Software Visualization

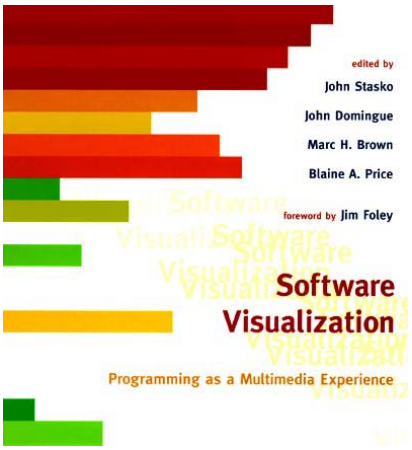
CS 7450 - Information Visualization
April 6, 2006
John Stasko

Software Visualization

- Definition

"The use of the crafts of typography, graphic design, animation, and cinematography with modern human-computer interaction and computer graphics technology to facilitate both the human understanding and effective use of computer software."

Price, Baecker and Small, '98



Challenge

- Software clearly is abstract data
- Unlike much information visualization, however, software is often dynamic, thus requiring our visualizations reflect the time dimension
  - History views
  - Animation
  - ...

Subdomains

- Two main subareas of software visualization
  - Program visualization - Use of visualization to help programmers, coders, developers. Software engineering focus
  - Algorithm visualization - Use of visualization to help teach algorithms and data structures. Pedagogy focus

Subareas

- Program Visualization
  - Software engineering
  - Debugging
  - Program analysis
  - Systems
- Algorithm Visualization
  - Pedagogy
  - Systems
  - Use in classroom
  - Empirical study

Today's Talk

- This is a very big area, so talk will just give a flavor of the kinds of techniques and systems that have been created
  - Lots of figures
  - Some demos and videos

Program Visualization

- Can be as simple as enhanced views of program source
- Can be as complex as views of the execution of a highly parallel program, its data structures, run-time heap, etc.

Enhanced Code Views

Primarily used as a documentation aid

```
protected double distance(int i, int j) {
        double dx, dy;

        dx = loc[i][0] - loc[j][0];
        dy = loc[i][1] - loc[j][1];
        return Math.sqrt(dx*dx + dy*dy);
}

protected void visit(int k) {
        int t;

        now += 1;
        val[k] = now;
        for (t=0; t<n; ++t)
                if ((adj[k][t] != 0) && (val[t] == 0)) {
                        dfs.Visit(t, now);
                        visit(t);
                        dfs.Backtrack(k);
                }
        dfs.Done(k);
}

public DFS() {
        dfs = new DFSAnimator();
        loc = new double[GraphView.MAXVERT][2];
        adj = new int[GraphView.MAXVERT][GraphView.MAXVERT];
        val = new int[GraphView.MAXVERT];
}
```

SeeSoft System

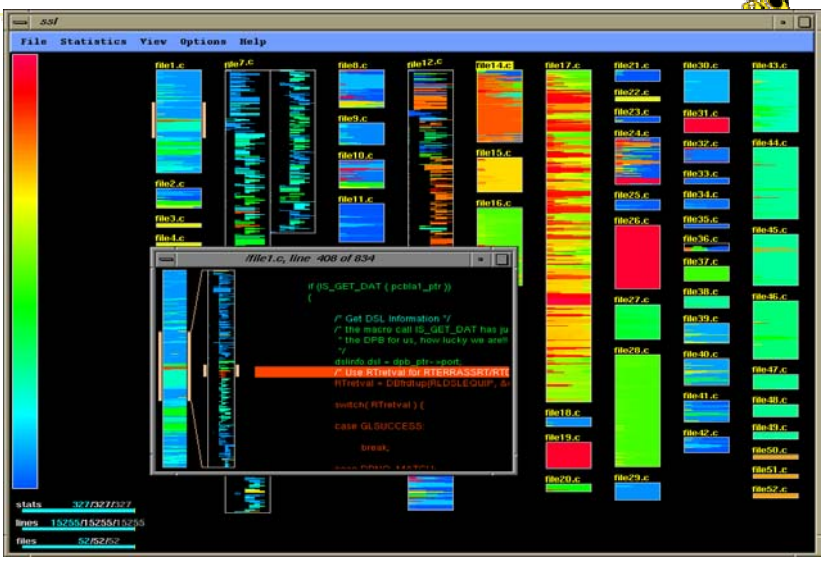
- Pulled-back, far away view of source code
- Map one line of source to one line of pixels
  - Can indicate line indentation, etc.
- Like taping your source code to the wall, walking far away, then looking back at it

