NetBeans™ Common Framework for Information Visualization

David Kašpar
Peter Liu
Martin Rýzl

Sun Microsystems, Inc.

TS-8943

Goal

Learn the building blocks of a visual application and the requirements and the implementation of a open-project framework that helps you build it on the NetBeans™ platform application framework

Agenda

Introduction to Visualization
Background
Framework Requirements
Proposed Implementation
Summary

Agenda

Introduction to Visualization

Background

Framework Requirements

Proposed Implementation

Summary

Introduction to Visualization

- Organizes your data in graphical forms
  - Trees—Hierarchy
  - Graphs—Relations between objects
  - Nested structures—An element is another structure
  - Complex structures—Mixture
- Improves understanding of your data
- Improves ease of use

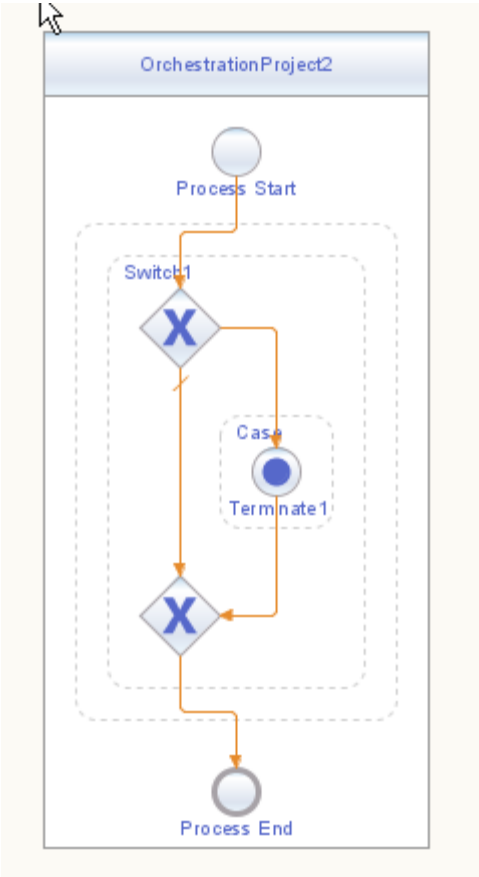
Traditional Approach

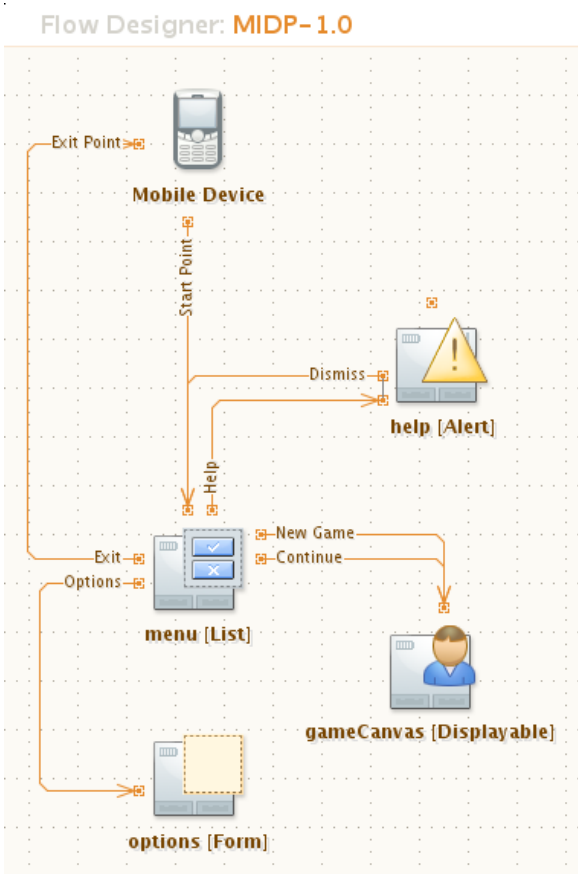
- Text editor—Java™ source code files, XML documents
  - Easy to implement and portable
  - Hard to understand
- Swing-based UI—Property sheets, tree nodes
  - Part of JDK™ software and accessible to developers
  - Not targeted at showing relationships

Visual Approach

- Diagramming—UML Tools
  - Easy to understand interrelationship of your data
  - Easy to use
  - Java 2D™ API is too low-level and requires expertise
  - Usually requires a framework to build it

Diagramming Examples





Issues Facing Developers

- Java 2D API, Swing
  - Too low-level drawing and events
- Needs to implement a visual model
- Needs to implement event support
- Needs to implement the visualization itself

