Information Visualization with Accordion Drawing

Tamara Munzner
University of British Columbia
Accordion Drawing

• rubber-sheet navigation
  – stretch out part of surface, the rest squishes
  – borders nailed down
  – Focus+Context technique
    • integrated overview, details
    – old idea
      • [Sarkar et al 93], ...
  • guaranteed visibility
    – marks always visible
    – important for scalability
    – new idea
      • [Munzner et al 03]
Guaranteed Visibility

• marks are always visible
• easy with small datasets
Guaranteed Visibility Challenges

• hard with larger datasets
• reasons a mark could be invisible
  – outside the window
    • AD solution: constrained navigation
  – underneath other marks
    • AD solution: avoid 3D
  – smaller than a pixel
    • AD solution: smart culling
Guaranteed Visibility: Small Items

- naive culling may not draw all marked items
Outline

• trees
  – TreeJuxtaposer
• sequences
  – SequenceJuxtaposer
• scaling up trees
  – TJC
• general AD framework
  – PRISAD
• power sets
  – PowerSetViewer
• evaluation
Phylogenetic/Evolutionary Tree

Common Dataset Size Today

Future Goal: 10M Node Tree of Life

Paper Comparison: Multiple Trees

focus

context
TreeJuxtaposer

- comparison of evolutionary trees
  - side by side
- demo
  - olduvai.sf.net/tj
TJ Contributions

• first interactive tree comparison system
  – automatic structural difference computation
  – guaranteed visibility of marked areas
• scalable to large datasets
  – 250,000 to 500,000 total nodes
  – all preprocessing subquadratic
  – all realtime rendering sublinear
• introduced accordion drawing (AD)
• introduced guaranteed visibility (GV)
Joint Work: TJ Credits

Tamara Munzner, Francois Guimbretiere, Serdar Tasiran, Li Zhang, and Yunhong Zhou.
TreeJuxtaposer: Scalable Tree Comparison using Focus+Context with Guaranteed Visibility.
SIGGRAPH 2003
www.cs.ubc.ca/~tmm/papers/tj

James Slack, Tamara Munzner, and Francois Guimbretiere.
TreeJuxtaposer: InfoVis03 Contest Entry. (Overall Winner)
InfoVis 2003 Contest
www.cs.ubc.ca/~tmm/papers/contest03
Outline

• trees
  – TreeJuxtaposer

• sequences
  – SequenceJuxtaposer

• scaling up trees
  – TJC

• general AD framework
  – PRISAD

• power sets
  – PowerSetViewer

• evaluation
Genomic Sequences

• multiple aligned sequences of DNA
• now commonly browsed with web apps
  – zoom and pan with abrupt jumps
• check benefits of accordion drawing
  – smooth transitions between states
  – guaranteed visibility for globally visible landmarks
**SequenceJuxtaposer**

- dense grid, following conventions
  - rows of sequences partially correlated
  - columns of aligned nucleotides
  - videos
SJ Contributions

• accordion drawing for gene sequences
• paper results: 1.7M nucleotides
  – current with PRISAD: 40M nucleotides
• joint work: SJ credits

James Slack, Kristian Hildebrand, Tamara Munzner, and Katherine St. John.

SequenceJuxtaposer: Fluid Navigation For Large-Scale Sequence Comparison In Context.
Proc. German Conference on Bioinformatics 2004
www.cs.ubc.ca/~tmm/papers/sj
Outline

• trees
  – TreeJuxtaposer
• sequences
  – SequenceJuxtaposer
• scaling up trees
  – TJC
• general AD framework
  – PRISAD
• power sets
  – PowerSetViewer
• evaluation
Scaling Up Trees

• TJ limits
  – large memory footprint
  – CPU-bound, far from achieving peak rendering performance of graphics card

• quadtree data structure used for
  – placing nodes during layout
  – drawing edges given navigation
  – culling edges with GV
  – selecting edges during interaction
Navigation Without Quadtrees
Eliminating the Quadtree

• new drawing algorithm
  – addresses both ordering and culling
• new way to pick edges
  – uses advances in recent graphics hardware
• find a different way to place nodes
  – modification of O-buffer for interaction
Drawing the Tree

- continue recursion only if sub-tree vertical extent larger than apixel
  - otherwise draw flattened path

\[ y_1 \] \hspace{2cm} \hspace{2cm} y_1

\[ y_2 \] \hspace{2cm} \hspace{2cm} 22 \quad y_2
Guaranteed Visibility

• continue recursion only if subtree contains both marked and unmarked nodes
Picking Edges

- Multiple Render Targets
  - draw edges to displayed buffer
  - encoding edge identifier information in auxiliary buffer
TJC/TJC-Q Results

• TJC
  – no quadtree
  – requires HW multiple render target support
  – 15M nodes

• TJC-Q
  – lightweight quadtree
  – 5M nodes

• both support tree browsing only
  – no comparison data structures
Joint Work: TJC, TJC-Q Credits

Dale Beermann, Tamara Munzner, and Greg Humphreys. Scalable, Robust Visualization of Large Trees. Proc. EuroVis 2005
www.cs.virginia.edu/~gfx/pubs/TJC
Outline

