An Event-Based Approach to Visualization

Christian Tominski Heidrun Schumann
Institute for Computer Science
University of Rostock, Germany
{ct,schumann}@informatik.uni-rostock.de

Abstract

Visualization of large data sets is a demanding task, especially if different users are interested in different aspects of the same data set. Today's visualization techniques often do not distinguish different aspects and thus visualize all aspects of the data, which leads to overcrowded and cluttered representations. Moreover, users are provided with information irrelevant to them.

In this paper we describe an event-based approach for visualizing large data sets. The basic idea is to let users describe interesting aspects of the data by means of events and then adapt the visual representation of the data on occurrence of events. We present a formal description of events, assuming that the data are given in relational form. Furthermore, a model for event-based visualization is developed. A brief overview of an early realization of the model along with first visual results is part of this paper.

1. Introduction

Nowadays modern database technology enables scientific institutions, governmental authorities, and companies to collect large amounts of data. These data are for instance experimental datasets, health statistics, or customer surveys. Visualization has been proven to be an efficient method for analyzing such data. Therefore, (abstract) data are mapped to visual representations in order to benefit form the capabilities of the human visual system.

North, Conklin, and Saini state in [1] that modern relational database technologies allow efficient and flexible data management, but today's visualization techniques do not reach this level of flexibility. Developed by a single programmer, visualization techniques are often specific to only one certain problem. Moreover, in the case of large datasets it is common that different users explore the data with respect to different aspects. This means that a visualization technique must be able to represent any of these different aspects. However, most visualizations do not distinguish between different aspects and thus, users are obliged to create a dataset consisting of only relevant data aspects prior to visualization. Otherwise, users are provided with a lot of information they are not interested in at all.

With respect to modern visualization methods for large datasets and different users, this leads to the following two demands:

- Increase of flexibility, i.e. allow users to specify their interests and then adapt the visualization automatically.
- Increases of efficiency, i.e. emphasizing portions of the data users are interested in and reduce information irrelevant to them.

In this paper we present a model for event-based information visualization, which can help to reach the described goals. The basic idea of the event-based visualization is to allow users to specify pattern they are interested in. Once a pattern is detected within the data (either a static or a dynamic dataset) an event is triggered. According to the event and its properties the visualization is adapted automatically.

The following sections will introduce our ideas for event-based visualization. Related work and a state-of-the-art overview are given in Section 2. The description of events (Section 3) is the basis for the development of our model in Section 4. Furthermore, in Section 5 we show first results of a realization of our model in the axes-based visualization framework VisAxes.

2. Related work

Events and event-based approaches are widely used in a variety of information technology related problems. Events are a prerequisite for reactive

behavior and thus a requirement for automatically adapting software systems.

With respect to visualization, only events limited to specific problems have been used in the past years. Matkowic et.al. [2] describe how simple events can be used in order to adapt virtual instruments in process visualization. In fact, the visual representation and thus the level of detail of virtual instruments is changed if certain thresholds are reached. In this way the visualization supports an observer in finding critical points in a real time process.

From the area of flow visualization the approach from Reinders et al. [3] is known to make use of events. Based on feature extraction relevant portions (i.e. features) of time-dependent flow data are visualized. Additionally, events – described as *any development in the evolution of a feature that is significant* ([4] p.107) – are detected and presented in an event graph. Birth and death, entry and exit, as well as split and merge of features are considered significant and thus relevant for the visualization of features in the event graph. By representing relevant events in combination with icon-based flow visualization it is easier for users to comprehend the characteristics of the flow data.

The visual representation of algorithms [5] as well as distributed program debugging and visualization [6], [7] can also be enhanced by means of event-based techniques. The transition from one state of a program or algorithm to another is considered an event. There is an ongoing discussion [5] whether state-based or event-based approaches are more expressive for the purpose of specifying program visualization.

