there are definite peaks of error between the two seismograms, but it is hard to tell exactly where these peaks are on the synthetic seismogram. One guess might be that they correspond to the areas of high seismic activity since error in this case is simply calculated as the absolute value of the difference between the observed and synthetic data values.

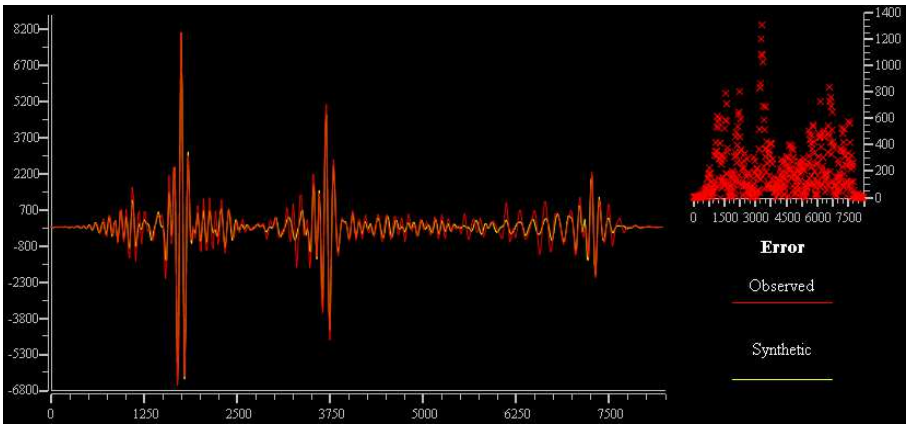


Figure 22: Seismograms with an adjacent error scatter plot

To clarify this, however, the error values need to be overlaid onto the original seismogram. This can be done by removing the error graphs previous horizontal axis and binding its vertical axis and the graph point to the right side of the horizontal axis used to represent the seismogram's time value. The result of this change is seen in Figure 23. It

shows that the error peaks are not from the areas of high seismic activity but from the areas before and after high seismic activity.

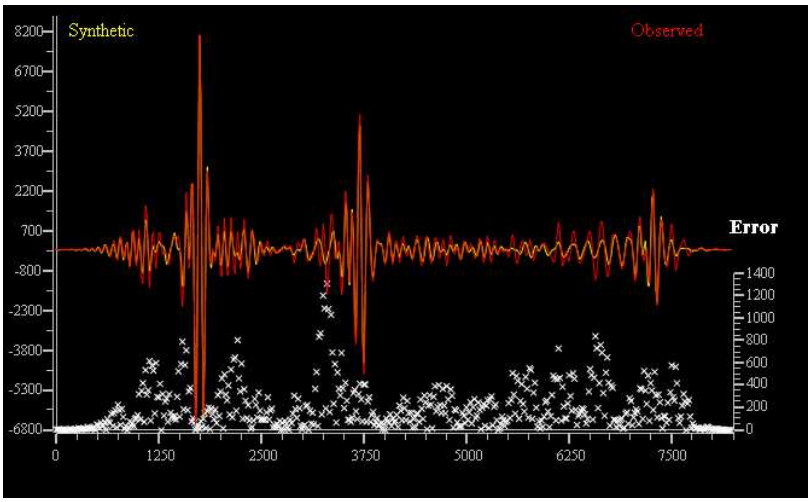


Figure 23: Seismograms with overlaid error scatter plot.

A cleaner way to represent the same data would be to show the error term immediately on the synthetic graph line. This can be accomplished by allowing the graph point's color, and consequently the line's color, to change with error. Figure 24 implements the desired effect by adding a color range primitive that is associated with error, and binding the graph point that represents the synthetic data to it. The synthetic seismogram line is made slightly thicker by changing the width of its underlying graph point, and the observed seismogram data is removed from the graph for clarity. In this graph, the color-coding makes it immediately apparent which particular curves of the seismogram have high error values.

