

Software Visualization

Reflections and Future Directions

John Stasko

Information Interfaces Research Group
College of Computing and GVV Center
Georgia Tech

stasko@cc.gatech.edu

www.cc.gatech.edu/~john.stasko

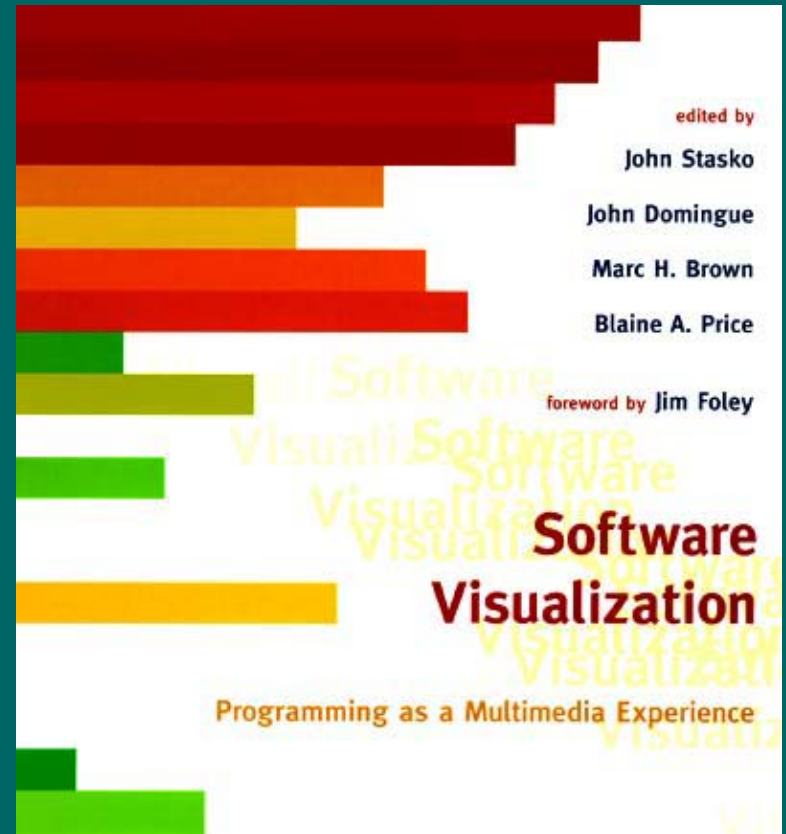


Software Visualization

- 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



Software Visualization Areas

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



Algorithm Animation

Program Visualization

89
90
91
92
93
94
95
96
97
98
99

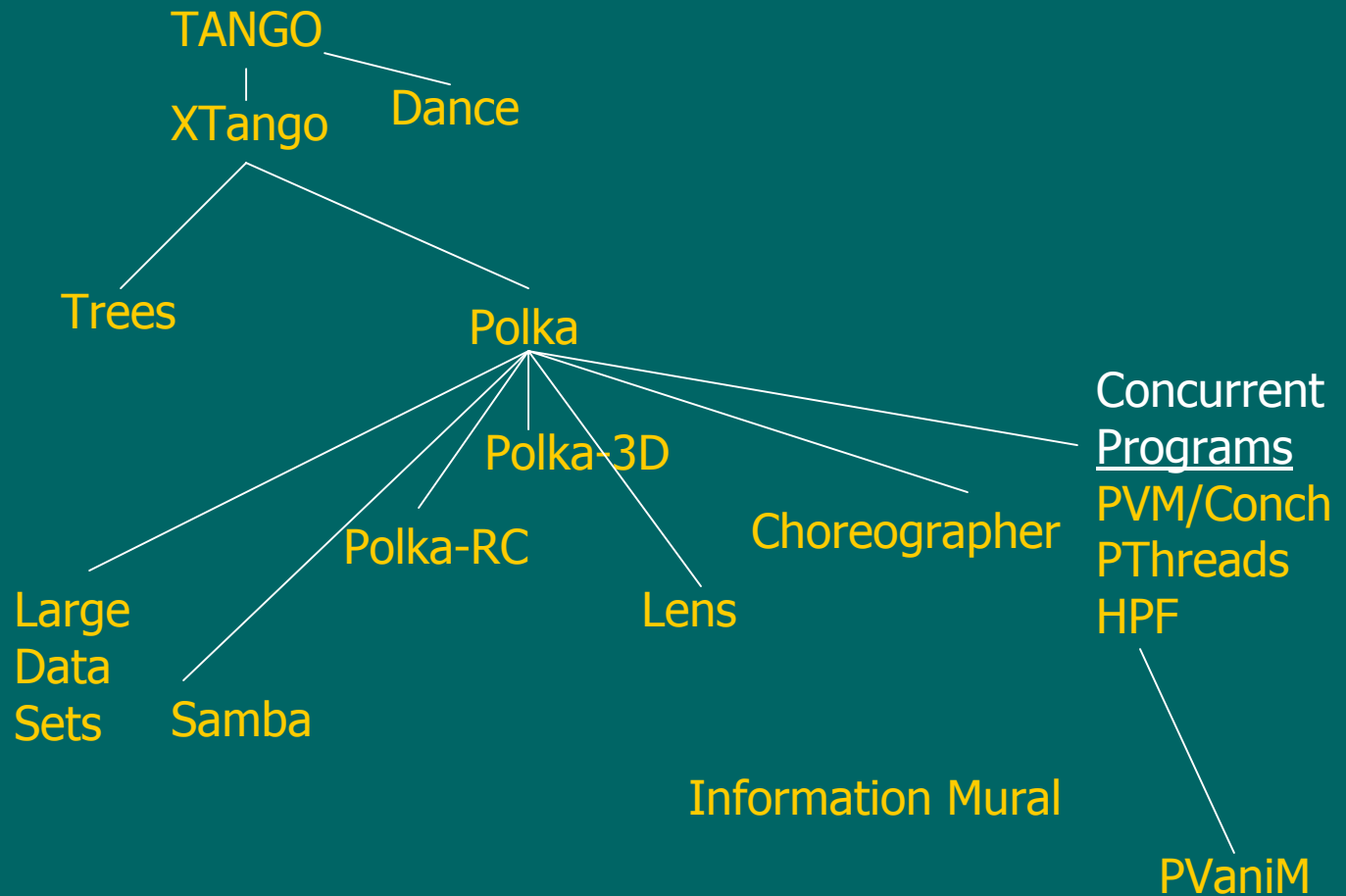
Evaluation

Pairheap

Intro Algos

BH & DFS

HW Scenario



John Stasko

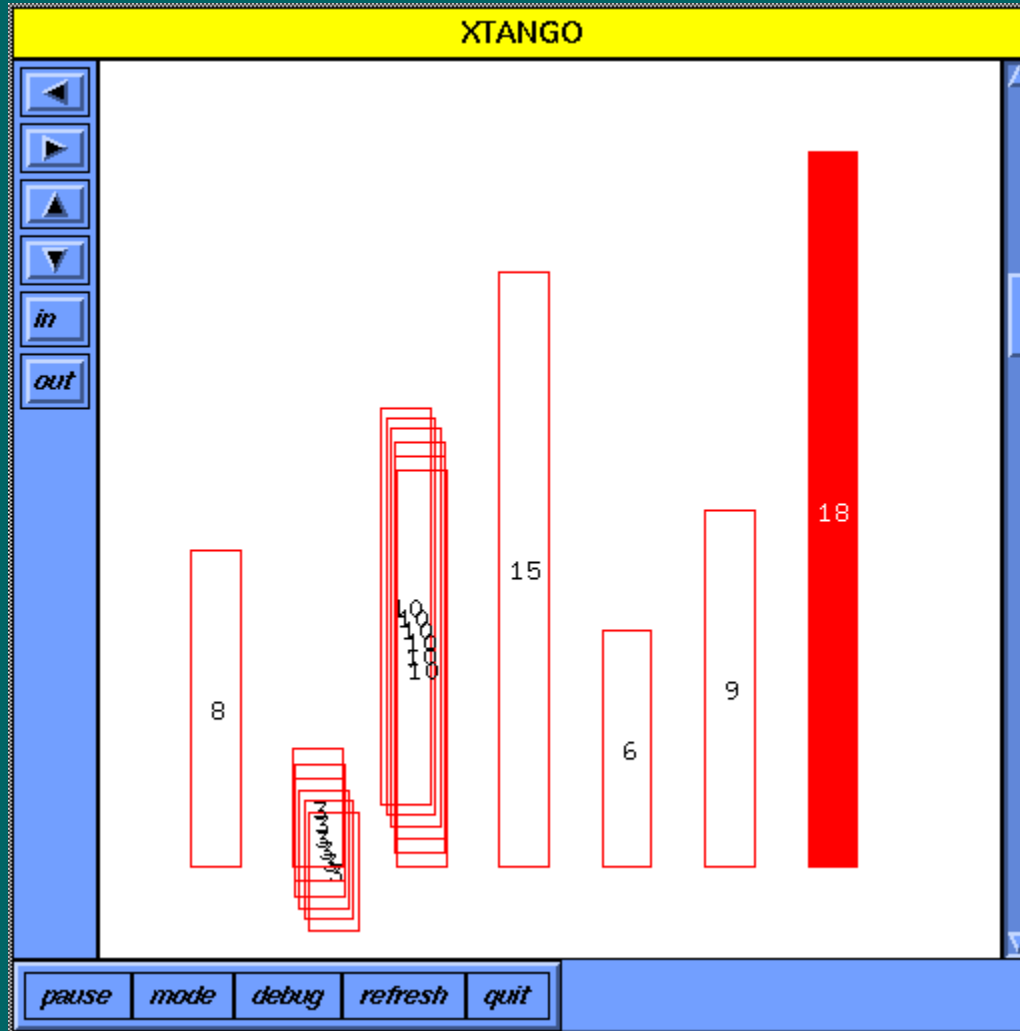
www.cc.gatech.edu/~john.stasko

stasko@cc.gatech.edu



Tango

Multiple frames from bubblesort



IEEE Computer '90
JVLC '90



Tango Model

- Image, Location, Path, Transition

```
Location    fromloc, toloc;  
Image       ball;  
Path        path1, path2;  
Transition  mover;
```

Sample
code

```
ball = AssocRetrieve("ID", paramvalue);  
fromloc = ImageLoc(ball, Center);  
toloc = LocCreate(0.4, 0.7);  
path1 = PathMakeType(Clockwise);  
path2 = PathExample(fromloc, toloc, path1);  
mover = TransCreate(Move, ball, path2);  