Eick, Steffen and Sumner
IEEE ToSE '92

System View

15,000 lines of code in 52 files

Video



Use

- Tracking (typically means mapping this data attribute to color)
  - Code modification (when, by whom)
  - Bug fixes
  - Code coverage or hotspots
- Interactive, can change color mappings, can brush views, and so on

Tarantula

- System developed here at GT
- Utilizes SeeSoft code view methodology
- Takes results of test suite run and helps developer find program faults
- Clever color mapping is the key

Eagan, Harrold, Jones & Stasko
InfoVis '01
Jones, Harrold & Stasko
ICSE '02

Tarantula View

Demo

Paper Recap

"CVSscan: Visualization of Code Evolution"
Lucian Voinea, Alex Telea, and Jarke J. van Wijk
SoftVis 2005

Summer Adams

Overview

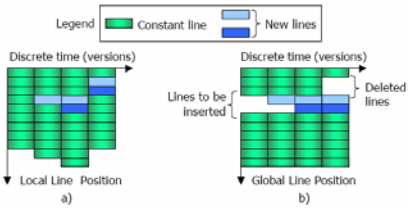
- Task is to gain insight into the software system by visualizing the evolution of changes
- Includes code organization, semantics, and attributes
- Focus on software maintenance

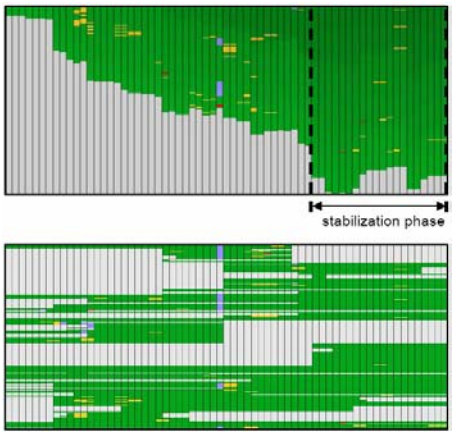
Approach

- Source Code
  - CVS as version control source
  - UNIX's *diff* command to compare versions
  - Six different line status values represented
- Visualization
  - Each line of code is a pixel line
  - One file is shown at a time
  - Shows by line status, construct, or author

Details

- File vs. Line Based





Full Application

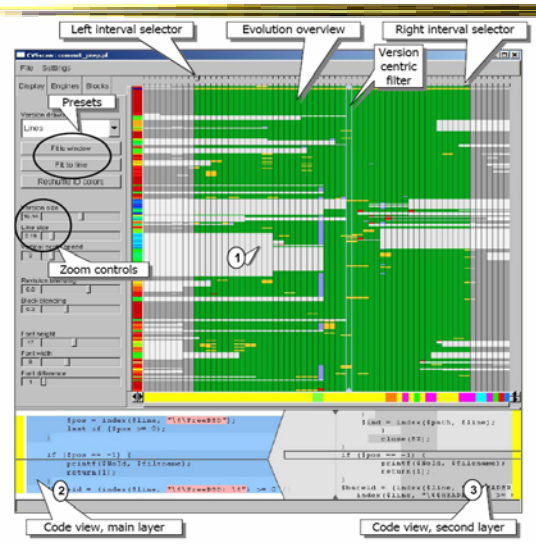


Figure 14: CVSscan tool overview. The file version and line number under the mouse (1) is shown in detail in the text views (2,3)

Conclusion

- User Studies
  - Content
  - History
- Future Work
  - Higher Level Views
  - Dependence on *diff* operator