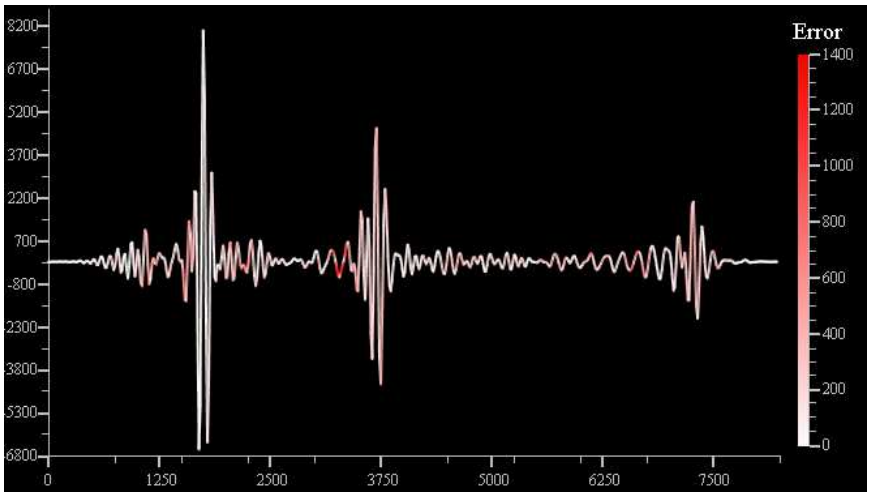


Figure 24: Synthetic seismogram colored by error values

5.2.2. Treasury Data – Using Animation to Demonstrate Trends

U.S. Treasury Securities are sold over different maturing dates at different yields. The general principle in the marketplace is that longer bonds, or those with a greater maturity date, should pay more than shorter-term bonds because an investor should be paid for their liquidity constraint. The concept of being paid for lack of liquidity is also seen in local banks where higher interest rates are offered for CD's, which have withdrawal constraints, than for regular savings accounts, which have no such constraints. Investors look at the differences between longer bonds and shorter bonds to determine how the market is rewarding liquidity. One such representation of this relationship might be to graph the thirty-year yield vs. the three-month yield along with the ten-year yield vs. the one-month yield. The result of this graph is shown in Figure 25. It is formed by a single two-dimensional axis with two variables associated with both axes, and scales are physically placed and colored for clarity.

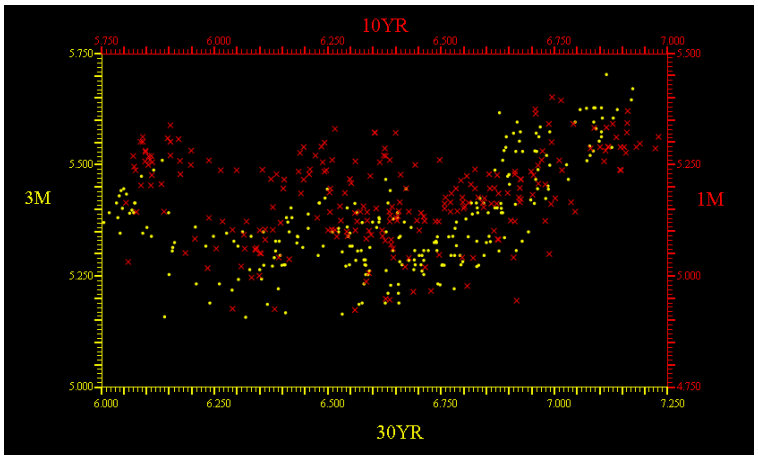


Figure 25: Thirty-Year vs. Three-Month and Ten-Year vs. One-Month U.S. Treasury Securities' yield scatter plot

The association is not as linear as one might initially suspect. A shift in the ten-year yield and thirty-year yield seems to have a limited influence on the three-month yield and one month yield respectively. Further investigation with animation, however, can provide some more insight into this. Look at Figure 26 which shows three consecutive

