Constant Information Density in Zoomable Interfaces

Allison Woodruff, James Landay, Michael Stonebraker
The DataSplash Environment

- Direct-manipulation interface for constructing pannable/zoomable database visualizations
- Users can specify how much information is displayed at different elevations by a layer manager

![Diagram showing DataSplash environment with layer manager, elevation bar, tabular data, and layer rendering.]
The Problem

- The Principle of Constant Information Density – Number of objects per display unit should be constant -> Amount of information should remain constant as users pan and zoom
- DataSplash’s users have difficulty constructing well-formed applications that conforms to this principle, displaying constant level of detail at all elevations.

The Solution - “Measure, Visualize, Bound”

- Give users visual feedback about information density as they create each layer
- Guide users to maintain constant density
Visual Information Density Adjuster

- **Measures**
  - Density Metrics: number of objects or number of vertices
  - Other density functions can be defined

- **Visualizes**
  - Width of layer bars encodes density at a given elevation
  - Color of the elevation gauge indicates whether a level is too dense

- **Bounds**
  - Enforcing density boundaries is left to visualization designers
Semi-automatic Adjustment of Layer Density

- **Modification Functions**: modifying a layer’s density via
  - Creating views of data table (select/join)
  - Changing the graphical presentation of data

**Original Visualization**

- **Select**
- **Aggregate**
- **Reclassify**

- **Chg Shape**
- **Chg Size**
- **Chg Color**
- **Remove Attribute Assoc.**
Critique

Strengths

- Comprehensive description of techniques
- Extensive considerations of problems and possible solutions
- Encoding density with width is intuitive, because the cumulative width of all layers at a zoom level = cumulative density

Weaknesses

- A lot of repetition
- Pilot trial added as an after-thought and only mildly relevant to the paper’s topic
Speed-dependent Automatic Zooming
for Browsing Large Documents

Takeo Igarashi & Ken Hinckley
Rate-Based Scrolling – Scroll faster as you move your mouse faster

Problem 1: Motion Blur (Excessive Visual Flow)

Problem 2: Multiple pan/zoom needed

SDAZ – Automatic zoom-out to cover more distance instead of scrolling faster
SDAZ Implementation

- Mouse speed simulated by displacement of mouse cursor
- Scroll/Zoom is engaged by holding down a mouse button
- Releasing the mouse button will trigger a zoom-in with the center of the screen as reference
- The scale is first calculated

\[ \text{scale} = s_0(dy-d_0)(d_1-d_0) \]

\( s_0, d_0, d_1 = \text{const: minimum scale, starting mouse movement, maximum mouse movement} \)

- Then scrolling speed is calculated

\[ \text{Scrolling Speed} = \frac{v_0}{\text{scale}} \]

\( v_0 = \text{const: initial scrolling speed} \)
Reverse and Cessation Problems

Introduce a zoom-in delay factor to avoid “swellings” when changing direction.

Introduce a constant default zoom-in rate for when the user simply stop holding down the mouse button.

Sudden drops when reverse scrolling direction

Sudden catapulting downward when button is lifted
Test Applications

Web-browser with semantic zooming

Map viewer

Other Applications
- Image Browser
- Dictionary with semantic zooming (word-skip)
- Sound editor (zooming the waveform)
Usability Studies

- **Web-browser: SDAZ vs. Scrollbars**
  - Task completion time: roughly equal
  - Subjective preference: SDAZ
  - Video game players performed better
  - Constant flow of text can cause dizziness
  - Isometric input (joysticks) might improve performance, but not tried

- **Map Viewer: SDAZ vs. manual zoom-in/out buttons**
  - Task completion time: mixed to negative (for SDAZ)
  - Subjective preference: roughly equal
  - Overshoot and course-correction problem
  - Many subject develops coping strategies
Critique

Strengths

• Works well for 1D apps like web or image browser
• Requires no extra screen real estate
• Requires very simple input device
• Good for mobile!

Weaknesses

• Demanding high-dexterity, especially for 2D apps
• Unclear whether performance comes from SDAZ or semantic-zooming
Critical Zones in Desert Fog: Aids to Multiscale Navigation

Susanne Jul & George W. Furnas
Desert Fog

Does this view contain anything?

How can this view look like the other one? (minimum object rendering size)

Where do I go from here? (zoom out/in? pan?)

Can be mitigated at the info design/embedding stage

Particularly bad when encountered at navigation time
Fighting Desert Fog – Residues of Objects

Multi-scale Residue of Objects: red squares visible at all scales

Objects are clustered spatially, recursively to reduce the number of residues as you zoom out.

Problems: placement of landmarks, landmarks changing position during zoom-in, landmark can suggest false semantic associations.
Fighting Desert Fog – Residues of Views (Ztracker)

Critical Zones: residues of interesting views, zooming in reveals more interesting views (and critical zones representation of them)

Calculating 1 crit-zone: Bounding box of all objs in current view

Sub-divide and recurse: Critical Zone rectangle changes color when covers all world objects
View Navigation Analysis

- View-navigation theory provides a characterization of the properties that make an information structure navigable, adapted for spatial data.

- Viewing-graph a d-graph, nodes = views, links = traversible paths between views.

- A traversible world
  - Short path must exists between all nodes.
  - All nodes must have small number of outlinks.
  - “Small” and “Short” is relative to the complexity of the viewing graph.
Navigation Requirements

- All views must have good residue on all nodes
- All views must have small outlink info
- **Good residue**: correctly points out the shortest link to a node

=> In a zoomable world, merely providing residues solve the desert fog problem, because the lack residue means zoom-out

- **outlink-info**: the representation of the residue. E.g. a text label

- **Small**: Relative to number of overall views? Or navigator’s info processing capabilities?

=> Grouping such as landmarking and ZTracker
Critique

Strengths

- Novel concept: providing residue of views, not objects
- Thorough treatment of the subject from an implementation pov and a theoretical pov

Weaknesses

- Ztracker algorithm might be expensive. Some heuristics?
- Repeating diagrams with small differences makes navigating the paper confusing
- More examples of desert fog please?
Q&A

Thomas Dang, dqluan@gmail.com