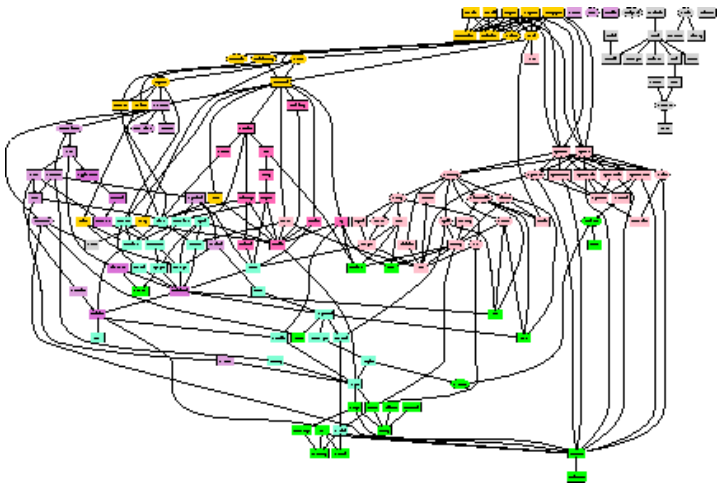
Stepping Up

- Next step in program visualization is to help debugging and performance optimization by visualizing program executions
  - Data, data structures, run-time heap, memory, control flow, data flow, hot spots, ...

Graph Visualization

- Area that we've already studied which is very important to SV
- Graphs pop up everywhere in software
- Common use: Visualize a call graph, visualize a flow chart, ...

Sample CallGraph View

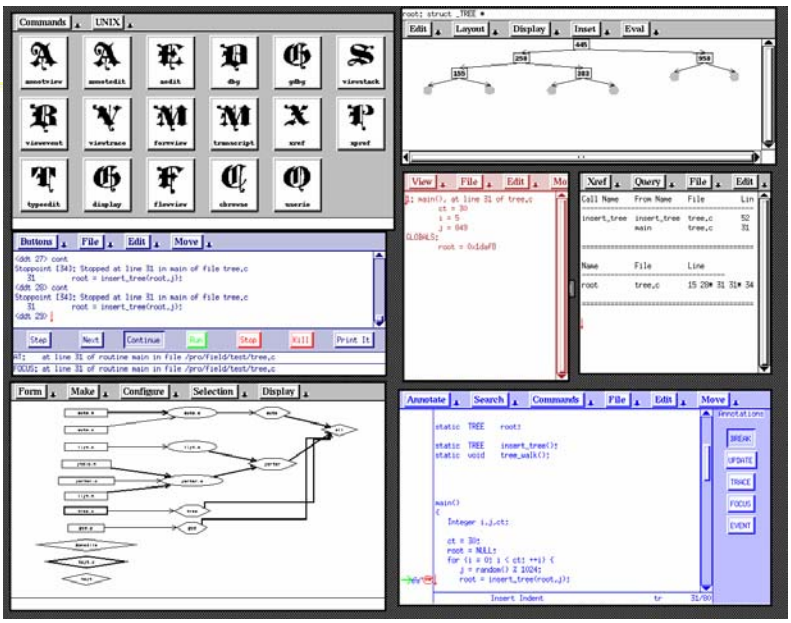


FIELD

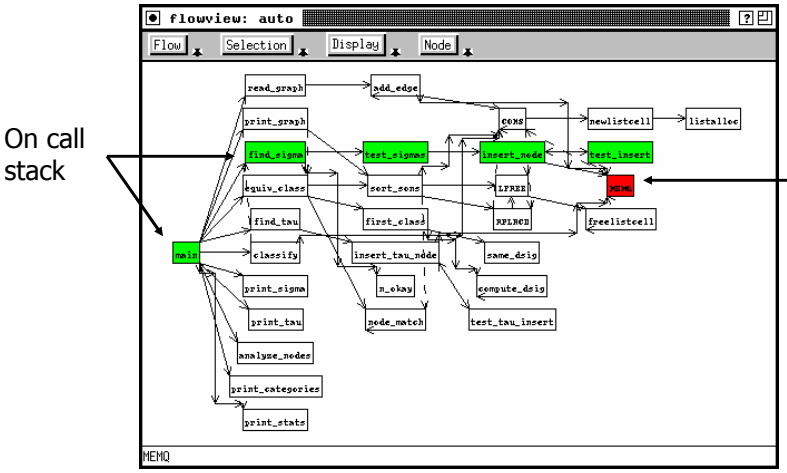
- Program development and analysis environment with a wide assortment of different program views
  - Integrated a variety of UNIX tools
  - Utilized central message server architecture in which tools communicated through message passing