Intrusion and misuse detection [8] in IT-networks is an evolving area of visualization. In [9] an event-based architecture for such visualizations is described. Based on events read from server log files in combination with events generated by the user interface, the visualization is refreshed automatically at runtime. By such means, events like port scans, file transfers, or arbitrary net connections are represented visually. This helps administrators to find weak spots in a network, heavy load times, or misuse of resources.

These examples for visualizations show that event-based approaches can be used to ease the analysis of arbitrary data. However, all described systems developed their own proprietary events and are limited to a specific visualization problem.

Our aim is to define a more general event-based approach in order to allow an effective and flexible analysis of large static or dynamic data sets. We assume that these data sets are given in relational form.

3. Events

The essential requirement for a general event-based visualization model is the description of the term *event*. In our context we consider events with respect to relational data sets, attributes, and data records (i.e. tuples). By doing so, we want to close the gap in flexibility (cp. [1]) between relational data management and data representation.

3.1. Event description

Events in our words are special portions of a data set complying with certain conditions and constraints regarding attributes and records. Special about these portions is that a user is interested in them. Once detected, events may have associated a variety of parameters, like attributes involved, data records involved, relations to other events, importance, and so forth. Concrete values for these parameters are determined either by means of the event detection (e.g. for involved records) or by users themselves (e.g. for importance).

An example of an elementary event is an exceeding of a certain threshold of an attribute (e.g. cases of influenza > 300). Since elementary events are eligible for simple visualization tasks only, it is necessary to allow for composite events created from elementary events and certain operators (for instance logical operators like *and*, *or*, *not*). By doing so, certain combinations of values in a data record (e.g. sex = male *and* profession = manager *and* diagnosis = S.A.R.S.) can be detected.

3.2. Event formalism

Requirements: Suppose, $\mathcal{A} = \{A_1, \ldots, A_n\}$ and $\mathcal{R} = \{R_1, \ldots, R_m\}$ are finite and non-empty sets of *attributes* and value *ranges*, respectively, and a function $range : \mathcal{A} \rightarrow \mathcal{R}$ assigning each attribute a range. We call $\mathcal{U} = \bigcup_{i=1}^{m} R_i$ the data *universe*. Then, a *data set* can be modeled as a finite set of *tupels* $T = \{t \mid t : \mathcal{A} \rightarrow \mathcal{U} \text{ and } t(A_i) \in range(A_i)\}$ where a tuple is a function that associates each attribute with a value from the attribute's value range.

Event definition: We now want to give a general definition of events. An event can be described by means of first order predicate logic (PL-I) formulas. The tuple relational calculus known form relational

database theory [10] defines formulas which can be adapted to our needs.

Symbols that might appear in a formula are:

- *tuple variables* (elements of a countable infinite set V disjoint from the universe $\mathcal{U}$),
- *aggregate functions* over attributes $\tilde{h}: \mathcal{A} \to \mathcal{U}$,
- k-ary *functions* of the form $g^k : \mathcal{U} \times_1 \cdots \times_{k-1} \mathcal{U} \to \mathcal{U}$ (if $k = 0$ then g is a *constant*), and
- k-ary logical *predicates* $P^k \subseteq \mathcal{U} \times_1 \cdots \times_{k-1} \mathcal{U}$.

These symbols except the latter one are used to define *atoms*, which are then used to construct formulas. Atoms are defined as follows:

1. If $x \in V$ is a tuple variable and $A_i \in \mathcal{A}$ an attribute, then $x.A_i$ is an atom denoting the value of attribute A_i in the tuple represented by x.
2. If $\tilde{h}$ is an aggregate function and $A_i \in \mathcal{A}$ an attribute, then $\tilde{h}(A_i)$ is an atom denoting the value of the function for A_i.
3. If g^k is a k-ary function and $a_1, \ldots, a_k$ are atoms, then $g^k(a_1, \ldots, a_k)$ is an atom denoting the value of the function.