• trees
  – TreeJuxtaposer
• sequences
  – SequenceJuxtaposer
• scaling up trees
  – TJC
• general AD framework
  – PRISAD
• power sets
  – PowerSetViewer
• evaluation
PRISAD

• generic accordion drawing infrastructure
  – handles many application types

• efficient
  – guarantees of correctness: no overculling
  – tight bounds on overdrawning
    • handles dense regions efficiently
  – new algorithms for rendering, culling, picking
    • exploit application dataset characteristics instead of requiring expensive additional data structures
<table>
<thead>
<tr>
<th>World-space discretization</th>
<th>Application</th>
<th>PRISAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layout</td>
<td>(x, y) size</td>
<td>Initialize</td>
</tr>
<tr>
<td>Gridding</td>
<td>(S_X, S_Y)</td>
<td>Mapping</td>
</tr>
<tr>
<td>S, node</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Screen-space rendering</td>
<td>Render</td>
<td>Partition</td>
</tr>
<tr>
<td>Seed</td>
<td>S, (\tau)</td>
<td>Progressive Rendering</td>
</tr>
<tr>
<td>Draw</td>
<td>S ranges</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Queue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Object</td>
<td></td>
</tr>
</tbody>
</table>
PRISAD Responsibilities

• initializing a generic 2D grid structure
  – split lines: both linear ordering and recursive hierarchy

• mapping geometric objects to world-space structures
• partitioning a binary tree data structure into adjacent ranges
• controlling drawing performance for progressive rendering
Application Responsibilities

• calculating the size of underlying PRISAD structures
• assigning dataset components to PRISAD structures
• initiating a rendering action with two partitioning parameters
• ordering the drawing of geometric objects through seeding
• drawing individual geometric objects
Example: PRITree

• rendering with generic infrastructure
  – partitioning
    • rendering requires sub-pixel segments
    • partition split lines into leaf ranges
  – seeding
    • 1st: roots of marked sub-trees, marked nodes
    • 2nd: interaction box, remainder of leaf ranges
  – drawing
    • ascent rendering from leaves to root
Tree Partitioning

• divide leaf nodes by screen location
  – partitioning follows split line hierarchy
  – tree application provides stopping size criterion
  – ranges \([1,1]; [2,2]; [3,5]\) are partitions

\[
L_0 = [1,1] \\
L_1 = [2,2] \\
L_2 = [3,5]
\]
Tree Seeding

- marked subtrees not drawn completely in first frame
  - draw “skeleton” of marks for each subtree for landmarks
  - solves guaranteed visibility of small subtree in big dataset
Tree Drawing Traversal

• ascent-based drawing
  – partition into leaf ranges before drawing
    • TreeJuxtaposer partitions during drawing
  – start from 1 leaf per range, draw path to root
  – carefully choose starting leaf
    • 3 categories of misleading gaps eliminated
      – leaf-range gaps
      – horizontal tree edge gaps
      – ascent path gaps
Leaf-range Gaps

- number of nodes rendered depends on number of partitioned leaf ranges
  - maximize leaf range size to reduce rendering
  - too much reduction results in gaps
Eliminating Leaf-range Gaps

- eliminate by rendering more leaves
  - partition into smaller leaf ranges
Rendering Time Performance

• TreeJuxtaposer renders all nodes for star trees
  – branching factor k leads to O(k) performance
• we achieve 5x rendering improvement with contest comparison dataset
• constant time, after threshold, for large binary trees
Rendering Time Performance

- constant time, after threshold, for large binary trees
  - we approach rendering limit of screen-space
- contest and OpenDirectory comparison render 2 trees
  - comparable to rendering two binary trees
Memory Performance

- linear memory usage for both
  - generic AD approach 5x better
- marked range storage changes improve scalability
  - 1GB difference for contest comparison
PRISAD Results

• video

• joint work: PRISAD credits

Outline

- trees
  - TreeJuxtaposer
- sequences
  - SequenceJuxtaposer
- scaling up trees
  - TJC
- general AD framework
  - PRISAD
- power sets
  - PowerSetViewer
- evaluation
PowerSetViewer

• data mining market-basket transactions
  – items bought together make a set
  – space of all possible sets is power set
    • place logged sets within enumeration of power set
PSV Results

• dynamic data
  – show progress of steerable data mining system with constraints
  – all other AD applications had static data
• handles alphabets of up to 40,000
• handles log files of 1.5 to 7 million items
• joint work in progress with
  – Qiang Kong, Raymond Ng
Outline

• trees
  – TreeJuxtaposer
• sequences
  – SequenceJuxtaposer
• scaling up trees
  – TJC
• general AD framework
  – PRISAD
• power sets
  – PowerSetViewer
• evaluation
Evaluation

• how focus and context are used with
  – rubber sheet navigation vs. pan and zoom
  – integrated scene vs. separate overview

• user studies of TJ
  – tasks based on biologist interviews

• joint work in progress, with
  – Adam Bodnar, Dmitry Nekrasovski, Joanna McGrenere
Conclusion

• accordion drawing effective for variety of application datasets
  – trees, sequences, sets

• guaranteed visibility is powerful technique
  – computational expense can be handled by generic algorithms
More Information

• papers, videos, images
  – www.cs.ubc.ca/~tmm

• free software
  – olduvai.sourceforge.net/tj
  – olduvai.sourceforge.net/sj