Building Blocks of Visual Applications

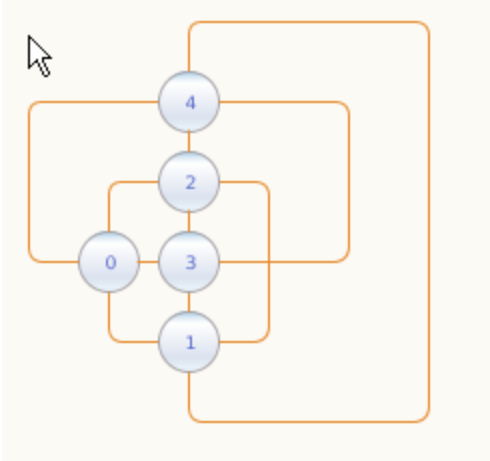
- User model
  - Application domain specific
- Visual model
  - Framework specific
  - Used for rendering
- User interaction
  - Common/expected user actions
  - Feedback

Visualization Using Graphs

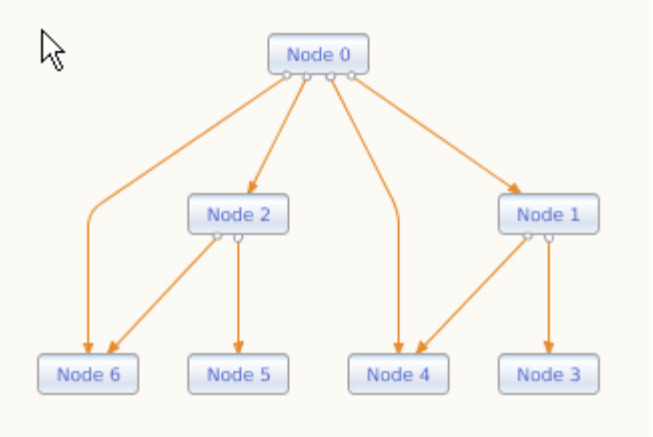
- Graph structures
  - Nodes—Entities
  - Edges—Relations
  - Pins—Connectors for edges
  - Nested graphs
- Graph organization
  - How to avoid edge overlapping?
  - Layout algorithms

Graph Layouts

- Orthogonal layout
  - Nodes are arranged in a rectangular grid
  - Edges have 90-degree bends



- Hierarchical layout
  - Nodes are organized in a hierarchy



Incremental Mode of Layout

- Incrementally changes layout
  - Node created/deleted/moved/resized
  - Edge created/deleted/relinked/reshaped
- Required for interactive visualization
  - Creating a new node must not layout whole scene
  - Moving a node must not layout whole scene
  - Free movement
    - Less restriction more usable
    - Do not be smarter than users

Agenda

Introduction to Visualization

Background

Framework Requirements

Proposed Implementation

Summary

Visual Frameworks in NetBeans IDE

- VisDev
  - General purpose drawing framework based on Scalable Vector Graphics (SVG)
  - Used to build BPEL Designer in NetBeans Enterprise Pack software
- GraphLib
  - Derived from NetBeans Mobility Pack
  - Open-sourced and hosted on http://graph.netbeans.org
  - Graph-oriented visualization
  - Java 2D API/Swing based

VisDev Project

- Support for descriptive visual using SVG
- Extensible event handling support
- Can create arbitrary visual structures other than graphs
- Flexible and easy to program
- No built-in components
- No NetBeans IDE integration

GraphLib Project

- Graph-oriented model
- MVC architecture
- Intuitive and fast UI
- Plug-able model, look and feel with built-in impl.
- Tons of small features
  - Multi-view, multi-layers rendering, in-place editing, zoom, undo/redo, object states
- Not a general visualization library
- Missing descriptive language for UI

Agenda

Introduction to Visualization

Background

Framework Requirements

Proposed Implementation

Summary

Framework Requirements

- Modern look and feel
  - UI guidelines
  - Common user actions
- Graphing support
- General visualization
- Built-in reusable components

Look Requirements

- Re-use existing Look patterns
- Using unified colors to identify object states
  - Normal, hover, selected, highlighted
- Look candies—Transparency, shadows, etc.
- Clean Look without distracting effects
- Feedback
  - Model changes
  - User actions
  - Different feedback for different action

Feel Requirements

- Graphical representation is in 2D space
  - Easy to control with pointing device like mouse
  - Keyboard is not used as a primary device
- User actions must not clash at any time
- Less UI actions, more gain
  - Expose important/frequent action
  - Hide unimportant/infrequent action
- Unified common/expected actions
  - Select, Copy/Move, Edit, Create, Delete
- Edit in place

Controller Devices

- Mouse events
  - Click (left, right), double-click, drag and drop
- Keyboard should be used minimally
  - Text input
  - Key modifiers for mouse events
    - Two keys: Shift, Ctrl or their alternatives (Alt is clashing)
- Operation System diversity
  - Example: Different ways to show context menu
    - Right-button-click
    - Middle-button-click
    - Ctrl key with mouse-button click, long-click