Reiss
Software Pract & Exp '90

FIELD Interface



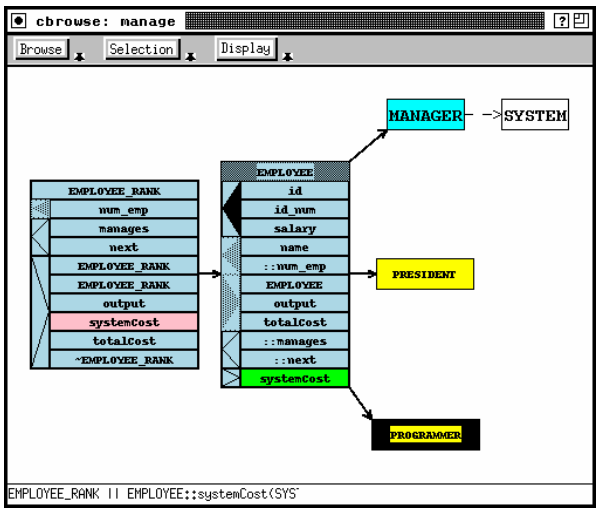
Dynamic Call Graph View



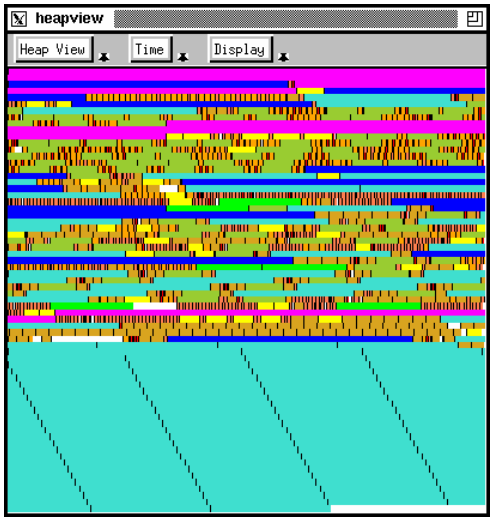
On call stack

Currently active

Class Browser

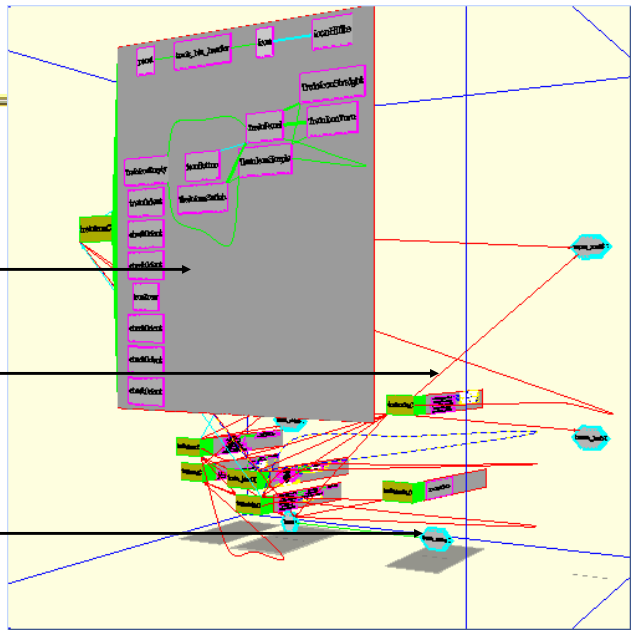


Heap View



Color could be
When allocated
Block size
Where allocated

3D Call Graph



Selected file

Groups of calls

Collapsed file

Paper Recap

"Strata-Various: Multi-Layer Visualization of Dynamics in Software System Behavior"
Kimelman, Rosenburg, Roth
Vis '94