TransPerform(mover);
```



Tango Contributions

- Importance of smooth animation
- Simplification of the design/programming process
- Formal model of the animation, the Path-Transition Paradigm



XTango

- Native X Windows version of Tango



D. Hayes

SIGACT News '92



DANCE

- Why not design an algorithm animation visually?
- Direct manipulation environment for interactive, visual design of algorithm animations
=> Automatically generates Tango code

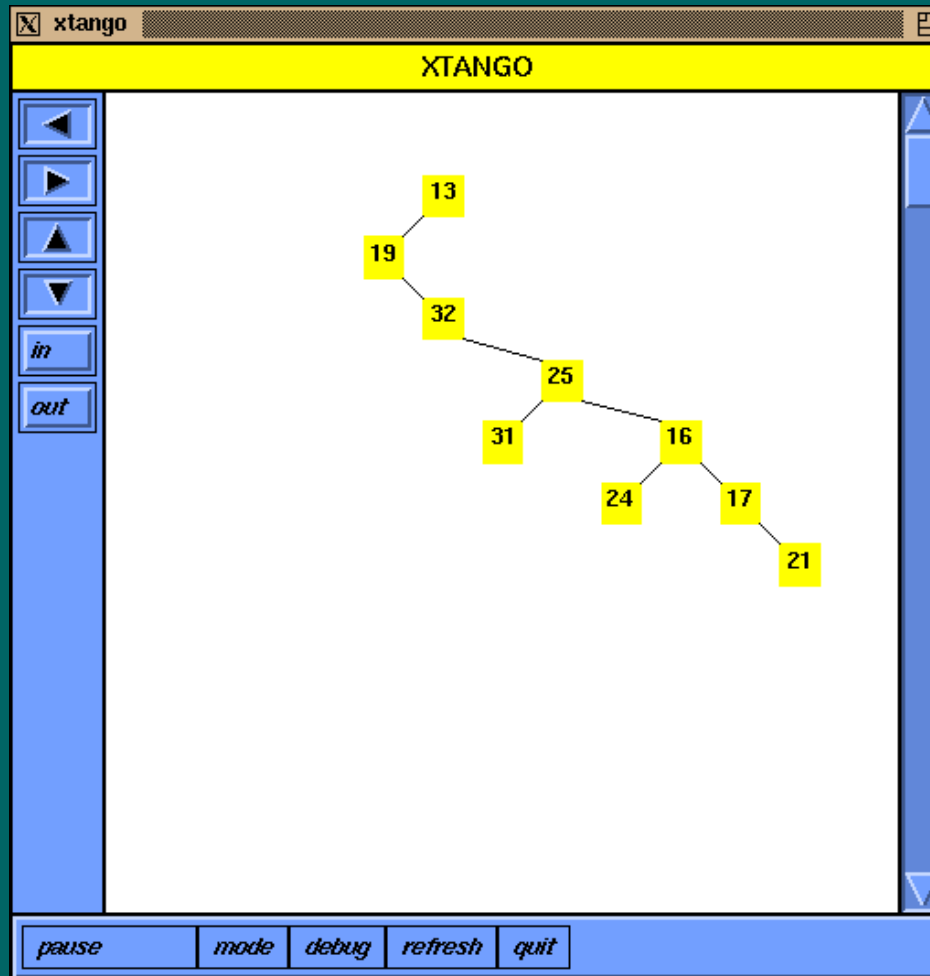


DANCE

The screenshot shows a window titled "dance" with a menu bar containing "Commands", "Scene", "Image", "Loc", "Path", and "Trans". The main workspace contains several objects: a blue line labeled "line1", a path of white circles labeled "path1", a red rectangle labeled "rect", a green circle labeled "circ" with a label "swloc" inside it, a label "loc1" with a small black dot below it, a path of white circles labeled "path2", and the text "some sample text" with a label "tvar" below it. A "Transitions:" panel on the right lists: "raiser", "colorer", "resizer", "composer", and "iterator", each with a small icon. At the bottom, it says "Defining scene: ExampleScene(i p1 i p2)".



