prefuse Tutorial
beta release 2006.07.15

Wolfgang Aigner
wolfgang@is.wwt.ac.at
http://www.is.wwt.ac.at/wolfgang/
27.11.2006

What is prefuse?
Extensible software framework to create interactive
information visualization applications
Java
Open Source

Download and Build

Download
Homepage: http://prefuse.org
Beta release
Download link in the upper left corner
Ubuntu
Build
Set "JAVA_HOME" environment variable
Ant build script build.xml
Ant = Java Build System
highly recommended in general
Start Ant script via build.ch (Linux/Mac) or build.bat (Win)
1) Build classes and jar
Option "all"
2) Build API documentation
Option "api"
JAVA_HOME="/usr"; export JAVA_HOME
in build.ch apt

InfoVis Reference Model

188.308, 1.0h

Why prefuse?
- OS independent (pure Java)
- available demos and examples
- good documentation
- active community
- many built-in components
  - I/O methods
  - table, graphics, trees
- layouts
- components for color, size, and shape
- encodings
- filter techniques
- animation (e.g., smooth transitions)
- dynamic queries / interactive filtering
- integrated text search
- physical force simulation engine
- SQL-like expression language for writing queries

How to run demos

10 Available demos
  - AggregateDemo - Vis of groups of graph nodes
  - Congress - Scatterplot of annual income of congressmen in different states
  - DataMountain - Spatial arrangement of book covers
  - FishEyeMenu - Focus + Context list of numbers
  - GraphView - Graph vis with adjustable parameters for layout
  - RadialGraphView - Radial graph of a social network
  - ScatterPlot - Scatterplot of iris dataset
  - TreeMap - Treemap vis
  - TreeView - DOTTree
  - ZipDecode - Vis of US zip codes
1) Go to "build" folder
2) Launch demo of interest
   java -cp prefuse.jar; demos.jar prefuse.demos.\name_of_demos\here

27. November 2006 UE Informationsvisualisierung
Basic Architecture

Visual Abstraction

Visualizations are created by generating visual representations of data records (VisualItems) in data tables.

central data structure: Visualization
manages mappings between source data and VisualItems
Table --> VisualTable
Graph --> VisualGraph
Tree --> VisualTree

manages VisualItems
visual representation of data elements
interactive visual object
properties of source data + visual properties
Location, color, size, shape, text
specialization of VisualItems into NodeItems and EdgeItems for graphs

Process is called Visual Mapping

Built-in Actions

Assignment
ColorAction
DataColorAction
SizeAction
DataIconAction
ShapeAction
DataShapeAction

Filter
VisibilityFilter
GraphDataFilter
FishEyeFilter

Layout
ArcLayout
ArcInLayout
GridLayout
CircleLayout
StackedAreaChart
RandomLayout
SpecifiedLayout
CollapseStackLayout
CollapseHiddenLayout

Source Data & Data Tables

Source Data
Raw data
formatted files
databases
etc.

Data Tables
internal storage and management for data read from source data
table columns are typed
each column contains values for a named data field with a specific
data-type
graphs and tables are internally also stored as tables (nodes, edges)
Data records do not contain any visual information like assignment
attributes or color settings.
Instead, own visual analogs are created

Visual Abstraction 2

Specific visual mappings are provided by Action
modules
Actions are independent processing modules that operate on the VisualItem instances in a Visualization
setting item visibility, computing layouts, assigning color
values, etc.
can be grouped into ActionLists
Actions can be run once or repeatedly over a time interval,
controlled by an ActivityManager

Interactive Views

actual drawing of VisualItems is done by Renderers
responsible for drawing items and computing item bounds
choice of Renderer is done by RendererFactory that is assigned to
a Visualization
Display component acts as a camera onto the contents of a
Visualization
is the component where the actual drawing takes place
draws all the items within its current view, and can be panned,
zoomed, and rotated
first-class user interface components
can be added into Java applications and applets
single Visualization can be associated with multiple Display
instances
multiple views, overview + detail, small multiples
Interactive Views 2

Controls for user interaction
- process mouse or keyboard actions on the Display and on individual VisualItems
- built-in controls
  - selecting focus items
  - dragging items around
  - panning
  - zooming
  - rotating

Dynamic Query Bindings
- create a binding between a column of table data and an expression
  - Predicate (or query) over that column
- bindings can automatically generate appropriate user interface components (e.g., sliders, radio buttons, check boxes, text search boxes, etc)

1) Load data

data sources
- File
- Database
- Custom

prefuse data structures
- Table
- Graph
- Tree

Application Building Overview

Overview

2) Create Visualization

maps loaded data to Visual Abstraction
- Tables, Graphs, and/or Trees are added to the Visualization
- manages VisualItems
  - visual representation of data elements
  - interactive visual object
  - properties of source data + visual properties

3) Create RenderFactory and register with Visualization

is responsible for assigning Renderers to VisualItems
- Renderers do the actual drawing of VisualItems

DefaultRendererFactory
- EdgeRenderer for any EdgeItems
  - straight-line edges by default
- ShapeRenderer for all other items
  - draws items as basic shapes such as squares and triangles

4) Construct processing Actions (Visualization Operators)

operate on the visual abstraction
- e.g., setting the location, color, size, and shape of visual items or animating these properties between different configurations
5) Initialize Display(s)

for viewing and manipulating visual items

6) Specify interactive behavior

by adding Controls to the Displays

Search and filtering over data items can be added using "dynamic query bindings"