Angela Navarro

Motivation

- No adequate software visualization tools
- Available tools are mostly static
- Dynamic tools present incomplete visualizations
- Significant increase of software performance

Purpose

- PV, Program Visualization prototype
- A mechanism for exploring applications
- Visualization of software behavior over time
- Comprehensive view of the system
- Navigation across gathered data

Goals

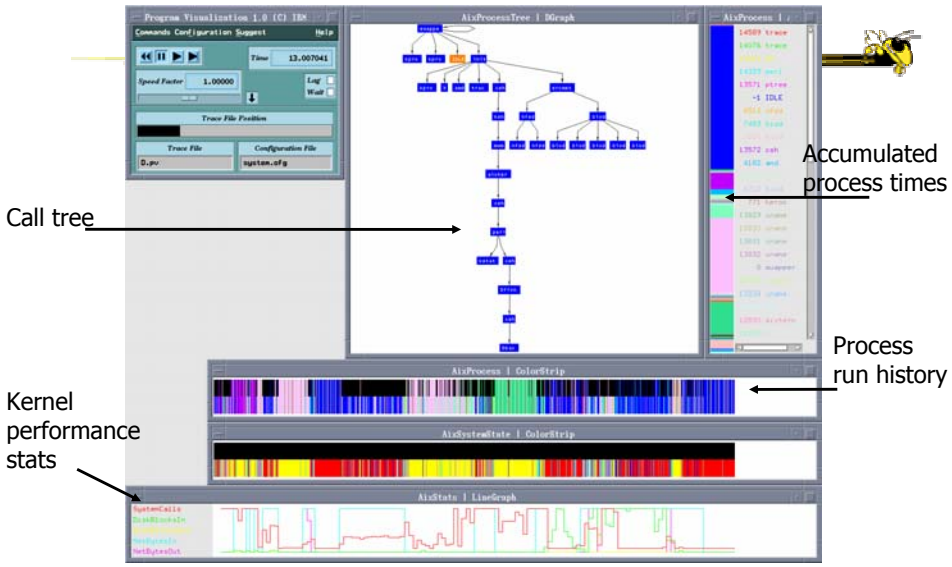
- Facilitate debugging by providing execution time information
- Faster assimilation of execution data
- Facilitate visual correlations, trends and anomalies
- Several layers including user-level libraries, OS and hardware

PV

- Trace-driven
- Trace: Time ordered sequence of events of interest
- Synchronous and asynchronous configurations
- Multiple concurrent views representing the different layers

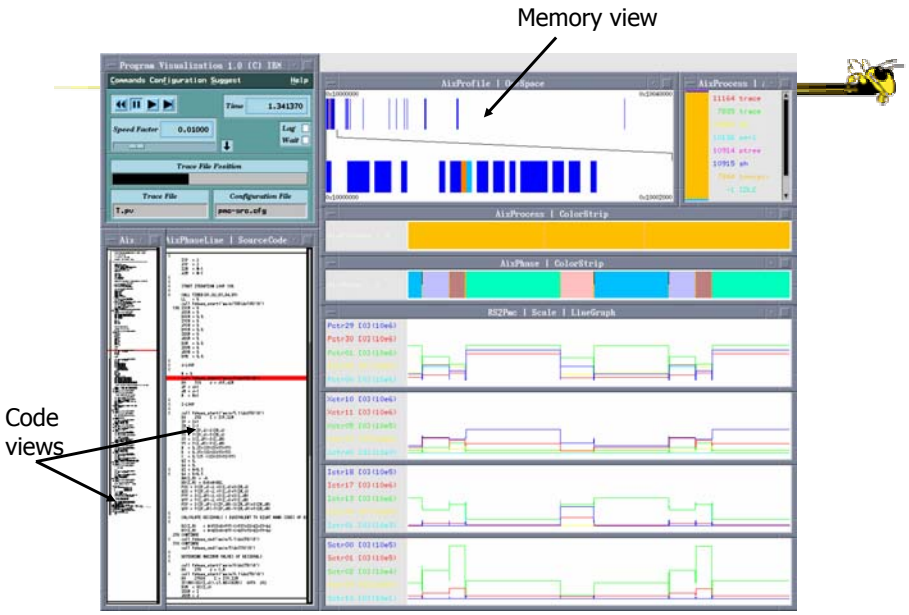
Views

- Process scheduling and System activity
  - Scheduling view
  - Activity view



Views

- Memory Activity and Application Progress
  - Memory size view
  - Source of memory allocation
  - State of the page of the data segment