We now can define an *elementary formula* using the defined atoms and predicates as follows. If P^k is an k-ary logical predicate and $a_1, \ldots, a_k$ are atoms (as defined in 1.-3.), then $f^{elem} = P(a_1, \ldots, a_k)$ is an elementary formula. In order to evaluate a formula we require a *substitution* of tuple variables with tuples of the data set given by a complete function $s: V \to T$. We denote as $s(a_i)$ the atom after applying s to each tuple variable in a_i.

Now we can state that a formula f^{elem} evaluates to *true* iff there exists a substitution s so that $(s(a_1), \ldots, s(a_k)) \in P^k$.

Elementary formulas (with predicates like $<, =, \ldots$), common logical *connectors* $(\neg, \wedge, \vee, \to)$ and *quantifiers* $(\exists, \forall)$ can now be used to construct *general event formulas* f as usual.

Following [10], we define the general form of an event as $\{\dot{x} \mid f\}$, where f is a formula (the *event formula*) and $\dot{x}$ (the *target*) is a tuple variable in f. We call $e_t = \{\dot{x} \mid f \mid s\}$ an *instance of an event* $\{\dot{x} \mid f\}$ for a concrete tuple $t \in T$ iff f is true under substitution s and $s(\dot{x}) = t$.

Examples: These definitions provide us with a powerful formalism for describing a variety of events with respect to the relational data model. Some formal examples of event types shall be given. The exceeding of a threshold can be described as $\{\dot{x} \mid \dot{x}.\text{influenza} \geq 300\}$. A certain range of values is detected with $\{\dot{x} \mid \dot{x}.\text{temp} \geq -1 \wedge \dot{x}.\text{temp} \leq 7\}$. An "unusual network load per user" event can be defined as $\{\dot{x} \mid (\dot{x}.\text{load} / \dot{x}.\text{users}) > 1.024\}$. If a user is interested in maximum values of the pollutant carbon-monoxide (CO) a corresponding event can be defined either by using more than one tuple variable (i.e. $\dot{x}, y$) in $\{\dot{x} \mid \forall y(y.\text{co} \leq \dot{x}.\text{co})\}$ or by means of an aggregate function $\{\dot{x} \mid \dot{x}.\text{co} = max(\text{co})\}$. An increase of the pollutant with respect to the "previous" measurement can be described by using two tuple variables in $\{\dot{x} \mid \dot{x}.\text{co} > y.\text{co} \wedge \dot{x}.\text{date} = (y.\text{date} + 1)\}$.

4. Processing events for visualization

Having defined our interpretation of events, we now will describe how these events can be incorporated to visualization.

4.1. Event specification

The first question arising is how events can be specified by users. For that we identified the following requirements:

- Intuitive event specification is required to allow different user groups (e.g. decision makers or data analysts) working with events efficiently.
- Events must be storable with respect to the dataset addressed and events must be editable to allow an easy adaptation to changes in the working environment.
- The specified events should be tested for satisfiability.

In order to reach these goals we intend to use *event templates* in combination with a visual interface. The templates are described by means of XML grammar. They provide parameters, which are set by users. Instances of parameterized templates can then be used for event detection. Figure 1 gives an example of an event template and its parameterized visual representation. In order to create a visual representation of an event, symbols occurring in event formulas are mapped to visual primitives, which are linked regarding the symbol's position in the formula. This allows for event specification by means of dragging and linking the primitives in a visual editor.

```
<event name="threshold">
<param name="attribute" type="string"/>
<param name="relation" type="choice" data="<,=,>"/>
<param name="value" type="string"/>
</event>
```

Figure 1. Exemplary threshold event template and its parameterized visual representation.

4.2. Event detection

Given a set of events, it is the task of the event detection to determine, for which data records these events evaluate to concrete event instances. This means, that each event formula has to be evaluated regarding each data record of a data set. The result of the event detection is a set of event instances bound to data records.

With respect to static (i.e. invariant) data sets, event detection can be easily realized in a pre-process prior to visualization. On the other hand, for dynamic data sets, whose content could be changing during visualization (e.g. records are inserted, altered, or removed) a pre-process can only initially detect events. If a change to the data occurs, event detection must be re-performed for each data record. This is necessary since a change to a single data record may influence the detection of events regarding other records.