Animating Tree Algorithms



Binary representation
of a pairing heap
data structure

C. Turner

VL '92



Concurrent Programs

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



Tango Insufficiencies

- Simulated object-oriented
- One animation window
- Transition model



Must compose all transitions into one
"super" transition, then perform it



POLKA

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

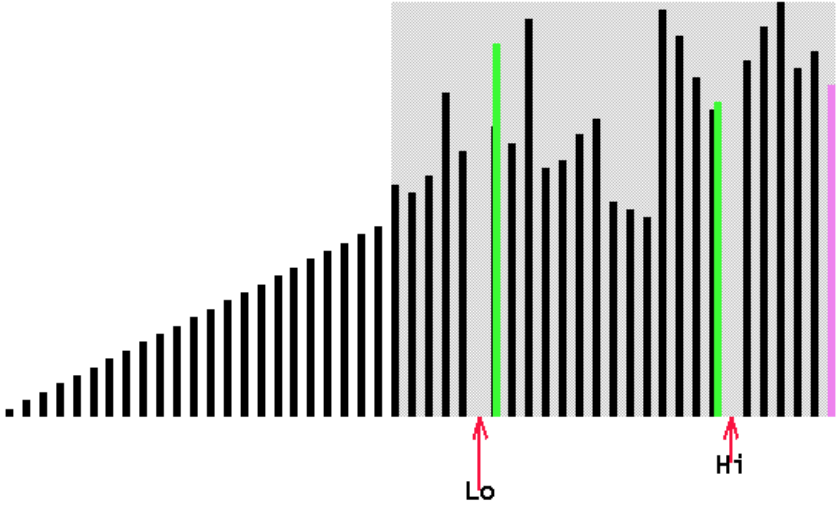


POLKA

Polka Control Panel

Run Step Slow Fast Quit

Quicksort -- Array View



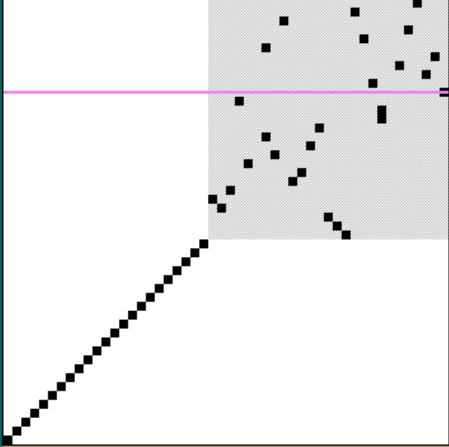
Lo Hi

Quicksort (23 -> 49)

Debug Refresh Close

The Array View window displays a bar chart representing an array of 49 elements. The bars are sorted in ascending order of height. Two vertical green bars are highlighted, labeled 'Lo' and 'Hi' with red arrows pointing to their positions. A horizontal purple line is drawn across the chart, indicating a pivot value. Below the chart, a stack of horizontal lines represents the recursive call stack, showing the current call at the bottom and several recursive calls above it.

Quicksort -- Dots view



In Out Bug Refresh Close

The Dots view window shows the same array as the Array View, but represented as a grid of black dots. A horizontal purple line is drawn across the grid, indicating the pivot value. The dots are arranged in a pattern that shows the relative positions of the elements in the array.



POLKA Model

- Location, AnimObject, Action
- Introduce explicit animation time (frame)



POLKA Programming Model

```
Circle *circ;  
Loc *loc, *center;  
Action *act;  
int len;  
  
circ = new Circle(this,1, 0.2,0.3, 0.1,"red", 1.0);  
circ->Originate(time);  
  
center = circ->Where(PART_C);  
loc = new Loc(0.6, 0.5);  
  
act = new Action("MOVE", center, loc, 20);  
len = circ->Program(time, act);  
time = Animate(time, len);
```



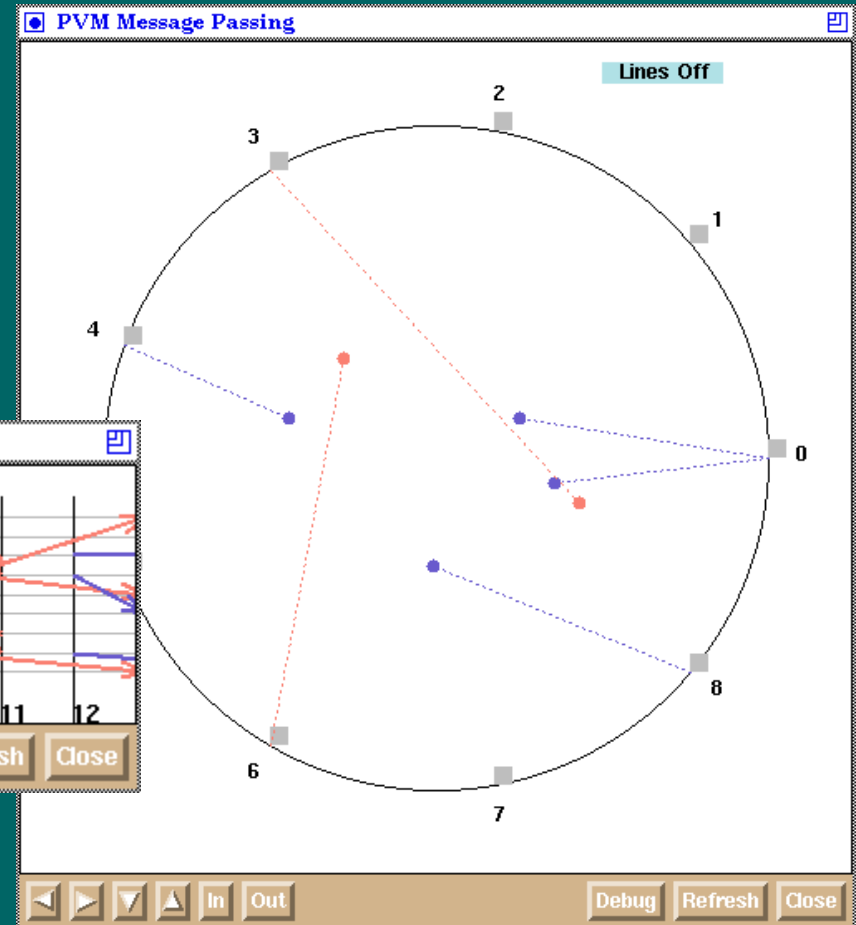
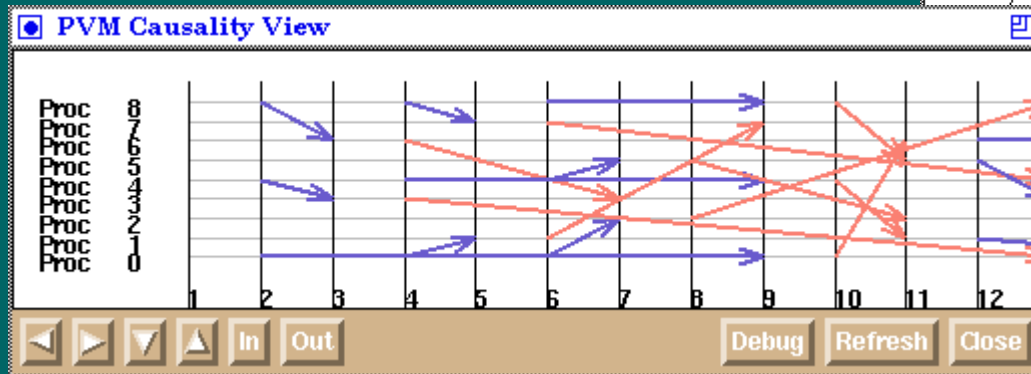
Concurrent Programs