Views

- Hardware Activity and Source Progress
  - Active loop view
  - Hardware performance statistics

Conclusion

- A more effective and comprehensive visualization of software
- Worthwhile for developers facing performance issues
- Room for improvement in visualization
- More direct traversal
- Presentation of derived data

GAMMATELLA

- Extending Tarantula to show executions on larger systems
- Discuss

Jones et al
Information Visualization '04

Visual Technique

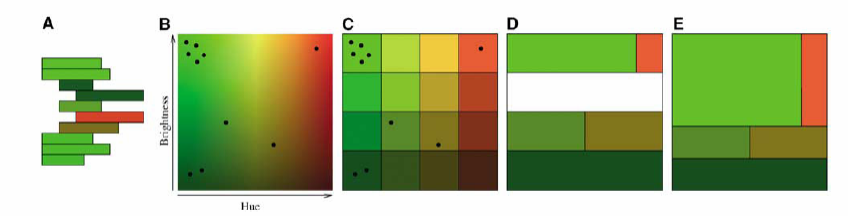


Figure 4 Example that illustrates the steps of the treemap node drawing.

- Segment region
- Use medians of colors
- Expand horizontally for proportion of lines in that segment
- Expand vertically for how much in that row

Full System View

Commercial Systems

- A number of commercial program development environments have begun to incorporate program visualization tools such as these
  - Majority are PC-based
- Has not become as wide-spread as I would have anticipated

Concurrent Programs

- Understanding parallel programs is even more difficult than serial
- Visualization and animation seem naturals for illustrating concurrency
- Temporal mapping of program execution to animation becomes critical

Kraemer & Stasko
JPDC '93

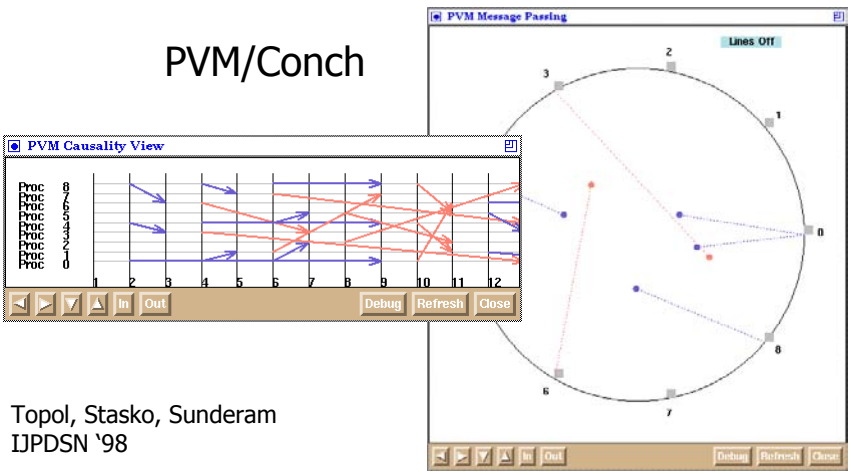
POLKA

- We developed a system, POLKA, that was designed to help people build visualizations of concurrent programs
  - Used for both program and algorithm visualization
  - Used for different programming models: message passing, shared memory threads, ...