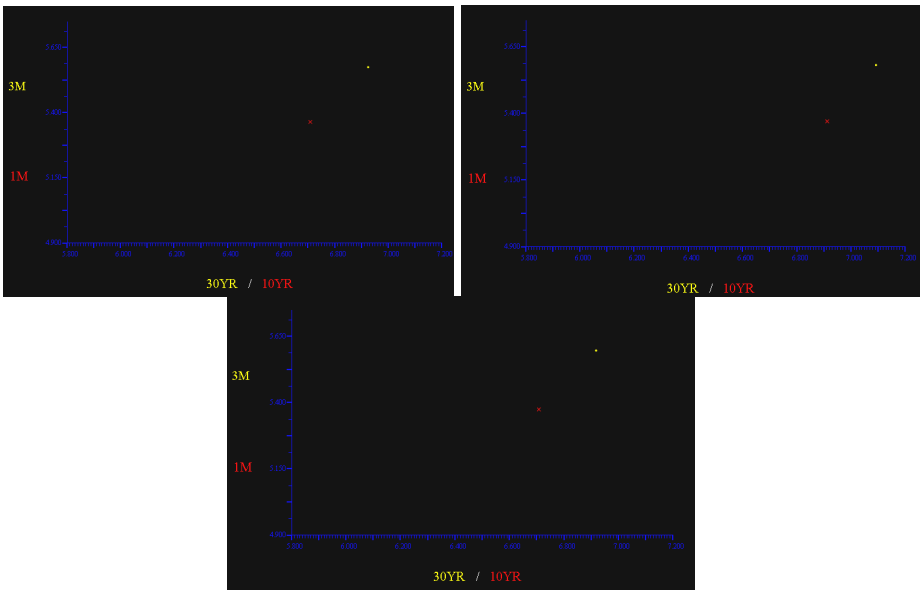


Figure 26: Sequence of frames from the animation of Thirty Year vs. Three Month and Ten Year vs. One Month U.S. Treasury Securities' yields. The graph is being animated by the Three-Month's yield

frames from an animation over the three-month's yield. By animating over the three month treasury's yield, it becomes more evident that even though the ten year and one month securities are not clearly related to each other and the thirty year and three month securities are not clearly related to each other, the relation between these longer and shorter securities move somewhat in tandem. Figure 26 demonstrates this as the two graph points move together over the animation. This means that there is a comparable spread, or yield difference, between these longer and shorter securities, and this spread is the relative reward for liquidity constraints.

The concept of rewarding liquidity constraints can be further demonstrated by graphing the yield curve. The yield curve is formed by a series of yield values for securities with increasing maturities. Figure 27 shows the yield curve for a single day over what are known as on-the-run, or the most recently auctioned, US Treasuries.

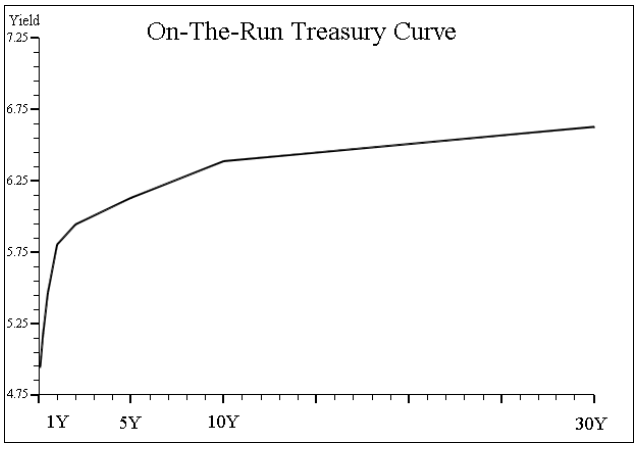


Figure 27: Single day yield curve over On-The-Run US Treasuries

This display makes it readily apparent that longer bonds are rewarded with higher yields. It is actually a single frame in a series of graphs that contain yield curve information for different days. The lines interconnecting the points on the graph in this case are from graph line objects. These are different from the lines used before, at least in the editing process, because previous lines were drawn between the same graph point, like the yellow synthetic seismogram against time graph in Figure 21 on page 20. The graph point itself can implement this option. In the above case, however, a graph line is being drawn between two separate graph points, like the ten year and thirty year security. An independent graph primitive handles this function.

The yield curve has other applications besides simply providing a graphical representation of rewarding liquidity constraints. By viewing daily yield curves from December of 1996 to December of 1997, the effect the Asia crisis had on the bond market becomes quite obvious. Shots from frames of the animation are shown in Figure 28. The graphs show that yields were dramatically reduced over this period as investors fled to the US market for security. Furthermore, dramatic selling of shorter securities by Japanese banks and purchasing of longer securities by other investors resulted in a significantly "flattened" the yield curve.

