Chapter 9 - Presentation

Dealing with the presentation problem

Vorlesung „Informationsvisualisierung“
Prof. Dr. Florian Alt, WS 2013/14

Konzept und Folien (4th revised edition):
Thorsten Büring, Andreas Butz, Michael Burch
Outline

• Presentation problem
• Zoomable user interfaces (ZUIs)
  – Development history
  – Space-scale diagrams
  – 2.5D
  – Advanced ZUI designs
  – Orientation in ZUIs
• Overview+detail interfaces
  – Abstract overviews
  – Performance issues
  – View coordination
  – View Layout
  – Zoom factors
• Focus+Context
  – Generalized Fisheye View
  – Graphical Fisheye
Presentation Problem
Presentation Problem

• Very often information spaces have to be displayed, which are significantly larger than the screen size
  – Too many data cases
  – Too many variables

• Potential techniques to maximize the number of information objects that can be displayed
  – Data encodings (see lectures 3 & 4)
  – Interaction and view transformations
  – Hybrid approaches
Presentation Problem

• Most common workaround: scrolling interfaces
• Advantages
  – Many users are familiar with scrollbars
  – Navigation at different speed
  – Thumbs show position and ratio of information space and view size
  – Have been found effective to move small distances
• Disadvantages
  – Only horizontal and vertical shifts
  – Scrollbars usually do not preview the content of the off-screen space
  – Take away screen space
  – Limited to linear navigation
  – Does not scale (search times and interaction sensitivity increase)
Overview & Detail Interfaces
Overview+Detail

• Overview+detail (O+D) interfaces are characterized by multi-window layout
  – Detail view presents details
  – Overview window provides overview information of the information space
  – Overview windows are usually also enhanced with visual cues

• O+d interface with field-of-view box gives users direct and constant feedback on their position in the information space

• Thus context information is preserved

North & Shneiderman 1997
The cars did not separate us.” (P21)

Again this is about travelling together, feeling as "one big group," although being separated. This time the communication is verbal (through the walky-talkies) but still a quite similar reflection on the value of a restricted communication is apparent. Both cars needed to stay close, the group needed to "earn the communication" by paying attention to each other. This seems a crucial aspect to the feeling of belonging.

This participant also described the "dynamics of separation", the moment they heard their friends before they saw them. Another participant shed light on those dynamics:

"I went on a weekend trip together with a couple of friends. We used two cars for the trip. On the highway, we quickly lost sight of our friends in the other car. A couple of hours later, a short time before we had to leave the highway, I suddenly saw the other car again. I was surprised, because I didn’t expect the others to be that close to us. Happily, I took my cell phone and called a friend in the other car. They were also happy to be close to us again." (P19)

This story further details this play of loosing and finding each other again, and the resulting pleasure. And also this story is about the same experience:

"After a weekend of wedding celebration, the guests all said goodbye to each other and went on their way in their cars – each one heading for very different cities in Germany. None

DESIGN

From a story headline to the Experience Story and the Storyboard

Based on the "motorcade" pattern we started to specifically design the experience to be created through the envisioned interactive product. First, we created a "story headline", which summarized the most important ingredients of the experience to be created:

Clique Trip provides the experience of being one group even when being in different cars - as if the interiors of the cars unite. To create this experience Clique Trip offers a communication channel, which is restricted and needs some effort to be established and maintained. In addition, Clique Trip plays with the tension between the feeling of separation and closeness.

Based on this story headline, we developed an Experience Story. This designed story is told from a user's point of view while interacting with the envisioned product. It already describes interactions with the product and the (ideally) resulting emotions and experiences. It is this explicit focus on emotion and experience that distinguishes an Experience Story from scenarios used, for example, in extreme programming [1] or in interaction design [16]. Another important aspect is that the story must be compelling for every involved person in the design process [6]. As a narrative provides an overall structure [25], the Experience Story thereby facilitates communication between the different people involved in the design. The non-technical character of the story is crucial to the
O+D in Google Maps
Abstract Overviews

- When showing a miniature of a reasonably large information space much detail information may be lost
- Could in some cases be solved by presenting intermediate views, but: display space limitations
- Abstract overviews use encodings to use limited screen space more effectively
- May also contain extra information not present in the detail view
- Example: document overview (Jerding & Stasko 1995)
  - Overview always shows the entire document
  - Intensity scale indicates text density
  - Color denotes sections
Abstract Overviews

\subsection{Our Solution}

We have developed a method for displaying and navigating large information spaces using the multiple view technique. This work is derived from our implementation of the message trace visualization.

Tsakonas, Jarc, and Shneiderman propose three important considerations in the design of multiple-view browsers: window-placement strategy, view coordination, and the global view itself. The last section describes the design of the message view and its navigation mechanisms. We call our views of large information spaces \textit{Information Murals}, and describe them in Section \ref{sec:nomral}. In Section \ref{sec:nomral} we discuss our initial application and how the information murals are useful, and compare our method with related work in those areas.