The event detection can be realized by implementing the described formalism. In fact, the event formulas can be mapped to SQL-queries. After query execution a non-empty result-set indicates an evaluation of a formula to true (i.e an event is detected). Techniques required for implementing event detection (e.g. SQL connectivity, XML handling or table data structures) are inherent to modern programming environments like .Net or Java.

4.3. Event representation

The main task of event-based visualization is to represent detected events. Depending on the actual application a variety of events may occur. Therefore, it is important that:

- occurrences of events are highlighted for easy recognition and that
- different event types are visually differentiated so that they can be easily distinguished.

Current visualizations represent events by means of some sort of a temporal axis (e.g. a timeline [6], [7] or an event graph [4]) or by adapting a given visual representation (e.g. different levels of detail in [2]). Therefore, we differentiate:

- *explicit* event representation (i.e. using separate representations for events and data) and
- *implicit* event representation (i.e. adapting the data visualization).

Though our main interest regards implicit representation, in most applications it is useful to additionally provide explicit event visualization upon user request (e.g. by highlighting rows in a table view).

In order to allow an adaptation of a visualization technique it is necessary to find expressive parameters that can be altered. Furthermore, it must be described how these parameters change. Instantaneous changes can be expressed by *actions*. Moreover, in dynamic visualization environments parameter changes can be dynamic as well. Therefore, *processes* are defined, which describe such changes. First results of how visualization can be adapted by means of actions are given in Section 5.

4.4. Model of event-based visualization

We use the classic visualization model as basis for our model. The three main steps of the classic model are filtering (i.e. preprocessing the raw data), mapping (i.e. mapping the data to geometric primitives and their attributes), and rendering (i.e. rendering geometric data to images). We extend this model with our event-based concepts. Therefore, parameters influencing each step of the classic model are introduced. Furthermore, events and actions/processes as well as the event detection are integrated in the model (cp. Figure 2). The presented model of event-based visualization can now be used as basis for an actual realization presented in Section 5.

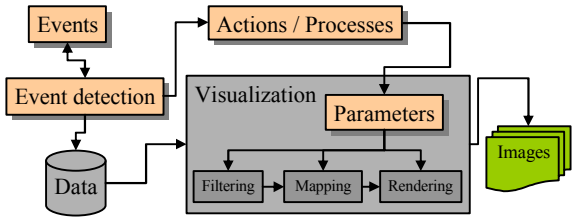

Figure 2. Model of event-based visualization.

5. Realization and preliminary results

Especially when data sets are large, representing all information visually is a demanding task. Parallel Coordinates [11] are commonly used for multivariate data visualization. They constitute a projection of n-dimensional data space onto 2-dimensional screen by means of parallel axes representing attributes. The data records are then represented by line segments crossing the axes regarding the attribute values. With extensions like histograms and brushing (cp. [12], [13]) Parallel Coordinates are a powerful tool, which is applicable for integration of a general event-based model for visualization.

Therefore, we decided to realize our event-based model in an axes-based visualization framework called VisAxes. This led to the following design goals:

- integration of the event-based visualization model,
- expressive visual representation of relational data,
- high degree of adaptability of the visual representation by means of events, and
- high degree of interactivity for easy data exploration.

In order to reach these goals a modular object-oriented class design was chosen, whereas the event-based model is combined with axes-based visualization by means of well defined interfaces.

With respect to axes-based visual representations, conceptually it is important to separate the design of an axis, which is used to represent the values of an attribute, from the arrangement of all axes on the screen. Besides a simple static axis, we have integrated three interactive axes for easy data exploration [14]:

- *scroll axis*, which allows for a selection of a value range of interest,
- *focus+context axis*, which allows for an emphasis of a certain point of interest on an axis, and
- *hierarchical axis*, which is useful for hierarchically structured value ranges.

These axes can be adapted by interactions of a user or automatically as a result to the detection of a certain event.