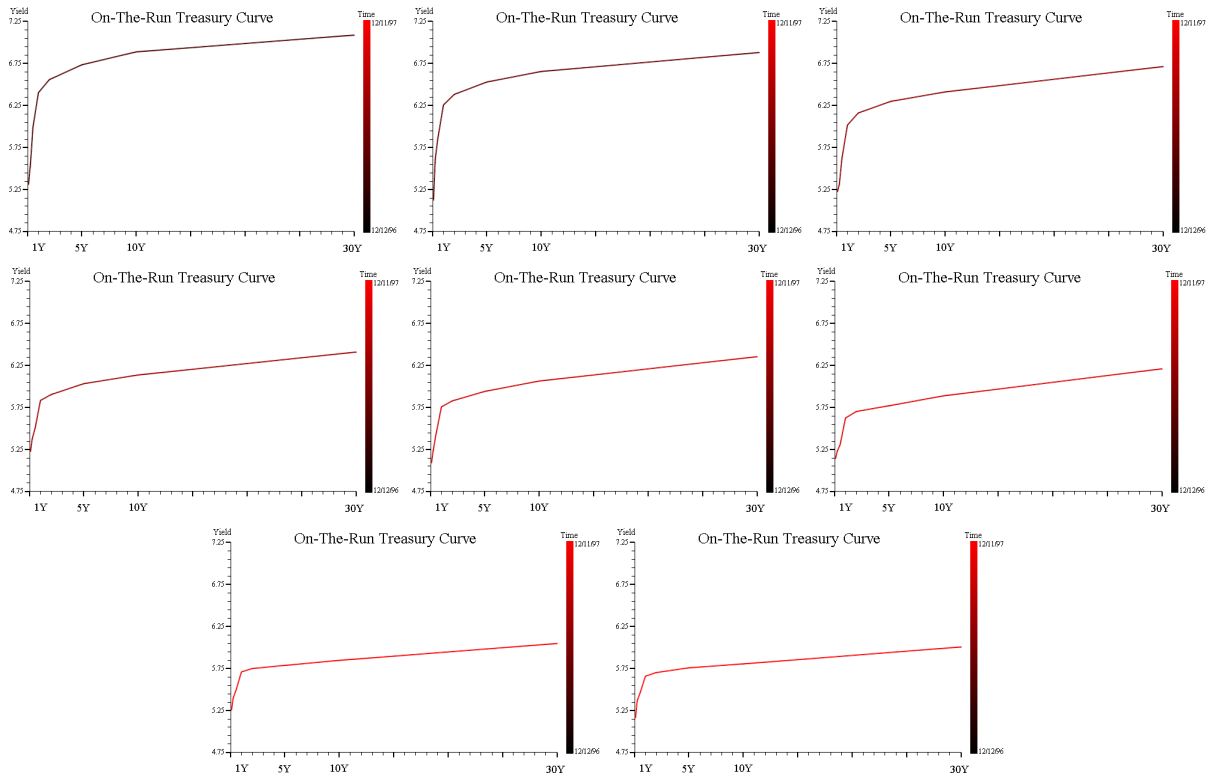


Figure 28: Series of shots from the animation of the On The Run yield curve from 12/12/96 to 12/11/97

5.2.3. Galapagos Island Data – Different Views of Identical Data

In 1973 Johnson and Raven[12] collected data from the Galapagos Islands to see what factors influence the development of life. Their data is organized by island and includes information on the number of native species, the number of total species, the size of the island, the distance to the nearest island, and the area of the nearest island. It would seem to make sense that islands with close neighbors should have a greater difference between total species and native species because it is easier for non-native species to migrate from one island to another. It also might be logical to assume that the island size might have an impact on whether a certain species would migrate. Maybe smaller islands are more crowded and are more likely to encourage emigration. Even if these theories are not correct, a data graph can provide some insight into their validity. Using AnGraf's animated framework a graphic can be created that allows a particular island's data to be displayed in each frame. Figure 29 shows one such graphical frame.

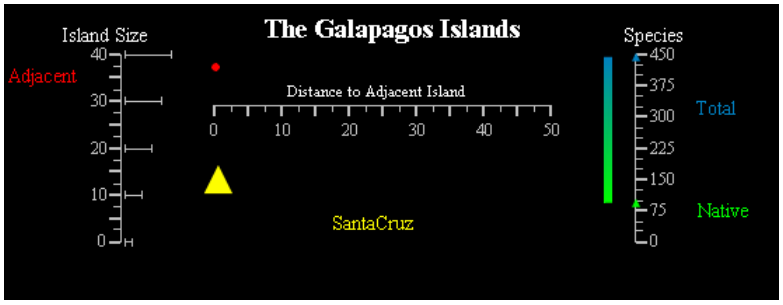


Figure 29: A single frame of Galapagos Island Data

Many different primitives are being used to form this graph. The island's name is given in the yellow graph label at the bottom, and it is updated in each frame because of its bond with a variable text source that is associated with island name data. The yellow triangle above the island's name shows two things. First, its horizontal position marks its distance from its nearest island. Second, its size represents the island's size. The size value can be compared against the size of the red graph point above the distance axis to get some sense of how islands areas differ. Finally, the blue to green bar represents the total difference between native and total species which are marked by blue and green points.

