FACULTY OF !NFORMATICS

Jazz: An Extensible Zoomable User Interface Graphics Toolkit in Java

Benjamin B. Bederson, Jon Meyer, Lance Good
Human-Computer Interaction Lab, University of Maryland

Proceedings of the ACM Symposium on
User interface software and technology (UIST), 2000, p. 171-180

UE Informationsvisualisierung, SS 2004 **Gruppe 3**

WOLFGANG AIGNER
Matr.Nr.: 9755342
aigner@asgaard.tuwien.ac.at

MARTIN TOMITSCH
Matr.Nr.: 9726166
martin.tomitsch@rise.tuwien.ac.at

Content

Zoomable User Interfaces
Semantic Zooming
Jazz
Example Application
Monolithic vs. Minilithic Approach
Scene Graphs
Piccolo
Our Implementation

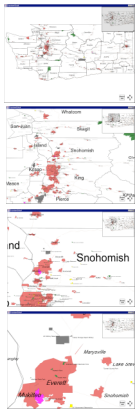
Zoomable User Interfaces

using a virtual canvas and cameras for displaying information

- abstract data is mapped to 2D graphic representations (onto a virtual canvas)
- a portion of the virtual canvas is seen on the display through a virtual "camera"

requirements for ZUIs:

- support for custom application graphics and traditional widgets
- performance should scale with complex scenes
- view navigations (zooms and pans) should be smooth and continuously animated
  - space-scale diagrams (Furnas and Bederson)
  - smooth and efficient zooming and panning (Wijk and Nuij)



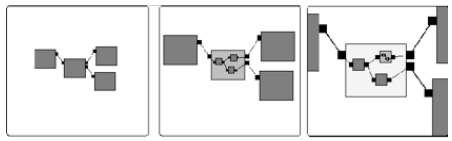
Semantic Zooming

conventional geometric zoom

- objects change only size

semantic zoom

- affects not only size of objects but their representation
- objects change shape or even their very presence in the display



Jazz

predecessors: Pad/Pad++

- written in tcl/tk
- supports semantic zooming
- monolithic and inflexible

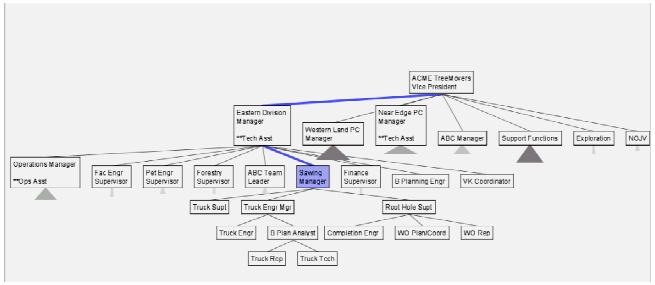
Jazz

- a minilithic Java toolkit for creating ZUI applications
- uses the Java2D renderer
- supports embedded Swing widgets
- uses a basic hierarchical scene graph model

successor: Piccolo

Example Application

Space tree: a novel node-link tree browser



Monolithic vs. Minilithic

monolithic (standard) approach

relatively small number of classes to provide a large amount of functionality

inherited by all of the widgets in the toolkit

drawbacks

- leads to a very complex hard-to-learn top level class
- developers are forced to accept the functionality provided by the top-level class
- all inherited classes pay for any extra functionality

minilithic design

one feature per class

classes are working together

advantages

- more flexible
- code is less complex

Scene Graph (1/2)

hierarchical scene graph model with cameras

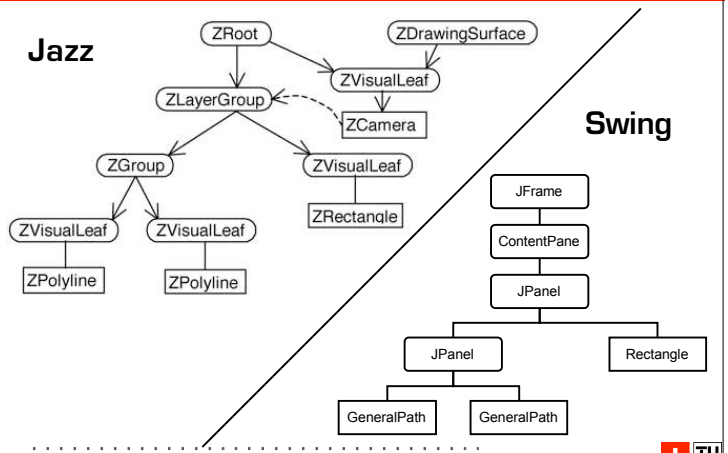
graph of visual and non visual elements

- **nodes**
  - relationships between objects
- **visual components**
  - geometry and color
- **cameras**
  - display a view of a scene graph visually

functionality by composing simple objects

- rather than by class inheritance

Scene Graph (2/2)



Piccolo

descendant of Jazz

goal: similar feature set + easier to use

complexity

no separation between nodes and visual components (no ZVisualLeaf decoration)

speed / memory

less memory usage (65 kB vs. 450 kB .jar)

features

not supported yet: hyperlinks, clip group, constraint group, fade group, layout managers, selection group, spatial index, embedding Swing objects

Our Implementation (1/2)

Basic Idea:

Using Piccolo to implement a zoomable version of "PlanningLines"

PlanningLines

further development of LifeLine/GANTT chart

visualizing temporal (planning) activities

hierarchical decomposition

temporal uncertainties

Our Implementation (2/2)