Depending on the actual data set several axes of certain types are arranged as Parallel Coordinates or as a *Coordinates Wheel*. The Coordinates Wheel is a novel radial arrangement for axes-based visualization (and was introduced in [14] as TimeWheel). It is used if an attribute of a data set provides an order of the records of the data set (e.g. for time-dependent data) and furthermore, the user intends to explore the data with respect to this attribute (i.e. the *attribute of reference*). The basic idea of the Coordinates Wheel is to place one axis representing the attribute of reference exposed in the center of the display and radially arrange further axes representing the remaining attributes. Just like for axes, it is possible to adapt arrangements either interactively or on occurrence of events.

Some examples for adaptation shall be given realized by instantaneous actions. For a first example we visualize a dynamic health data set consisting of the daily reported number of cases for a variety of diseases. Suppose an event $\{\dot{x} \mid \dot{x}.\text{influenza} \geq 300\}$ occurs if the number of cases of influenza exceeds a threshold indicating an epidemic. Automatic brushing can be used in order to emphasize such critical data records. Figure 3 clarifies that brushing is an effective means for adapting the visual representation in an event-based environment.

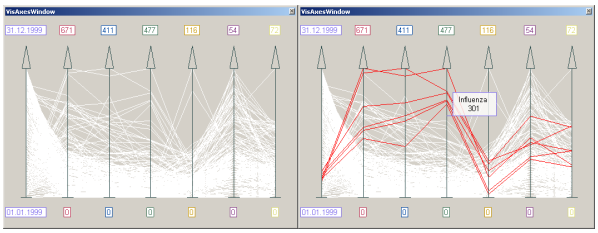

Figure 3. Brushing data records with respect to critical events.

In a second example we use the Coordinates Wheel to represent a multivariate time series of climate related attributes. The Coordinates Wheel in Figure 4 shows six of these time-dependent attributes, each of which mapped to one circular axis. The central axis represents time steps. In our case the relation between time and temperature represented by blue lines is of special interest. If we assume a scientist analyzing the data with respect to a special range of temperatures critical regarding a certain chemical process the event $\{\dot{x} \mid \dot{x}.\text{temp} \geq -1 \wedge \dot{x}.\text{temp} \leq 7\}$ can be defined. On occurrence of such an event it is useful to de-clutter the

visual representation by omitting irrelevant information. In Figure 4 the usefulness of such operation becomes obvious. It can be seen, that the reduction of blue lines in the Coordinates Wheel allows a better insight to the sub-range of interest. The same operation could be performed for more than one attribute to further de-clutter the visual representation. Moreover, other scientists may define different events of interest, which results in adapted visualizations for each of them.

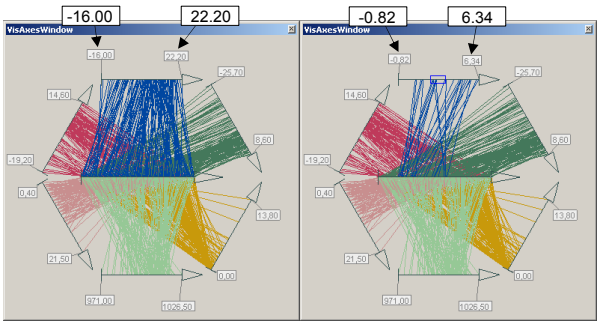

Figure 4. Using a scroll axis with respect to an event reduces irrelevant information and thus, cluttering in the image.

A more global adaptation than in the previous examples is realizable if the arrangement of the axes is calculated according to events. Figure 5 (left) depicts far outliers for the attribute precipitation from the climate data mapped onto the upper left axis connected with blue lines to the central time axis. An aggregate function $outlier : \mathcal{A} \rightarrow \mathbb{R}$ can be used to determine outliers in the event $\{\dot{x} \mid \dot{x}.\text{prec} > outlier(\text{prec})\}$. Once this event is detected the arrangement can be adapted automatically by means of rotating the Coordinates Wheel and scaling its axes. This leads to an arrangement, where the axis representing outliers is centered above the central time axis and furthermore, is provided with more drawing space.