- New model better, more flexible for illustrating concurrent program actions
- Polka used to build animation libraries for a variety of architectures/programming paradigms
 - message passing
 - shared memory
 - compiler-driven parallelism



Message Passing

PVM/Conch



B. Topol
V. Sunderam

ICDCS '95
IJPDSN '98



Threads

Pthreads

The screenshot displays the Polka Control Panel interface, which is used for debugging and monitoring parallel programs. The main window is divided into several panels:

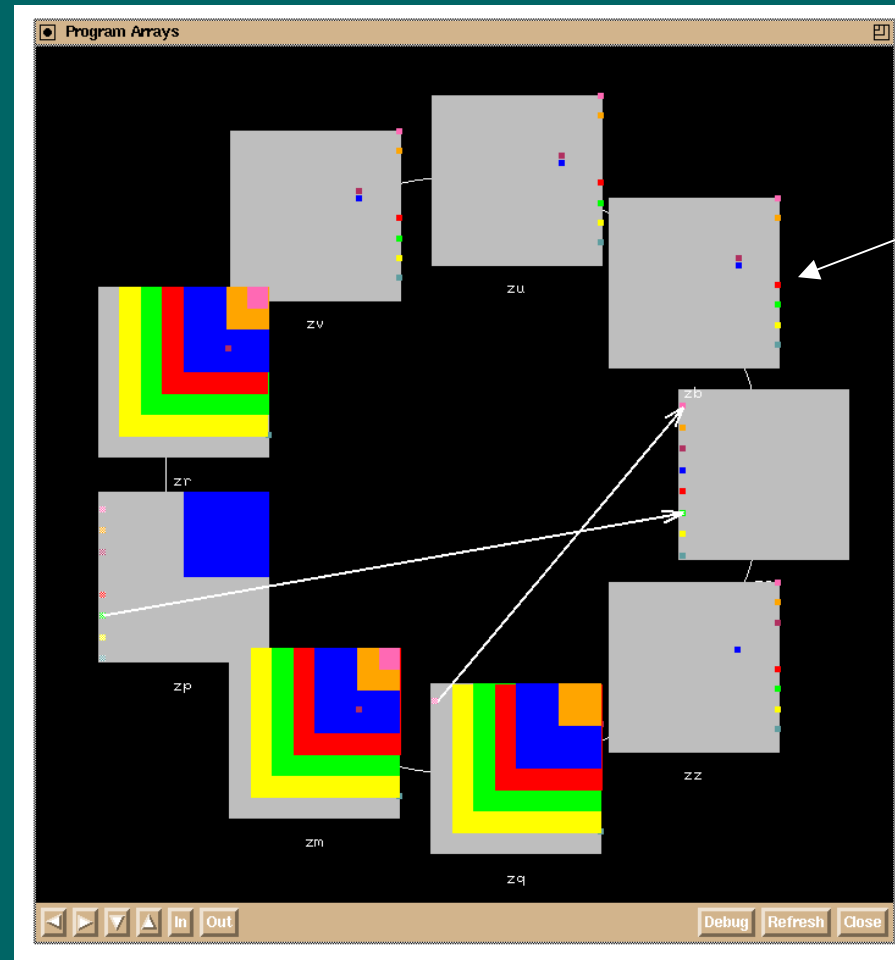
- Polka Control Panel:** Contains control buttons for **Run**, **Step**, **Slow**, **Fast**, and **Quit**.
- Threads:** A list of 9 threads (0-8) shown as horizontal bars. Thread 0 is highlighted in orange, while others are yellow.
- Functions:** A call graph showing the execution flow: **main** calls **setUp**, which calls **iterate**. **iterate** then branches into **findBest** and **updateClosest**.
- Mutex 0x81D6600:** A window showing a diagram of two overlapping circles, representing a mutex lock held by thread 0.
- Invocation History:** A Gantt chart showing the execution timeline of all 9 threads. Thread 0 starts at time 0 and runs for a long duration. Other threads start later and run for shorter durations.
- Barrier 0x81FDB88:** A table showing the state of a barrier. The table has 4 rows (In/Out for threads 0-8) and 9 columns (threads 0-8). The first two rows show all threads at the barrier. The third row shows thread 8 has passed. The fourth row shows all threads have passed.

A. Zhao

TR



High Performance FORTRAN



Arrays in program

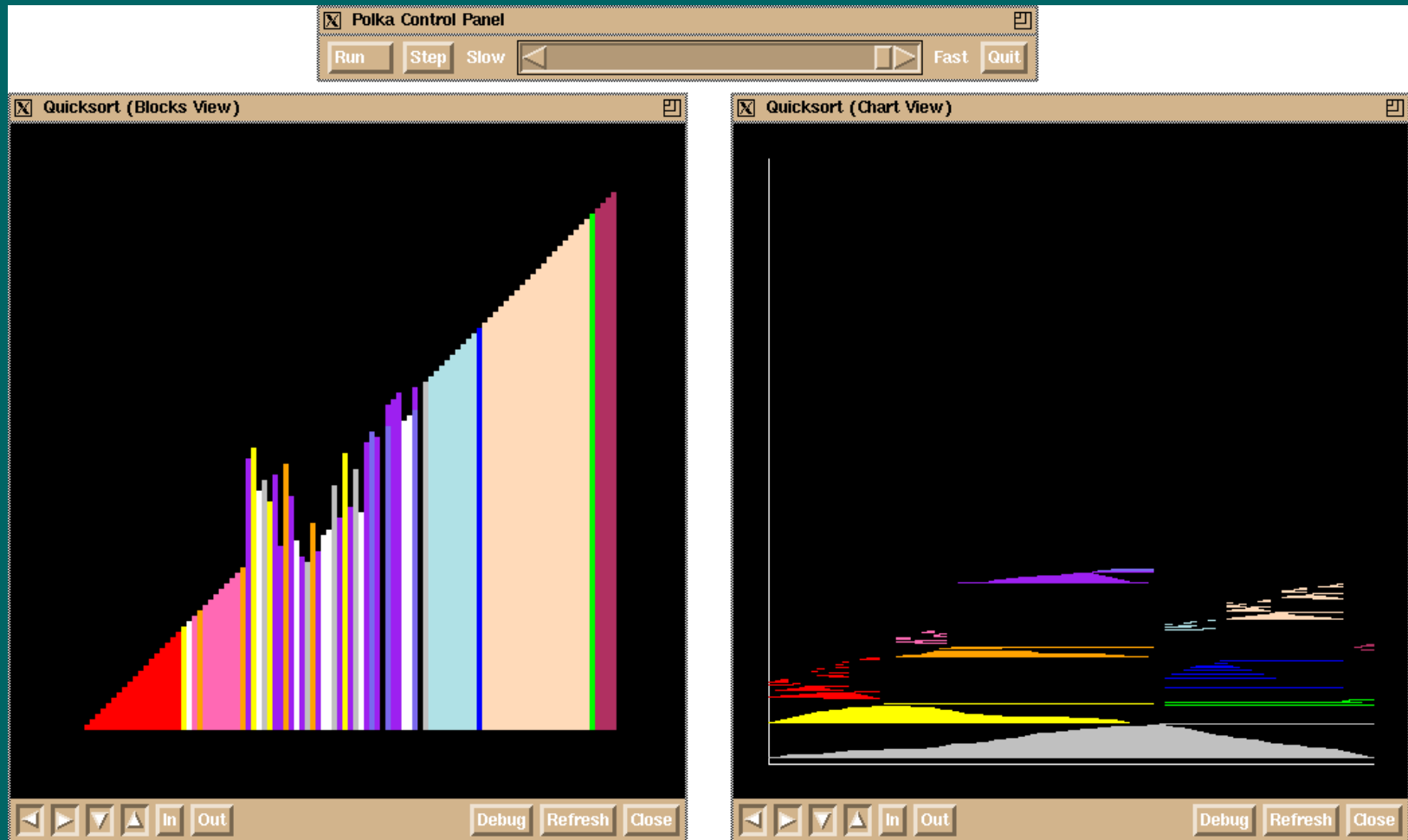
Color represents process accessing that memory

