PULSE - A time based Infovis Tool

Christian Breil
Media Informatics, University of Munich
Anna Follmann
Media Informatics, University of Munich
Roman Graebsch
Media Informatics, University of Munich

ABSTRACT

There are many visualization tools available in infovis, but most of them are designed to display customized visualizations for specific data sets. PULSE is an approach to handle various time based data sets, no matter which problem domain is addressed. Therefore time based data is organized in streams of events, each having at least a name, starting time and duration. PULSE provides a basic interface to enable the user to compare, analyze and visualize such data by converting XML-based input files into a zoom- and panable visualization. In this paper we want to discuss related work, then describe the design of PULSE and propose future improvements.

Keywords: Information visualisation, time-oriented data, framework, toolkit, user interfaces, interaction, navigation, design

Index Terms: H.5.2 [Information Interfaces]: User Interfaces—

1 INTRODUCTION

Due to the fact that many data sets in science, engineering and business include time oriented informations, visualization of such data gains increasing importance. But even though there have been various approaches to this task in recent years, most of the attempts can not be seen in a generalized context, since most are specifically tailored to only a particular analysis problem. The reason for this especially stems from one fact: visualization of time oriented data is difficult and needs the consideration of various aspects [1].

This paper presents PULSE, a project which addresses the issue of discrepancy between generalization and customization by offering default implementations to visualize time oriented data, while allowing a high level customisation. We summarize related work in visualisation of time oriented data, classify our work in the context of previous approaches and describe the design of PULSE in detail, followed by an overview of the PULSE library and usage.

2 RELATED WORK

In recent years there have been many approaches to visualize time dependent data which offered appropriate solutions to specific problems. Previous publications [1] [3] deal with this issue and provide systematic classifications. They also suggest to pursue the development of general frameworks rather than customized applications in the future. Similar to the preface from J. Heer [2], our goal is to provide a toolkit that supports high diversity while providing common, reusable components.

Following the given systematic analysis principles of [1], PULSE can be classified as a calendar based 2D visualization. Data sets are visualized by static representations arranged along a linear time axis. Each data set, called Event, is defined by a time point (starting time) and a time interval (duration). Events are abstract and univariate in the default implementation, but can be extended to meet spatial or multivariate ones. An abstraction and aggregation of data sets is also possible but not implemented by default.

PULSE is written in Java using the processing graphics library [4], an open source project especially suitable for work within a visual context in a very easy way. Therewith we provide a platform independent toolkit, which allows even inexperienced programmers to expand or adapt the source code, and the resulting visualization.

3 DESIGN OF THE PULSE TOOLKIT

Following Schneiderman’s mantra of ‘overview first, zoom and filter’, then details on demand [5], PULSE provides both an overview and a detail window to display the representations of data sets. Within the overview the different used data sets, also called “event streams”, are arranged along a time axis. To allow a better orientation, events of the same kind are grouped into color coded “streams”. A number of interactions enables the user to manipulate the displayed time slot and therefore change the scaling of the overview. This way PULSE supports typical overview tasks like finding correlations or browsing events. To display more information about specific data sets, events can draw their tailored representation into the detail window. To address the issue of customisation PULSE enables each data set taking into account specific aspects, as for example data types, and drawing a customized visualisation to generate a maximum of insight. Furthermore, there is the possibility to use additional features like generating a pie diagram to compare one stream to another. In the following subsections we will introduce the data design of PULSE, then describe the visual design, followed by a summary of interaction and navigation functions.

3.1 Data Design

As mentioned above, PULSE enables a user to display time based data. In most cases we cannot know beforehand how a specific data set looks like, e.g. a life-logged data stream for jogging might have other attributes than a log of an SVN-repository. To design an abstract data format we searched for similarities between different scenarios of time based data streams and found four factors which they all have in common:

- A time based data-stream consists of unique events of the same type. That means, a sleeping-stream wont contain a sports-event.
- An event has a starting and an ending time
- An event has a description or a name
- An event can have other optional attributes

To map that description onto a data format, we first defined an XML schema, called StreamML, to represent a single event stream. That schema leans to the definition of TreeML J. Heers uses in his Prefuse Framework to describe a tree structured Graph [2].

Similar to that XML schema, the data model of the PULSE toolkit is built. We defined some interfaces like Stream, Event and ImageEvent, which provides additional support for displaying images. For each of those interfaces there exist either abstract base implementations or example implementations, a programmer can derive from. To read a valid XML file we provided a SAX Parser based abstract implementation, as well as several example implementations to open streams of different event types. Any programmer using our toolkit can easily create an appropriate and suitable
reader for his specific stream. This is needed, because it cannot be
determined beforehand which subclass of Event is to be created.

3.2 Visual Design

3.2.1 Layout

PULSE uses a liquid layout. Dependent on the given space the lay-
out assigns a certain space to each of the seven elements of the
graphical user interface: Logo, calendar, overview visualisation,
detailview visualisation, timeslider, button panel and stream details.
Figure 1 displays the arrangement of the elements.