This graph frame provides a great deal of information about an island, and by animating over distance to the adjacent island, one can get an impression of how distance affects the percentage of native species in the total number of species on an island. The series of graphical frames show that there is a great deal of variance in the data, but there is a general increase in percentage of native species in the total number of species on an island as distance increases. More significantly, however, the animation reveals an anomaly. Figure 30 shows the last frames of the animation. All of the distant islands have small difference between native and total species, which is represented by the blue

[12] Andrews, D.F., Herzberg, A.M., Data, (New York, NY: Springer-Verlag, 1985), 291. Data obtained electronically from StatLib at http://lib.stat.cmu.edu/datasets/.

to green bar in their frames, except for San Cristobal which also has the largest island size.

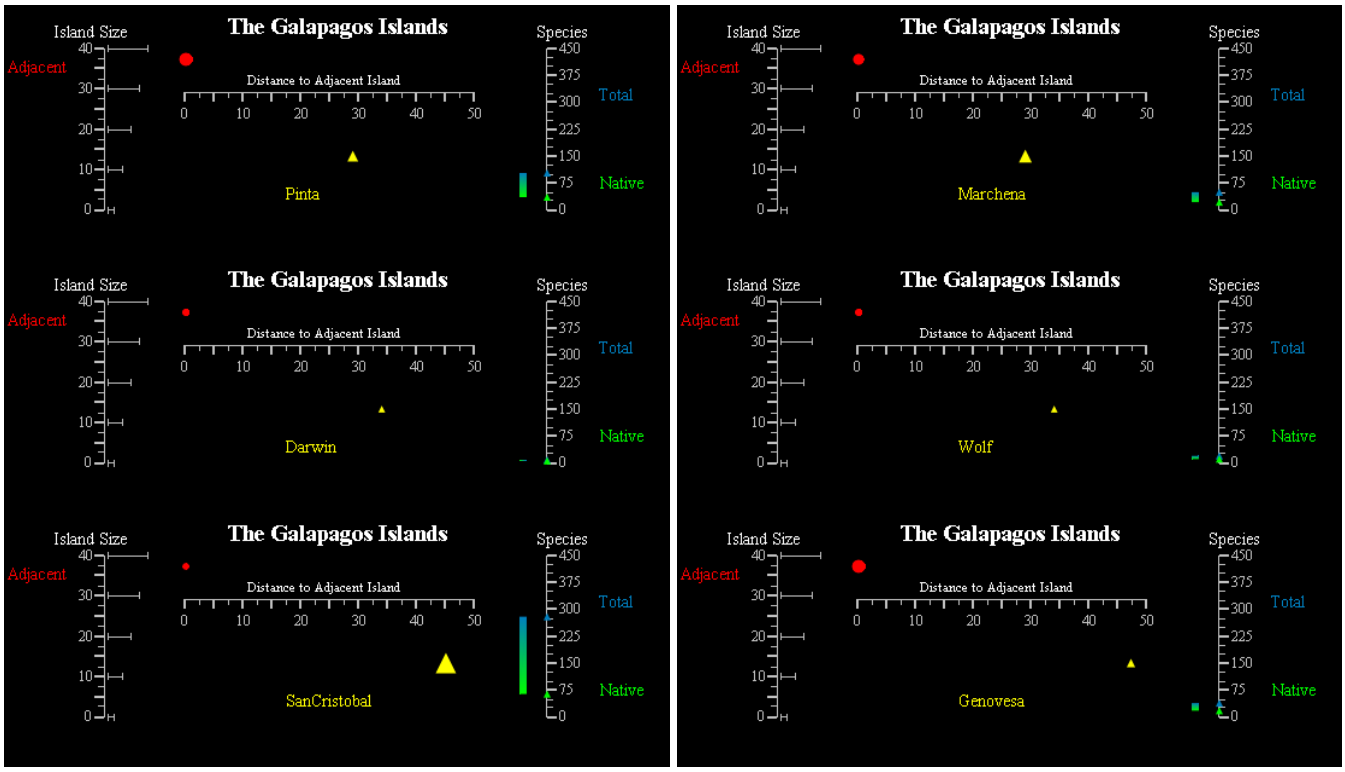


Figure 30: Last six frames of Galapagos Island data shown in increasing distance to adjacent island.

Seeing this data from San Cristobal hints that island size may directly affect the distribution of native and non-native species. A matrix graph can be generated to see the relationships between native species, total species all in a single graphic. A matrix graph merely aligns different two-dimensional axes so the relationship between different variables can be seen on the same page. Positioning and sizing of various axes identically to other axes is done using the tools in AnGraf's editing window.