These examples show that a variety of visual adaptations is possible and useful. Further visual event clues have to be examined with respect to their expressiveness and efficiency. Especially for dynamic visualization enhanced techniques like animation or color fading are promising.

6. Conclusion and future work

In this paper we addressed problems that arise if large data sets are analyzed visually by different users regarding different aspects of the data. An event-based approach is suggested as eligible means to solve these problems. We presented a formal description of events that allows for a variety of events to be specified by users. Furthermore, the model of event-based visualization describes how events can be incorporated to visualization.

Our approach has been evaluated in first user tests considering human health data and climate data in combination with simple event templates. In general, our test users gave us positive feedback regarding the adaptation of the visual representation. However, event specification turned out to be difficult.

Therefore, we think our event-based approach increases the flexibility (user specified events) as well as efficiency (relevant information emphasized) of visualization. However, more work has to be done in the future. As yet, events are formally described with respect to tuples. On the other hand, events regarding attributes could be of interest as well. Therefore, a formal description of attribute events is required. Furthermore, in a variety of applications time plays an important role. Although time-dependent events can be expressed by our formalism, the event specification can be improved by a visual interface which takes time into account (e.g. for specifying sequences of events [15]). The development of such a visual editor allowing users to specify events in a drag and drop manner and map them to action/processes is still in progress. Additionally, further parameters of visualization must be revealed to be able to define expressive actions and processes for adaptation of the visualization. Regarding this, it is necessary to investigate solutions for concurrent actions and processes. Finally, after incorporating the mentioned extensions, our approach has to undergo an extensive evaluation in order to fully prove its eligibility.

References

[1] C. North, N. Conklin, and V. Saini, "Visualization Schemas for Flexible Information Visualization", *Proceedings of Symposium on Information Visualization 2002*, Boston, October, 28-29, 2002, pp. 7-14.

[2] K. Matkovic, H. Hauser, R. Sainitzer, and M.E. Gröller, "Process Visualization with Levels of Detail", *Proceedings of Symposium on Information Visualization 2002*, Boston, October, 28-29, 2002, pp. 67-70.

[3] F. Reinders, F.H. Post, and H.J.W. Spoelder, "Visualization of time-dependent data with feature tracking and event detection", *The Visual Computer*, Vol. 17, No. 1, Springer Verlag, Heidelberg, 2001, pp. 55-71.

[4] Reinders, F., *Feature-Based Visualization of Time-Dependent Data*, Dissertation, Delft University of Technology, 2001.

[5] C. Demetrescul, I. Finocchi, and J.T. Stasko, "Specifying Algorithm Visualizations: Interesting Events or State Mapping?", In: Diehl, S. (ed.): *Software Visualization*, LNCS 2269, Springer-Verlag, Berlin, 2002, pp. 16–30.

[6] T. Kunz, "Visualizing abstract events", *Proceedings of CAS Conference 1994*, IBM Canada Ltd. Laboratory and National Research Council of Canada, 1994, pp. 334-343.

[7] M. Khouzam and T. Kunz, "Single Stepping in Event Visualization", *Proceedings of CAS Conference 1996*, IBM Canada Ltd. Laboratory and National Research Council of Canada, 1996, pp. 19-30.

[8] R.F. Erbacher, K.L. Walker, and D.A. Fricke, "Intrusion and Misuse Detection in Large-Scale Systems", *IEEE Computer Graphics and Applications*, Vol. 22, No. 1, 2002, pp. 38-48.

[9] R.F. Erbacher, "A Component-Based Event-Driven Interactive Visualization Software Architecture", *Proceedings of Symposium on Information Systems and Engineering (ISE'00)*, San Diego, July, 2000, pp. 237-243.

[10] Atzeni, P. and V. de Antonellis, *Relational Database Theory*, Benjamin/Cummings, Redwood City, 1993.