Stasko & Kraemer
JPDC '93

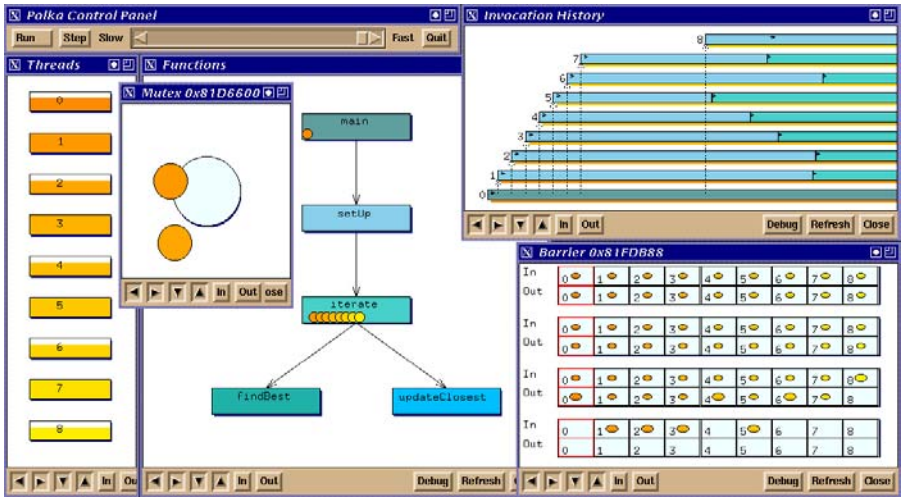
Message Passing Systems

PVM/Conch



Topol, Stasko, Sunderam
IJPDSN '98

Shared Memory Threads



Algorithm Visualization

- Learning about algorithms is one of the most difficult things for computer science students
  - Very abstract, complex, difficult to grasp
- Idea: Can we make the data and operations of algorithms more concrete to help people understand them?

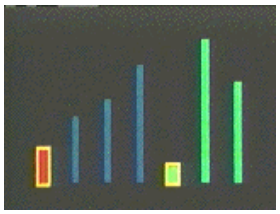
Algorithm Animation

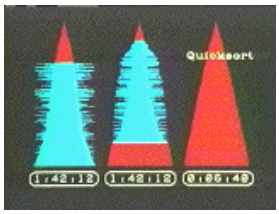
- Common name for area
- Dynamic visualizations of the operations and data of computer algorithm as it executes
  - Abstraction is key concept

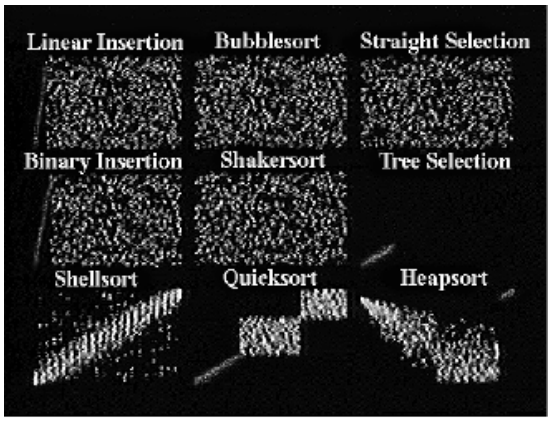
Sorting Out Sorting

- Seminal work in area
- 30 minute video produced by Ron Baecker at Toronto in 1981
- Illustrates and compares nine sorting algorithms as they run on different data sets

SoS Views







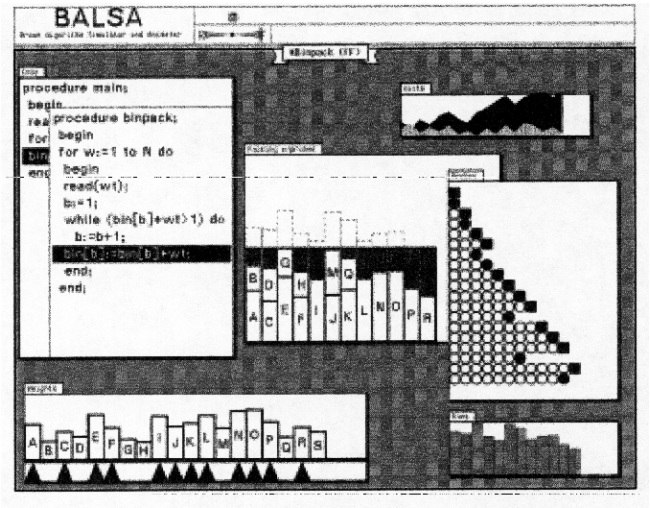
Video

Balsa

- First main system in area
- Used in "electronic classroom" at Brown
- Introduced use of multiple views and interesting event model