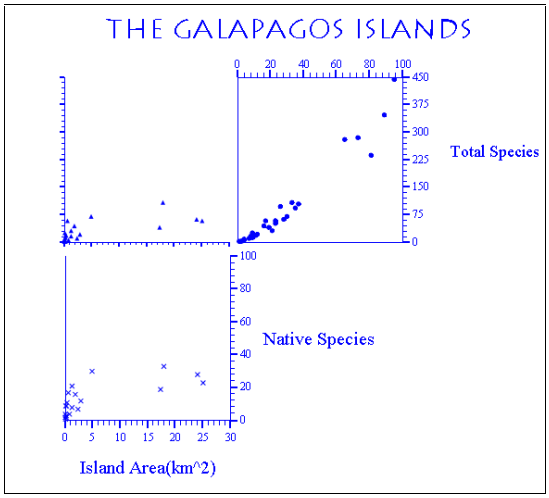


Figure 31: A matrix graph of Galapagos data variables.

5.2.4. Galaxy Data – A Different Use of Overlaid Graph Primitives

AnGraf allows for primitives to be placed in any location desired, and this can create interesting effects. As seen in Figure 14 and the Minard example, overlaying graph axis on a color range can clarify a graph. It is particularly useful in animated residual graphs that rely heavily on color because the graph point provides accurate interpretations of the current frame's numerical value while the changing color remains in the residual graph to show trends. This type overlaid primitive was again used in representing a data set of star information[13] which marks the star's position in the east/west direction, the north/south direction, and its relative speed. An animation of the position of stars as speed increases was created, and Figure 32 shows frames from this animation.

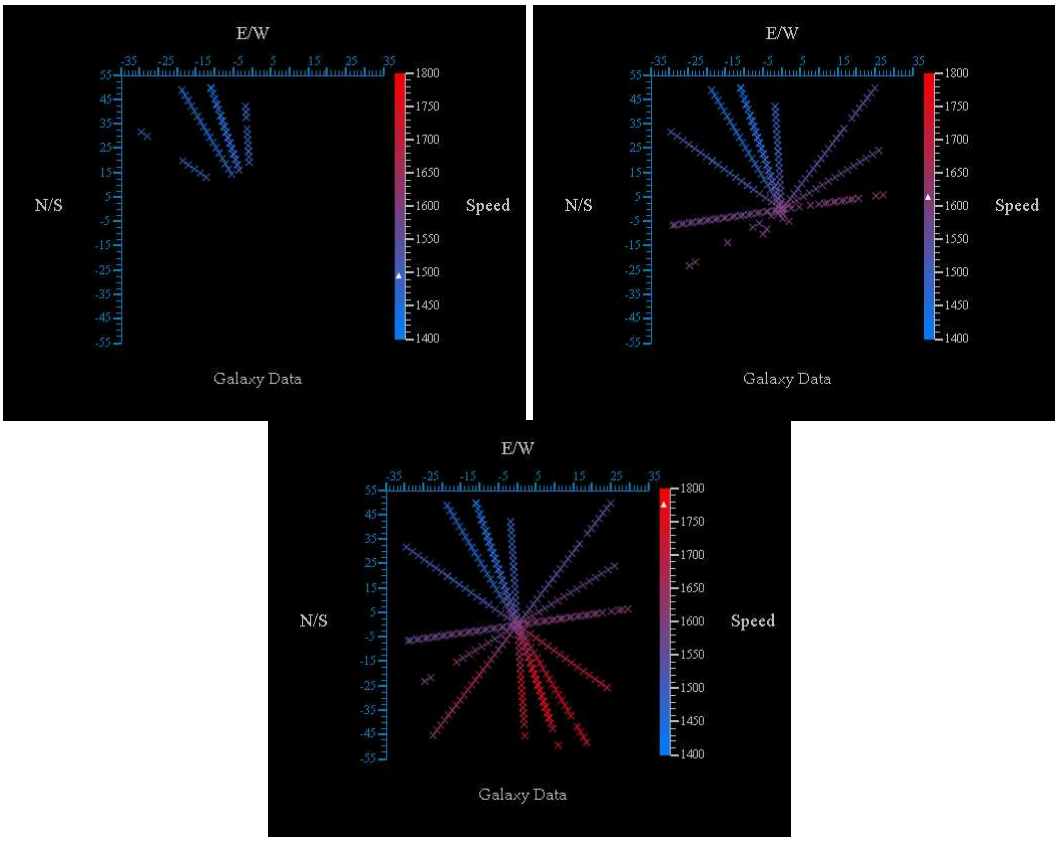


Figure 32: Series of graph frames from an animation of star's position as speed increases.

[13] Cleveland, W., Visualizing Data, (Murray Hill, NJ: AT&T Bell Laboratories, 1993). Data obtained electronically from StatLib at http://lib.stat.cmu.edu/datasets/.