W. Appelbe



Example Program Illustration

Parallel
Quicksort



Array values view

History of exchanges view



Temporal Mapping

- Many temporal mappings exist from a concurrent program's execution to its animation
 - timestamp
 - serialized
 - maximum concurrency



Animation Choreographer

- Visual depiction of program events and dependencies
- Allows viewer to manipulate events in time, then see animation that reflects that temporal order

E. Kraemer

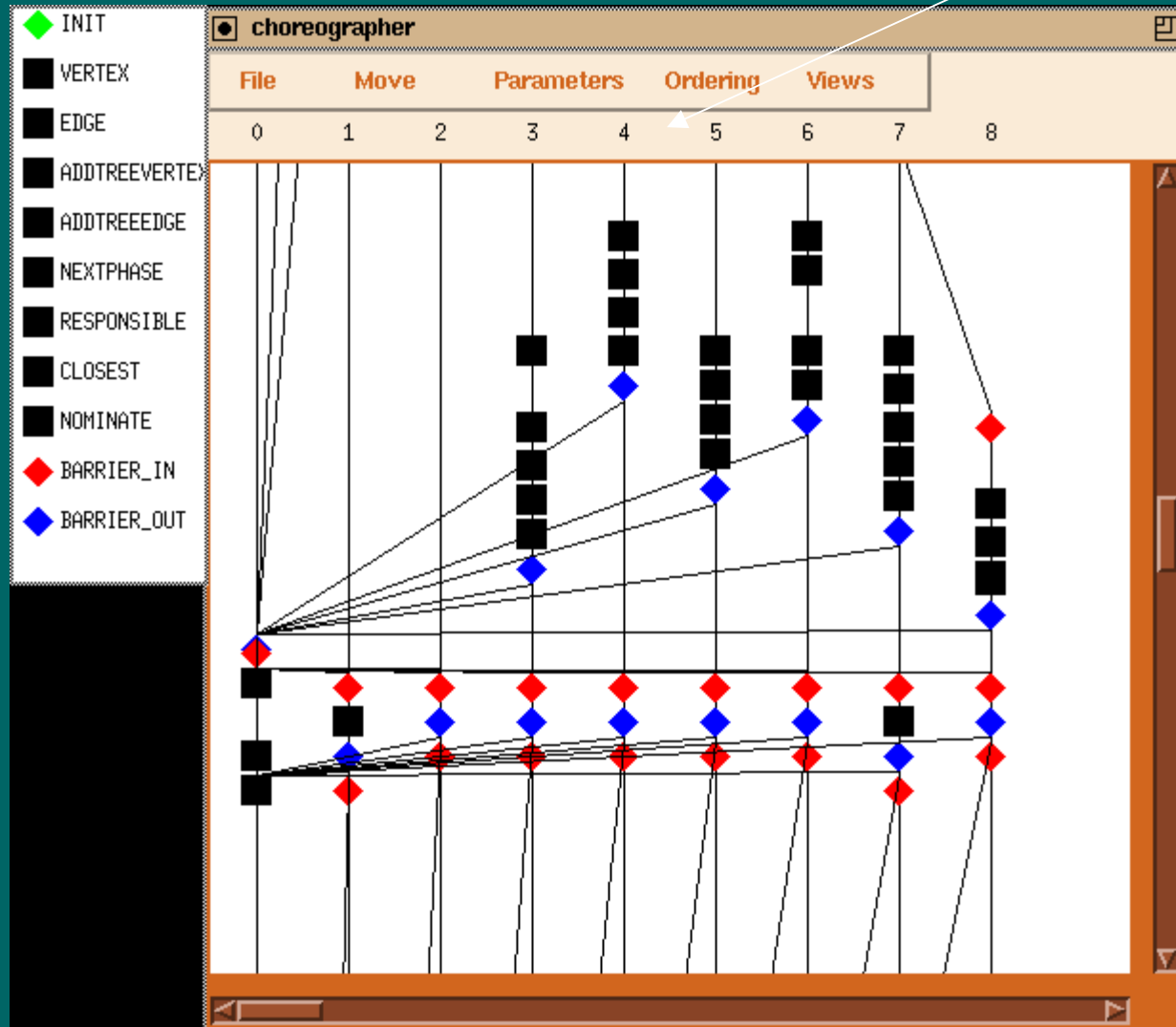
IPPS '94
WPC '94
Concurrency '98



Animation Choreographer

processes

Event types



time



PVaniM

- Visualizing PVM programs on-line
- Must use sampling, not tracing due to sheer number of events
- Shows machine loads, host utilization, memory used, messages sent, communication patterns, etc.