3.2.2 Individual elements and their relevance

(1) The PULSE logo with exit button. (2) The calendar, it’s lines are
drawn to indicate position of the events. The calendar divides the
displayed time period into the most convenient time periods such as
hours and days. (3) The left panel provides space to display the
details of the streams, as well as user input to control the streams.
(4) The overview displays the overall representation of streams and
their events. Vertical height is shared between the streams. Events
within a streams are drawn as rectangles, their X position depending
on their start time and their width depending on their duration. To
avoid overlapping and clutter following events are drawn one below
the other in a cyclic behavior. (5) The button panel is located in the
lower left corner , (6) the timeslider assigned to the lower middle
enables the user navigating in the time and displays the start and
duration of the current window of time. (7) The detail window at
the right displays the detail representation of streams and events.

3.3 Interaction Design

3.3.1 Basic Interactions

Basic interactions on events are done with simple mouse operations.
Hovering an event causes it to display its name, while left-clicking
it toggles the selection of an event. Right-click deselects all events.
Interactions on single streams can be performed by using the "mi-

" button (removing a stream) and the "select" button (toggle selec-
tion) in the left panel. New streams can be added by pushing the
"plus button" (adds a new standard stream) or other customized
buttons to add custom streams. By default, the pie diagram but-
tton creates a diagram that will compare the total time of all event
selection) in the left panel. New streams can be added by pushing
"plus button" (adds a new standard stream) or other customized
selection) in the left panel. New streams can be added by pushing
"plus button" (adds a new standard stream) or other customized

"plus button" (adds a new standard stream) or other customized
drawings to indicate position of the events. The calendar divides the
displayed time period into the most convenient time periods such as
hours and days. (3) The left panel provides space to display the
details of the streams, as well as user input to control the streams.
(4) The overview displays the overall representation of streams and
their events. Vertical height is shared between the streams. Events
within a streams are drawn as rectangles, their X position depending
on their start time and their width depending on their duration. To
avoid overlapping and clutter following events are drawn one below
the other in a cyclic behavior. (5) The button panel is located in the
lower left corner , (6) the timeslider assigned to the lower middle
enables the user navigating in the time and displays the start and
duration of the current window of time. (7) The detail window at
the right displays the detail representation of streams and events.

3.3.2 Navigation

PULSE employs manual panning and zooming for navigation in the
time via direct manipulation. Panning is performed by dragging
the background of the overview visualisation, zooming is performed
with the mouse wheel. The timeslider ca be used to change the cur-
rent time window. Dragging the left or right border of the timeslider
manipulates the start or end time as well as the duration of the time
window. By dragging the middle of the timeslider, one can adjust
the section shown in the overview. All navigation and display el-
ments of PULSE are linked together. Therefore a change in one
element updates all other elements and vice versa.

4 THE PULSE LIBRARY

The PULSE library contains several Java classes to support a pro-
grammer to create custom PULSE visualizations. First of all it is
necessary to define an XML file fitting the data it should represent.
Our StreamML enables a user to define a stream containing events
with specific Attributes, to create different data sets tailored to spe-
cific need. The second step is to create a specific reader and event
class suitable for this event stream. To create a reader it is sim-
ply needed to derive from "AbstractStreamReader" and implement
the constructor, the method "createCustomStreamHandler" as well
as a class deriving from "AbstractStreamHandler". The latter class
needs a constructor as well as a the generic Method "createEvent",
which should return an event of the specific type. Therefore it is
needed to create a class representing a single event, which can be
derived either from "AbstractEvent" for full control about an events
behavior, "SimpleEvent" or "SimpleImageEvent" for basig image
handling. In the event class it is possible to override the methods
"draw" and draw "detail". Both methods are called with an Area-
Object which describes where to draw. We added two different
examples to our toolkit, where subclasses can see how it can be
done.

5 Conclusion

In this paper we have introduced PULSE, an infovis tool which
enables a user to explore various time based data sets. Due to
its toolkit nature, PULSE can be used for displaying various dif-
terent data sets, without the need to implement basic functionality
by oneself. PULSE is completely open-source, and available at
https://wiki.medien.ifi.lmu.de/Main/IVPulse. The basic version of
PULSE could be further extended in different ways to improve the
usability of the application. In a next step, it would be advisable
to introduce a better algorithm to set the horizontal positioning of
events to avoid cluttering. Another feature that would upgrade the
toolkit is a search function. It could allow the user to use differ-
ent constraints, e.g. it might be interesting to search for all events
that happen during some other event, or showing all events taking
place in a cyclic behaviour. Further, apart from an improved pie
diagram, other aggregating or summarizing detailed views could be
implemented to generate insight in specific domains.

References

Visualizing time-oriented data—a systematic view. Comput. Graph.,
information visualization. In CHI ’05: Proceedings of the SIGCHI
conference on Human factors in computing systems, pages 421–430,
New York, NY , USA, 2005. ACM.
interactive web graphics. In Siggraph ’03: ACM Siggraph 2003
Web Graphics, pages 1–1, New York, NY, USA, 2003. ACM.
[5] B. Shneiderman. The eyes have it: A task by data type taxonomy for
Symposium on Visual Languages, page 336, Washington, DC, USA,