Feedback

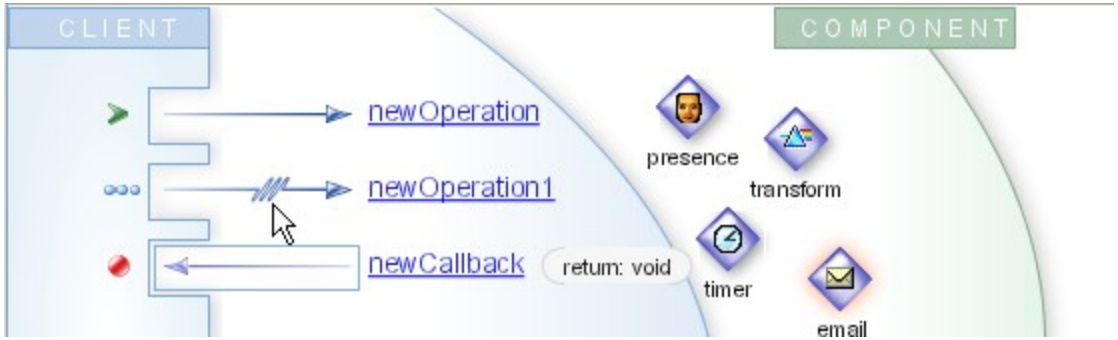
- Feedback required for
  - Each user action
  - Each start point of user action
  - Process/progress of long-term user action
- User should not lose contact with objects
  - Smooth, animated, and minimal reformatting/layout changes
  - Persistent

Graphing Support

- Graph primitives
  - Node, Edge, Pin
  - Nested graph support
- Easy to manipulate
- Layout support
  - Incremental mode of layout required
  - Pluggable layouts

General Visualization

- Support for Form editing
  - Interactive resize elements
  - Preview of moving element with snapping
- Support for any arbitrary structures



- Environment integration
  - Integrated selection model

Built-in Reusable Components

- Defined with
  - Visual model representation
  - Visual definition
- General components
  - Icon node
  - Control-point based edge
- Specialized components
  - UML: Class, Association, Note
  - Logic: Switch, Loop

Agenda

Introduction to Visualization

Background

Framework Requirements

Proposed Implementation

Summary

Programming Model

- Similar to Swing
  - A tree of generic visual elements called widgets
- Each widget is composite
- Built-in primitive widgets—Image, label
- Each widget has location, boundary, layout, ...
- Layout resolves location and boundary of sub-widgets

Actions

- Complex listener handling mouse and key events
- Assigned to widgets
- Swing events are processed by all actions of all affected widgets until they are consumed
- Built-in high-level actions
  - Mouse Hover, Select, Edit, Popup Menu
  - Move, Resize
  - Zoom, Pan
- Tools—Allows switching between sets of actions

Code Sample

```
{
    // Creates a scene with an icon node
    Scene scene = new Scene ();

    Widget iconNodeWidget = new Widget (scene);
    iconNodeWidget.setLayout (
        new SerialLayout(SerialLayout.Orientation.VERTICAL));
    scene.addChild (iconNodeWidget);

    iconNodeWidget.addChild (
        new ImageWidget (scene, myIconNodeImage));

    iconNodeWidget.addChild (
        new LabelWidget (scene, "My Icon Node"));

    JComponent view = scene.createView ();
}
```

Connection Widget

- A connection between two locations—Source and target
- Positions are resolved by anchors
- Control-points defines path
- Control-points are calculated by path routers
- Built-in anchors, anchor shapes, routers

Graph Support

- Support for two graph-oriented models
  - Nodes, edges
  - Nodes, edges, pins
- Scene additionally contains
  - Graph model
  - UI definition—Mapping graph elements to widgets
- Graph nodes layout is separate and independent on widget layout
- Edge routing is done by connection widget

Large-Scale Visualization

- On-demand creation of
  - Objects in user model
  - Widgets in visual model
- Widgets clipping and hiding
- Levels-of-Details widget
  - Based on scene zoom factor
  - Widget and its children are hidden, shown, or partly shown

NetBeans IDE Integration

- Help developers build visualization tools targeted for NetBeans platform application framework
- Environment integration
  - Multi-views
  - Palette
  - Selection
  - Property sheets
  - Context menu
  - Actions

DEMO

Hello World

NetBeans IDE Integration

Agenda

Introduction to Visualization

Background

Framework Requirements

Proposed Implementation

Summary

Roadmap and Plans

- Current state
  - Framework is still in-development
  - Open-sourcing in-progress, hosted on netbeans.org
  - Roadmap synchronized with NetBeans platform releases
- Contribution—Join the community
  - Come with new ideas
  - Develop specialized visualization extensions
  - Design new UI look and feels

Summary

- The building blocks of visualization
- Existing projects used in NetBeans platform application framework
- The framework requirements
- The proposed implementation

For More Information

- NetBeans IDE project
  http://www.netbeans.org
- GraphLib project
  http://graph.netbeans.org

Q&A

NetBeans™ Common Framework for Information Visualization

David Kašpar
Peter Liu
Martin Rýzl

Sun Microsystems, Inc.

TS-8943