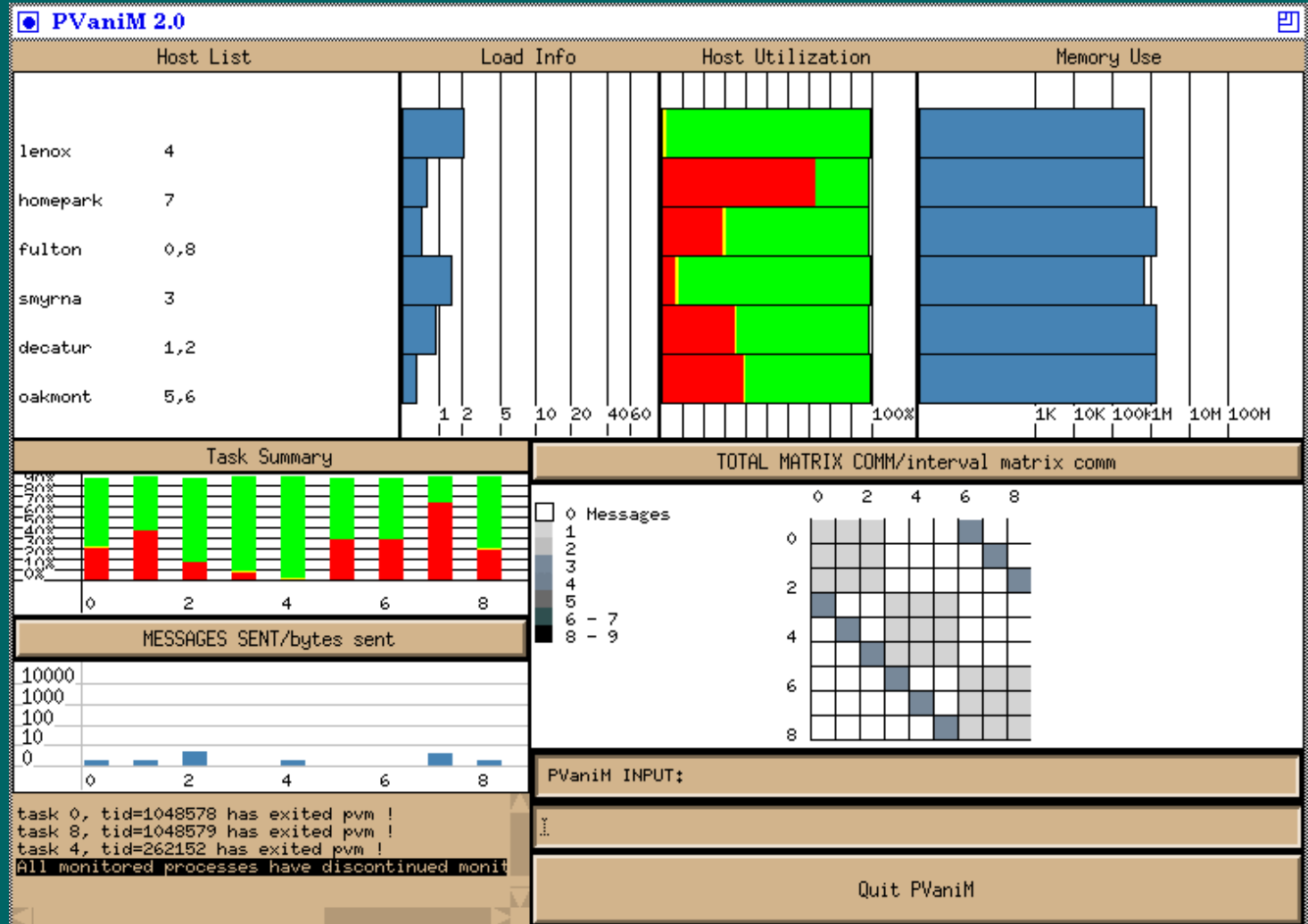
B. Topol
V. Sunderam

Concurrency: P & E '98



PVaniM

Basic
system
UI



Polka-3D

- 3-D and VR version of Polka
- Same animation model
- Use third dimension to
 - Enhance visual aesthetics
 - Portray 3-D data
 - Encode more program attributes
- Not sure appropriate for algo anim

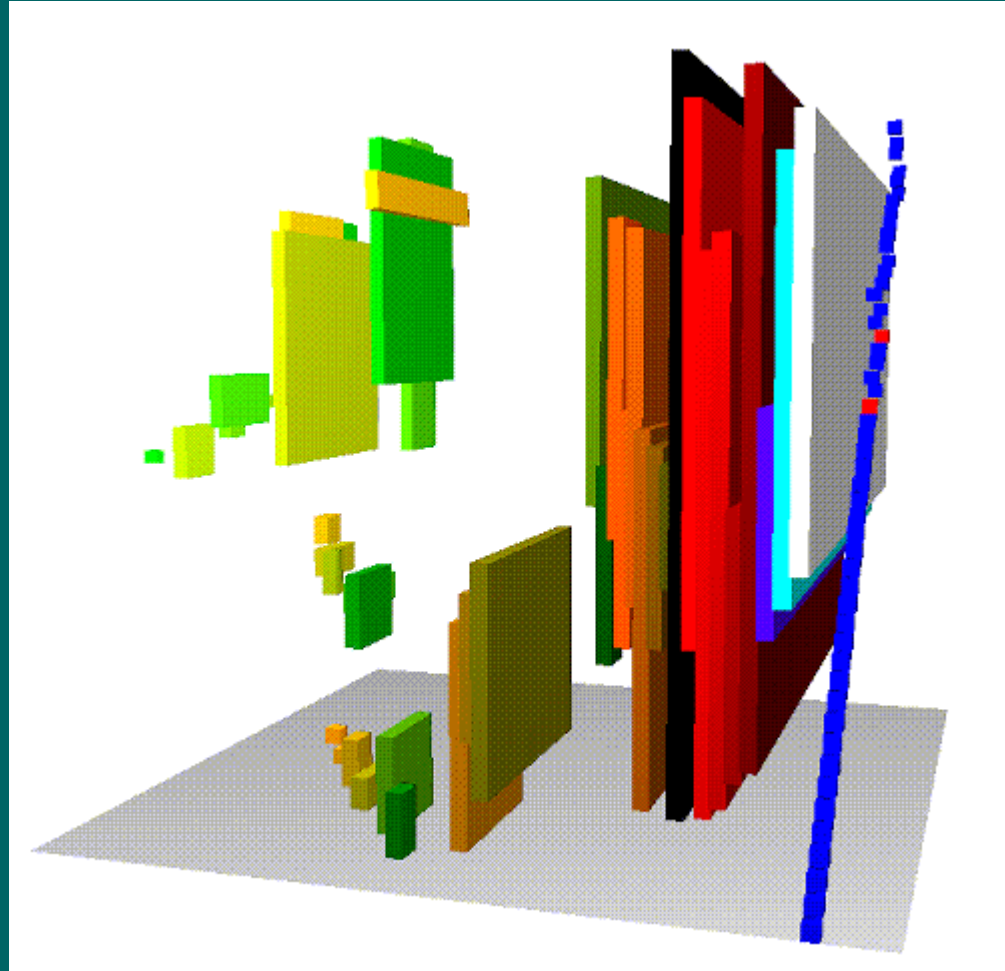


POLKA-3D

Quicksort side view

Blue dots are as in
2-d view

Colored planes
represent exchanges



Polka-RC

- Rather than time being animation frames, what if we use elapsed wall clock time?
 - Challenging under X Windows
 - Not clear if this is desirable for algorithm animations



Polka-RC

Programming
model

```
Action a1("RESIZE", rect1,  
          Traj(CLOCKWISE,0.2,-0.1,slowinout),  
          START_AT, Now(),  
          DURATION, Sec(1.5));
```

```
Action a2("MOVE", rect1,  
          Traj(STRAIGHT,loc1,loc2,uniform),  
          START_AFTER_END_OF, &a1, Sec(0.5),  
          VELOCITY,50)
```

```
Action mov1("MOVE", elt[i],  
            Traj(CLOCKWISE,from,to,uniform),  
            START_AT, ASAP(), VELOCITY, 50);
```