M. Brown
Computer `88

Example Animation



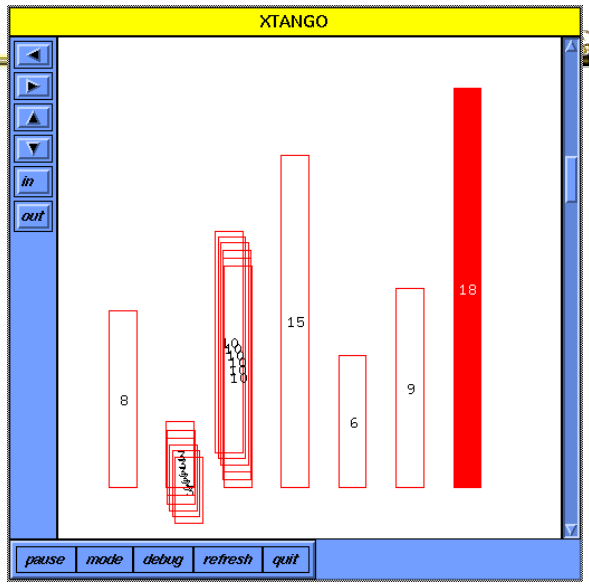
Tango

- Added smooth animation to algorithm animation
- Simplification of the design/programming process (easier programming model and framework)
- Formal model of the animation, the Path-Transition Paradigm

Stasko
Computer '90

Tango

Multiple frames from bubblesort

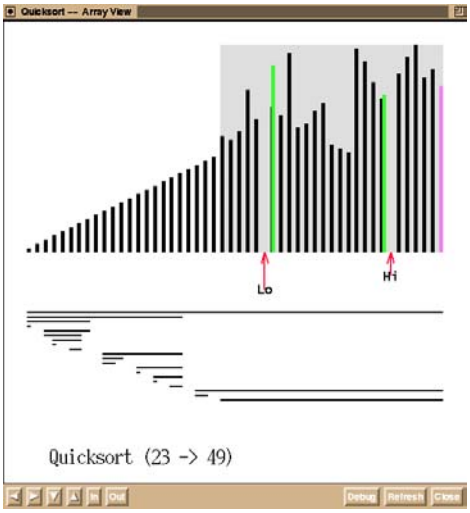


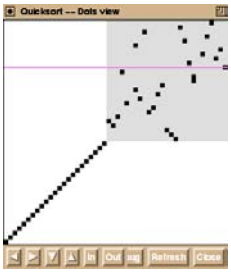
POLKA

- Improved animation design model
- Object-oriented paradigm
- Multiple animation windows
- Much richer visualization/animation capabilities

Stasko & Kraemer
JPDC '93

POLKA





Demo

Evaluating Effectiveness

- Do algorithm animations actually help students learn?
  - That's the claim, but unproven
- Conducted a number of empirical studies to answer question
  - Compare learning with animation to learning without
  - Measuring understanding is difficult

Results

- Somewhat mixed, but some studies do show benefits
  - -> Potential for pedagogical aid is there, but just can't throw animation at algorithm and expect it to help
  - Interaction is a key
    - Blindly watching algorithm animation not really helpful
    - Student must interact with animation and be engaged
  - Animations help motivation, can make algorithm less intimidating

Hundhausen, Douglas & Stasko
JVLC '02

Current State of SV

- Research continues…
- Some use of algorithm animations as pedagogical aids
- Program visualization trickling into commercial tools

What's Needed? (PV)

- Better analysis of what software developers want and need
- Flexible displays providing overview and detail
- Improved tracing/monitoring/analysis capabilities

What's Needed? (AA)

- Focus on interactive tools
- Simpler animation construction
- Empirical validation of value

HW 6

- Analytic scenario
- You come up with a hypothesis about what is being planned
- What kinds of visualizations might help?

Upcoming

- Vis for Information Security
  - Guest speaker: Greg Conti
  - Reading

    Conti et al
- Animation / Image-based InfoVis
  - Papers to discuss

References

- All referred to papers
- *Software Visualization*, MIT Press, 1998