5.2.5. Outbreak Data – Using Color Ranges to Have Primitives Variably Appear

Since AnGraf's structure places few limitations on primitives, they can be used in many creative ways. For example, by using a binary representation for some state in a graph, this data set could be associated with a color scale where the non-existent state is the canvas color and the active state is a different color. This would allow the animated frame to have items dynamically appear with this data.

This technique could be useful, in the representation of the spread of a deadly disease. A residual animation of locations of deaths would demonstrate the progress of the disease, and by using the method above, the animation could show when and where a quarantine had been established during the epidemic. A series of resulting frames over a hypothetical data set using this technique is shown in Figure 33. The frames show not only that the quarantine was established but that it failed.

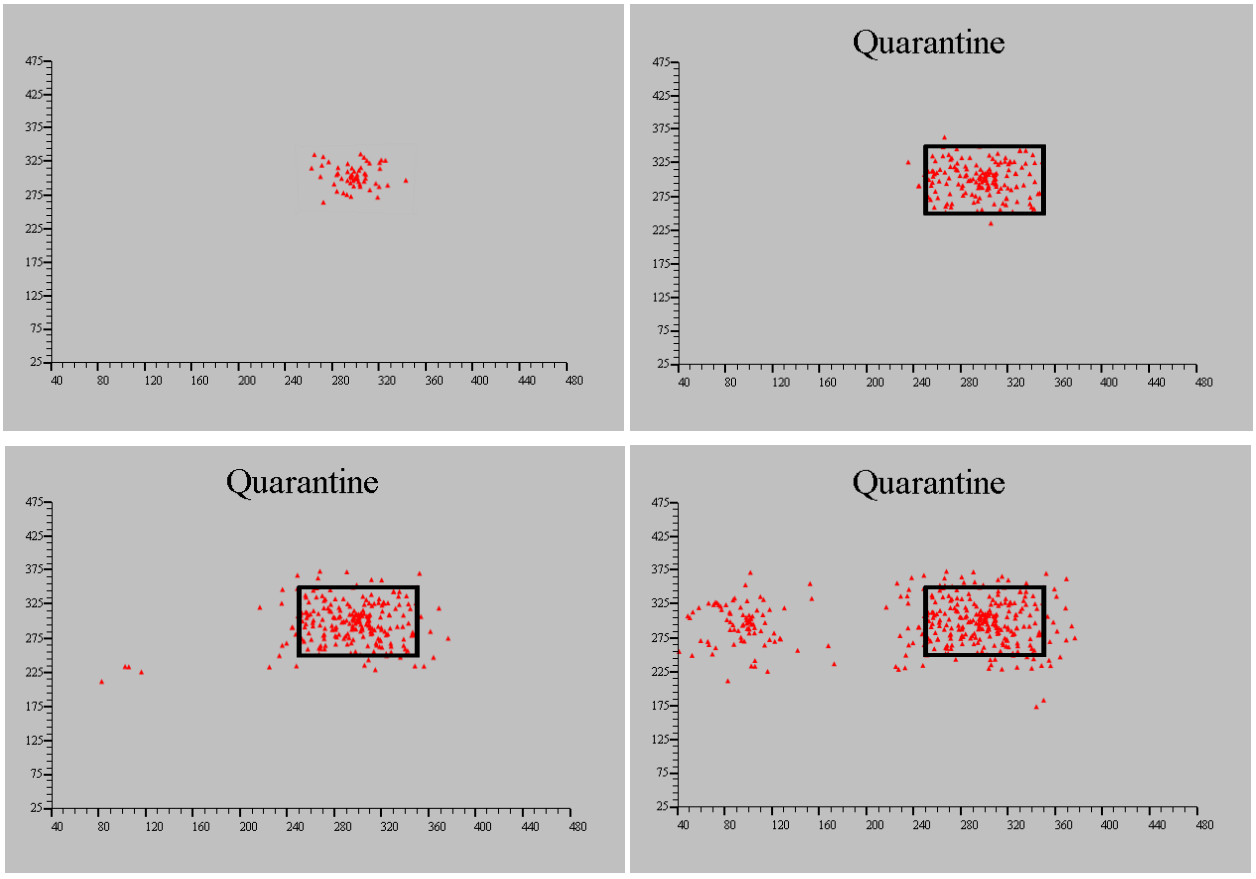


Figure 33: Series of animated frames from a hypothetical failing of a quarantine of a deadly disease. Triangles mark location of deaths.