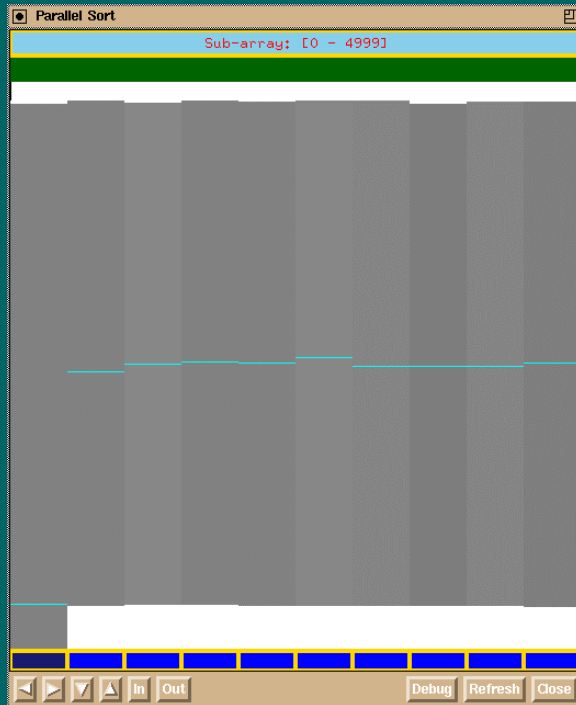
```
Schedule(&a1);  
Schedule(&a2);  
Schedule(&mov1);
```



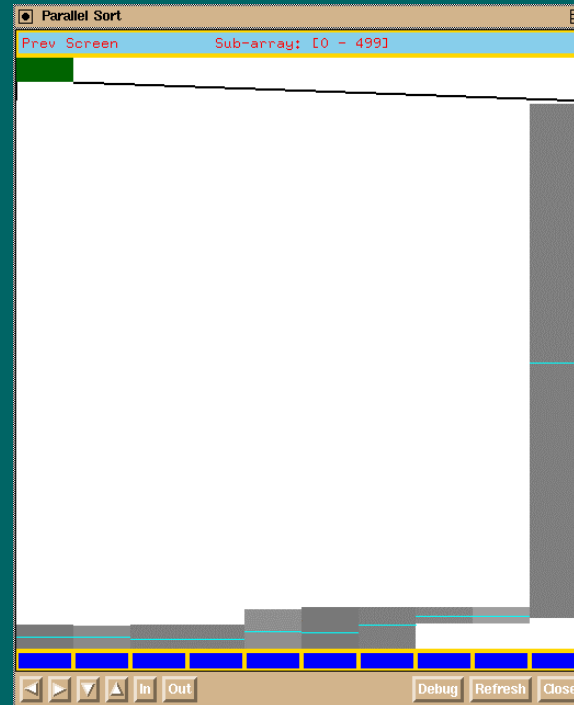
Visualizing Large Data Sets

Uses semantic zooming

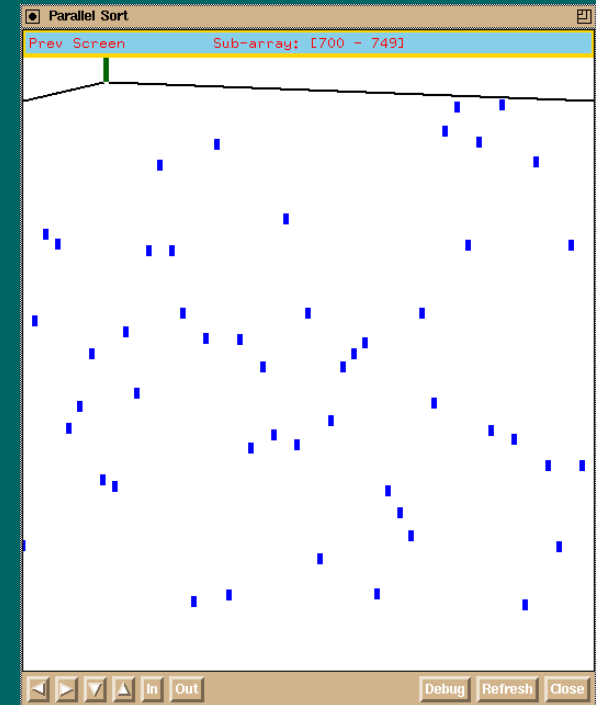
Sorting 5000 elements



View all data



Zoom to first 10%



Zoom further



Visual Debugging

- Can we adapt algorithm animation capabilities to help programmers debug their code?
 - Want to go beyond data structure displays
 - Show semantics of program's domain
 - Easy specification by programmer



Lens

Animation design palette

Source code

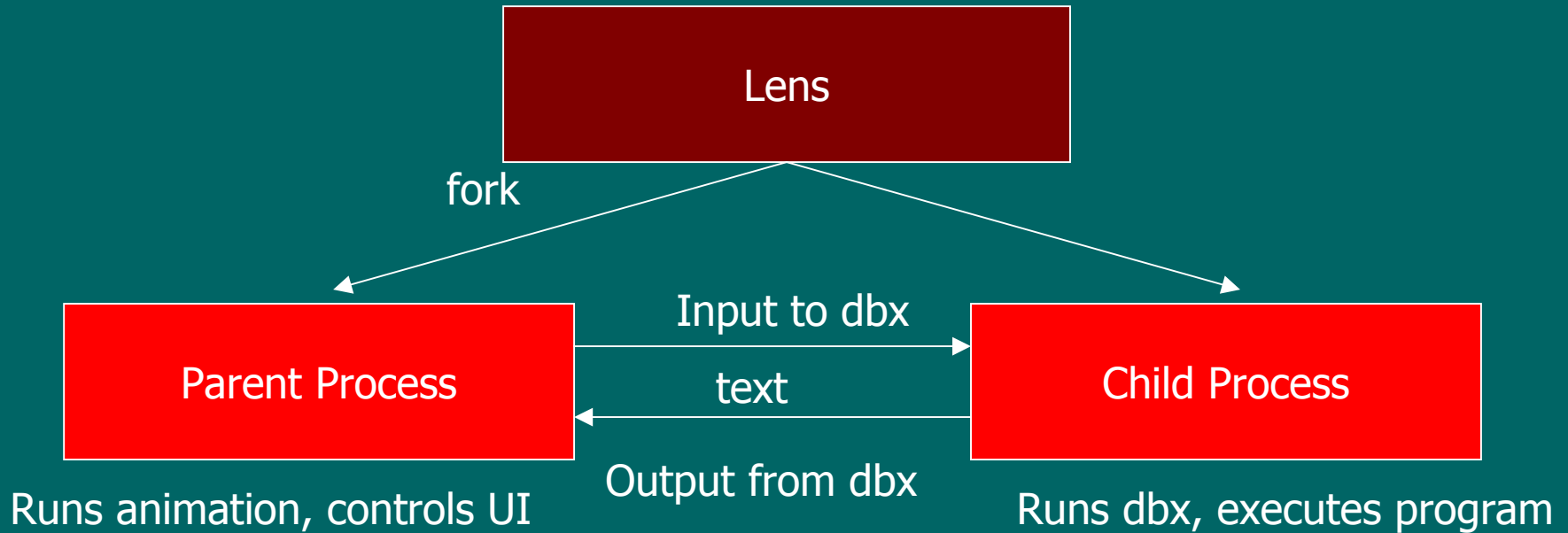
```
1 #include <stdio.h>
2
3 main()
4 {
5     int n,i,j;
6     int temp;
7     int a[50];
8
9     int count;
10
11     printf("Input number of elts in array\n");
12     scanf("%d",&n);
13
14     printf("Enter the elements\n");
15     for (count=0; count<n; ++count)
16         scanf("%d",&a[count]);
17
18     for (j=n-2; j>=0; --j)
19         for (i=0; i<j; ++i)
20             if (a[i] > a[i+1])
21                 { temp = a[i];
22                   a[i] = a[i+1];
23                   a[i+1] = temp;
24                 }
25 }
```