Coordinates in pre fuse

two different coordinate systems

- absolute coordinates
  - device-independent, logical coordinates
    - all visual attributes like positions or sizes are defined in absolute coordinates

- view coordinates
  - device-dependent (screen) coordinates

Transformations between absolute and view coordinates are done automatically by Java painting routines

Example 1

Built-in example network visualization "Example.java"

2) Create Visualization

create Visualization object

add graph to visualization

// add the graph to the visualization as the data group "graph"
// nodes and edges are accessible as "graph.nodes" and "graph.edges"
// Visualization vis = new Visualization();
vis.add("graph", graph);
3) Create RenderFactory and register with Visualization

create a new LabelRenderer (to see text labels on the nodes)

create a new DefaultRendererFactory

uses the new label renderer as the default renderer for all non-edge items

all Edgetight items will use the default EdgeRenderer

// draw the "name" label for Nodetight
LabelRenderer r = new LabelRenderer("name");
s.font = new Font(10, Font.PLAIN, Color.LIGHT_GRAY);
// round the corners

// create a new default renderer factory

// return our name label renderer as the default for all non-Edgetight
// includes straight line edges for Edgetight by default
vis.setRendererFactory(new DefaultRendererFactory(r));

4) Construct processing Actions

setting up visual encodings by creating Action modules that process the Visualitems in the Visualization

a) ColorActions

b) Animated Layout

c) Add ActionLists to Visualization

a) Color Actions 2

b) Animated Layout

All Action instances can either be parameterized to run once (the default), or to run repeatedly within a given time duration continuous update by setting parameter to "INFINITY"

add a ForceDirectedLayout to assign the spatial positions of the elements of the graph

add a RepaintAction to signal that any Displays should be repainted after the layout has been recomputed

// add the actions to the visualization
vis.putAction("color", color);
vis.putAction("layout", layout);

27. November 2006 25

27. November 2006 26

27. November 2006 27

27. November 2006 28

27. November 2006 29

27. November 2006 30
5) Initialize Display(s)

create a Display for the visualized data
// create a new Display that pull from our Visualization
Display display = new Display(vis);
display.setSize(720, 300); // set display size

6) Specify interactive behavior

add three interactive controls to the Display
DragControl for dragging Visualitems around with a left-click mouse drag
PanControl for moving the Display region with a left-click mouse drag on the Display background
ZoomControl for zooming the display in or out with a vertical right-click mouse drag
default settings of the Controls
mouse button used to trigger the control and other settings can be changed by using alternative constructors.

7) Launching the visualization

add the Display to a new application window
create a new JFrame instance
run the color assignment action list
start continuously-running layout list

Example 2

Line Chart
Climate data (Avg. temperatures over the course of a year)
draw points and lines
points can be drawn directly using axis layouts
lines have to be created from points "manually"
ranges sliders for both axes

Compile & Run

Compile
javac -cp lib/prefuse.jar Example.java

Run
java -cp lib/prefuse.jar:. Example

Documentation & Help

Manual
http://prefuse.org/doc/manual/
FAQ
http://prefuse.org/doc/faq/
Forum
API Documentation
http://prefuse.org/doc/api/
Demos
included in download package
InfoVis:Wiki page
http://www.infovis-wiki.net/index.php/Prefuse
Mini Glossary

Visualization
- maps loaded data to Visual Abstraction
- Central repository that manages VisualItems, RenderFactory, Actions, and Displays

VisualItem
- interactive visual representation of data elements

Action
- Actions are independent processing modules that operate on the VisualItem instances in a Visualization

Renderer
- responsible for drawing items and computing item bounds

RenderFactory
- is responsible for assigning Renderers to VisualItems

References

prefuse Manual
http://prefuse.org/doc/manual/

[Card et al., 1999] Card, S. and Mackinlay, J. and Shneiderman, B.,
**prefuse**

**package guide**

**prefuse.data.io**
Readers/writers for data files

**prefuse.data.io.sql**
SQL database connectivity and query management

**prefuse.action**
Action modules for filtering, layout, color, shape, and size assignment, distortion, and animation

**prefuse.render**
Renderer modules for drawing individual VisualItems. RendererFactory interface for assigning renderers to items.

---

**Source Data**
- formatted files: CSV, Tab-Delimited, GraphML, TreeML
- SQL databases: MySQL, Postgres, etc.

**Data Tables**
- **prefuse.data**
  - Table, Graph, and Tree data structures, Tuple, Node, and Edge classes for individual data items
- **prefuse.visualization**
  - Central repository managing the VisualItems, RendererFactory, Actions, and Displays.
- **prefuse.visual**
  - VisualItem instances representing visual, interactive elements

**Visual Abstraction**

**Interactive Views**
- **prefuse.display**
  - Provides a view onto the contents of a Visualization. Supports transforms such as panning and zooming.

**prefuse.control**
Interactive Controls for manipulating VisualItems and performing panning and zooming.

**prefuse.data.query**
Filter data using interactive components such as sliders, radio buttons, and check boxes. Perform full text searches over data fields.

---

**1) Load Data**
**2) Create Visualization**
**3) Create RendererFactory**
**4) Construct processing Actions**
**5) Initialize Display(s)**
**6) Specify interactive behavior**