6. Conclusions

While there are other products that overcome the limitation of standard graphs in graphing packages like Chart Wizard, they do not present complete solutions. Programming tools such as the Visualization Toolkit can be used to generate customized animated data graphs, but the coding can take a significant amount of time and requires more skill than the average user would have. Even though current research projects, such as Gold and SAGE, overcome this dilemma by providing relatively simple mechanisms for creating customized graphs, they fail to provide the user with the flexibility to control final graphical output. Furthermore, these projects have yet to address the issue of animation.

AnGraf overcomes many of these limitations by implementing a different approach to data graphics. Rather than presenting standard graph designs for a user to select, AnGraf provides the primitives to build graph designs according to a user's needs. By adding primitives, associating data with ranges, and binding markers with these ranges, users can layout the exact graph designs they wish to see. These designs then are translated into graphs with accurate representation of their data.

Whether it be the static matrix graph of Galapagos Island data or the animated graph of Napoleon's campaign, previous examples have demonstrated the variety of data graphs a simple subset of primitives can produce. AnGraf's model for creating data graphics, like the spreadsheet's approach to calculations, is very flexible and allows the user to build highly customizable data graphs.

7. Acknowledgements

I walked into Professor Shieber's office in the Fall of 1998 to tell him that I wanted to write a thesis on the visualization of risk in bond trading. Needless to say, after talking with him, I decided to look into a more general problem in data graphics, and AnGraf was born. As I finish this thesis I would like to thank him for challenging me to do more and guiding me through a very rewarding experience.

I also would like to thank my roommates for listening to my late night philosophizing on data graphics, watching my program demonstrations, and ultimately laughing with me as I wrote the code for this program. Their encouragement was instrumental.

Likewise, I would like to thank my family. Kimberley, Mom, and Dad, you were always supportive regardless of whether I said that things were going great or that I wanted to drop my thesis. I want to thank you for always being there.

Finally, I would like to thank Kimberley Dias, Henry Nuzum, and Andrea Stover for taking time out of their spring break to help me proofread and edit this document.

This material is based in part upon work supported by the National Science Foundation under Grant No. IRI-9618848 to Stuart M. Shieber.

8. Index of Figures

9. Bibliography

Andrews, D.F., Herzberg, A.M.. Data. New York, NY: Springer-Verlag, 1985.

Cleveland, W.. The Elements of Graphing Data. Monterey, CA: Wadsworth Advanced Books on Software, 1985.

Cleveland, W.. Visualizing Data. Murray Hill, NJ: AT&T Bell Laboratories, 1993.

Cleveland, W., McGill, M.. Dynamic Graphics for Statistics. Monterey, CA: Wadsworth, 1988.

Maybury, M.. Intelligent Multimedia Interfaces. Cambridge, MA: The MIT Press, 1993.

Myers, Brad A., Goldstein, Jade, and Goldberg, Matthew A.. "Creating Charts by Demonstration," *Proceedings CHI'94: Human Factors in Computing Systems*. Boston, MA, Apr. 24-28, 1994. pp. 106-111.

Roth, S.F., Kolojejchick, J., Mattis, J., Chuah, M., SageTools: An Intelligent Environment for Sketching, Browsing, and Customizing Data Graphics, *Conference Companion of the Conference on Human Factors in Computing Systems (SIGCHI '95)*, Denver, CO, May 1995, pp. 409-410.

Roth, S.F., Kolojejchick, J., Mattis, J., and Goldstein, J. Interactive Graphic Design Using Automatic Presentation Knowledge. *Proceedings of the Conference on Human Factors in Computing Systems (SIGCHI '94)*, Boston, MA, April 1994, pp. 112-117.

Roth, S.F. and Mattis, J., Data Characterization for Intelligent Graphics Presentation, *Proceedings of the Conference on Human Factors in Computing Systems (SIGCHI '90)*, Seattle, WA, April 1990, pp. 193-200

Schroeder, W., Martin, K., Lorensen, B. The Visualization Toolkit. Upper Saddle River, NJ: Prentice Hall, 1998.

Tufte, E. The Visual Display of Quantitative Information. Cheshire, Connecticut: Graphics Press, 1983.

Tukey, John. Exploratory Data Analysis. Reading, MA: Addison-Wesley, 1977.

World Wide Web Sites

"Brad Myers Home Page"
http://www.cs.cmu.edu/afs/cs.cmu.edu/user/bam/www/myers-home.html

"The Visualization and Intelligent User Interfaces Group" – (SAGE)
http://www.cs.cmu.edu/Groups/sage/sage.html

… and countless other programming web sites