<lens> alias p print
<lens> alias q quit
<lens> alias r run
<lens> alias s step
<lens> alias w where
<lens> use /net/ag13/projects/polka
<lens> file /hg9/stasko/Sun/lens/bsort.c
dbx: no such source file: "/hg9/stasko/Sun/lens/bsort.c"
<lens>



Lens System

- Architectural model



S. Mukherjea

ICSE '93
ToCHI '94



Empirical Evaluation of Algorithm Animations as Learning Aids

- Can we show that algorithm animations can help students learn?
- Compare learning with animation to learning without
- Measuring understanding is difficult
- Four main studies



1. Pairing Heaps

- Classical experimental design
- Just having animation doesn't make learning happen
- Difficult for student to leverage animation of complex algorithm when they don't understand algorithm and visual mapping yet

A. Badre
C. Lewis

InterCHI '93



2. Introductory Algorithms

- Sorting, graphs
- Interaction is the key
 - Students who enter their own data sets into the algorithm benefited from animation

A. Lawrence
A. Badre

VL '94



3. Binomial Heap & DFS

- Is animation like prediction?
- Does it help the student to anticipate what will happen next and learn from that?
- Mixed data, some support

M. Byrne
R. Catrambone

Computers & Ed '99



4. Homework Scenario

- Provide student with learning objectives/questions up front
- Give unlimited work time
- Animations appear to help motivation
 - Make a complex algorithm less intimidating
 - Animation helped learning

C. Kehoe
A. Taylor

IJHCS '00



Student-Built Animations

- Samba
 - Simple animation scripting language

```
circle 1 0.8 0.8 0.1 red half
line 2 0.1 0.1 0.2 0.2 green thin
rectangle 3 0.1 0.9 0.1 0.1 blue solid
text 4 0.0 0.0 0 black Hello
circle 6 0.3 0.3 0.2 wheat solid
triangle 7 0.5 1.0 0.6 0.8 0.4 0.9 cyan solid
bigtext 8 0.2 0.2 0 black Some Big Text
moveto 1 6
moverelative 3 0.05 -0.4
jumprelative 4 0.4 0.4
lower 1
color 6 blue
```

- Embed print statements in any program to generate

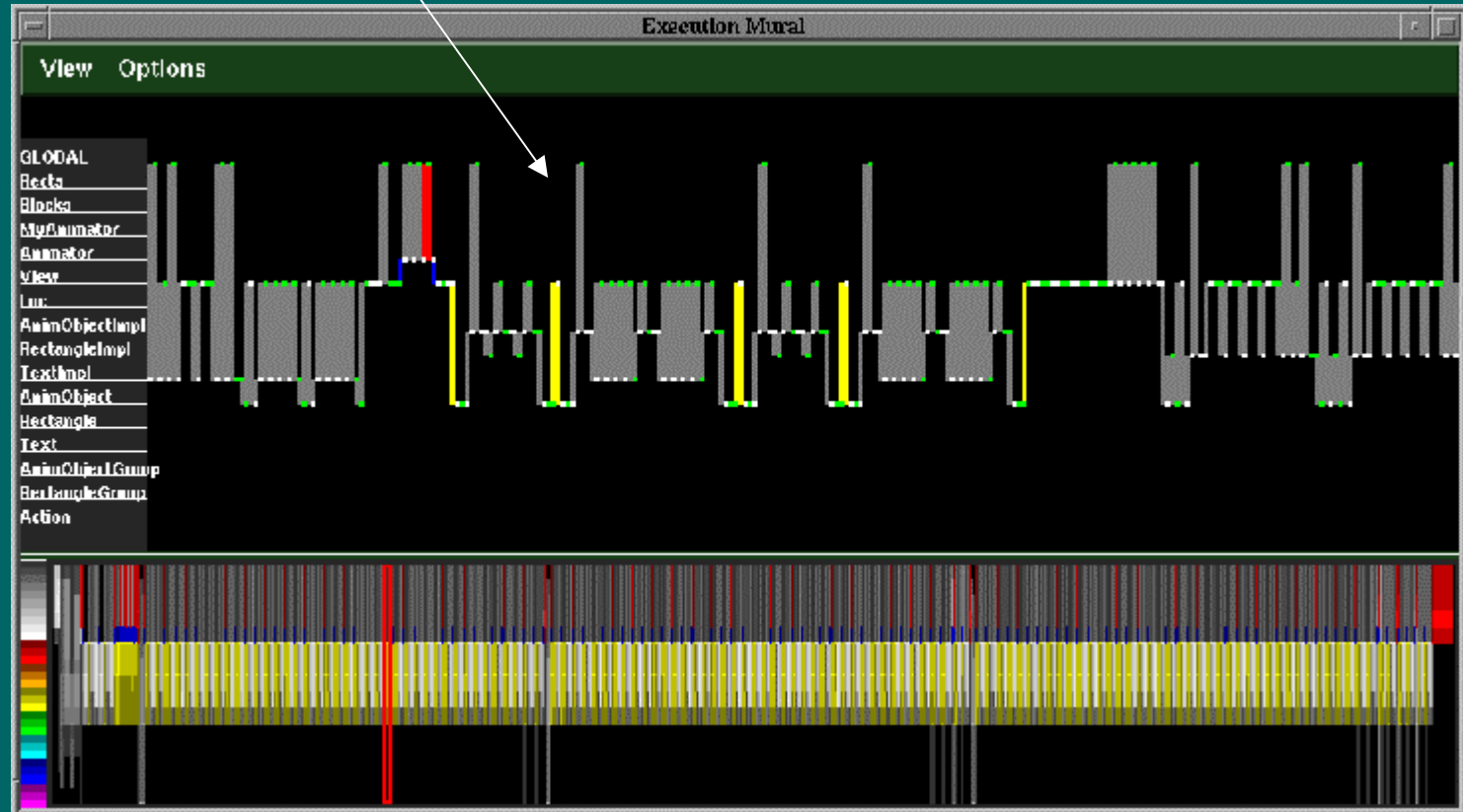
SIGCSE '97



Visualizing Large O-O Programs

Information Mural

messages



detail

overview

time →

D. Jerding

Dec. 1999



InfoVis '95

ICSE '97

ToVCG '98

42

Current State of Software Visualization

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



What's Needed? (AA)

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



What's Needed? (PV)

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



Acknowledgments

- Research supported by National Science Foundation, Sun, and the Gvu Center
- More info
 - www.cc.gatech.edu/gvu/softviz
 - stasko@cc.gatech.edu