[11] A. Inselberg and B. Dimsdale, "Parallel Coordinates: A Tool for Visualization Multi-dimensional Geometry", *Proceedings of IEEE Visualization (Vis'90)*, IEEE Computer Society Press, Los Alamitos, 1990, pp.361-375.

[12] M. Ward, "Xmdvtool: Integrating multiple methods for visualizing multivariate data", *Proceedings of IEEE Visualization (Vis'94)*, IEEE Computer Society Press, Los Alamitos, 1994, pp. 326-333.

[13] H. Hauser, F. Ledermann, and H. Doleisch, "Angular Brushing for Extended Parallel Coordinates", *Proceedings of Symposium on Information Visualization 2002*, Boston, October, 28-29, 2002, pp. 127-130.

[14] C. Tominski, J. Abello, and H. Schumann, "Axes-Based Visualizations with Radial Layouts", *Proceedings of ACM Symposium on Applied Computing (SAC'04)*, Nicosia, Cyprus, March, 14-17, 2004, (to appear).

[15] E. Hajnicz, "Time Structures – Formal Description and Algorithmic Representation", In: Carbonel, J.B. and J. Siekmann (eds.): *Lecture Notes in Artificial Intelligence 1047*, Springer-Verlag, Berlin, 1996.

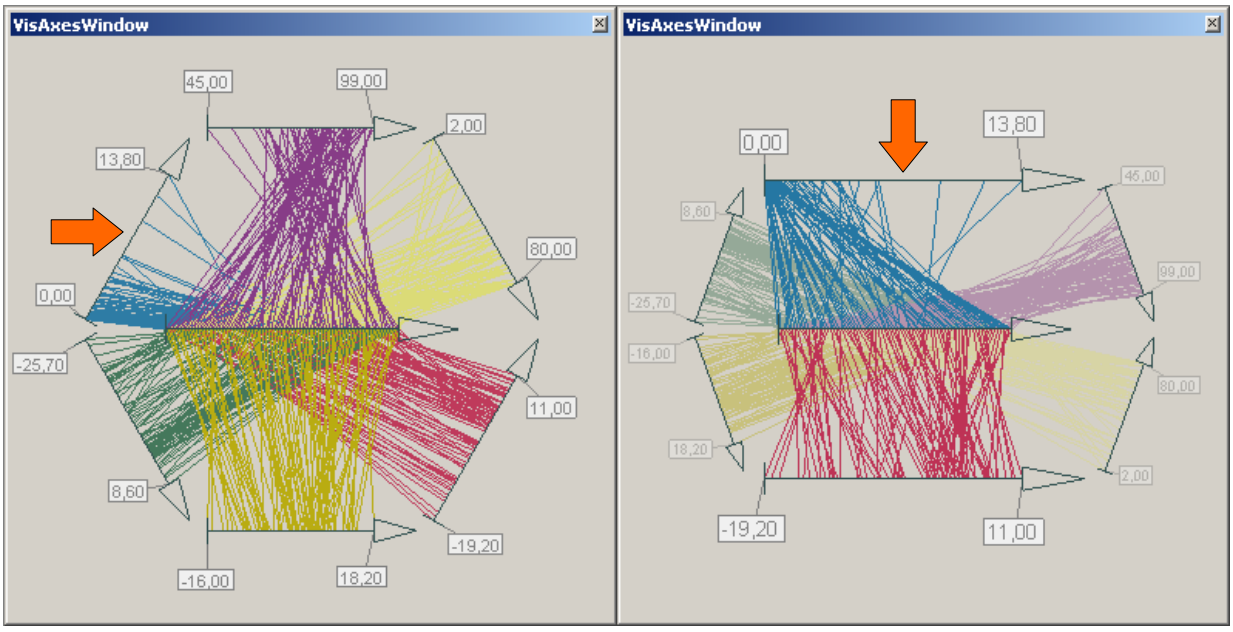

Figure 5. Adaptation of the axes arrangement can be used to focus on axes and to provide focused axes with more drawing space.