\section{The Execution Mural}

As mentioned in the last section, one area of our software visualization research involves visualizing the execution of object-oriented programs. As a component of an integrated set of views, we are designing a display of the messaging exchange between objects during the execution of a C++ program. This section describes the \textit{Execution Mural}, focusing specifically on the visual mechanisms used to provide navigation capabilities. While the current design of the system does not contain all the functionality envisaged for this view, it does provide an effective demonstration of our methods. The techniques discussed in this section will be generalized to other information spaces in Section \ref{sec:nomral}.
Interface Performance

• Task-completion time
  – Navigation on the overview may significantly improve the interface performance
  – E.g. users can directly navigate to locations that are currently not visible on the detail view
  – Drawback: multiple views require time-consuming visual switching between views

• User study by Hornbaek et al. 2002
  • 32 participants, counterbalanced within-subjects design
  • Browsing and navigation tasks on two maps
  • Two semantic ZUIs, one with and one without overview
  • Participants were faster with the detail-only interface
  • 80% preferred the overview-enhanced interface

Hornbaek et al. 2002
View Coordination

• Most simple o+d: overview shows a static image of the information space
  – Users are forced to compare the visual cues in the detail view with the cues in the overview
  – For reasonably large and complex information spaces, this approach is hardly usable

• Dynamic overviews
  – Visual cues such as a field-of-view box aid orientation
  – Implies coordination of views

• Coordination (also termed tight coupling)
  – Unidirectional: only one view is interactive
  – Bidirectional: supports user input in both views

• Study by North & Shneiderman 2000: coordinated views were found to be 30% to 50% faster than a detail-only interface and a o+d interface with two independent views
View Layout

• Basic side-by-side layout of views require that the available display space is partitioned between the views
• Problem: for both views the usability increases with a growing size
• No general solution for the space tradeoff
• Layout of the views is task-dependent (Plaisant 1995)
  – Open-ended exploration or drawing tasks require a larger detail view
  – Monitoring tasks require a larger overview
Alternative View Layouts

- Overlapping views
  - Overview overlaps with the detail view (e.g. Acrobat overview)
  - Users can drag and scale the overview view as desired
  - Problem: managing windows is time-consuming and adds extra complexity to the interface

- Automatic overviews
  - System decides when to (temporarily) display an overview
  - How to predict the need for an overview?
  - E.g. extensive zooming and panning on the detail view
  - Malfunction can be highly annoying

- Transparent overviews
  - Can be applied to both overlapping and automatic overviews
  - Problems: increased visual clutter and deteriorated readability of both detail view and overview
Zoom Factors

- Zoom factor: level of magnification between detail view and overview
- Should be
  - Less than 20 (Plaisant 1995)
  - Between 3 and 30 (Shneiderman & Plaisant 2005)
- Larger zoom factors may require intermediate views
Focus & Context
Focus+Context

• Recap presentation problem: information space is too large to be displayed on a single screen

• Approaches in previous lecture
  – Zoomable user interface: scale and translate a single view of the information space
  – Overview+detail: use multiple views with different scale / detail granularity

• Focus+Context (f+c) means a presentation technique where both focus and context information are integrated into a single view by employing distortion
  – Local detail for interaction
  – Context for orientation

• No need to zoom out to regain context as in ZUIs
• No need to switch and relate between multiple separate views as in overview+detail interfaces
• Focus+context is commonly known as fisheye views
• Earliest mentioning of the idea in Ph.D. thesis: Farrand 1973
Focus+Context Screens  [Baudisch 2001]

http://www.youtube.com/watch?v=yKe2ahApaz4
Generalized Fisheye Views

• Furnas 1986
• Idea: trade-off of detail with distance
• Naturally occurring, e.g.
  – Employees being asked about the management structure: they know local department heads, but only the Vice president of remote divisions
  – Regional newspaper contain local news stories and only more distant ones that are compensatingly of greater importance (e.g. war in a remote country)
• Formalization
  – Presentation problem: interface can only display n items of a structure that has a number of items > n
  – Degree-of-interest function: assign importance value to each item in structure - only display the n most important items
Degree-of-Interest

• $\text{DOI}_{\text{fisheye}}(x|y) = \text{API}(x) - D(x,y)$
  - $\text{DOI}_{\text{fisheye}}$: the users' degree of interest in point $x$, given the focus point $y$
  - $\text{API}(x)$: Global a priori importance of point $x$
  - $D(x,y)$: distance between $x$ and focus point $y$

• Can be applied to any structure for which the components can be defined

• Example: rooted tree structure of programming code

• Components definition
  - $D(x,y) = d_{\text{tree}}(x,y) = \text{path length between node } x \text{ and node } y \text{ in the tree}$
  - $\text{API}(x) = -d_{\text{tree}}(x,\text{root}) = \text{distance of node } x \text{ from the root node (nodes closer to the root are generally more important than nodes further away)}$

• $\text{DOI}_{\text{fisheye(tree)}}(x|y) = \text{API}(x) - D(x,y) = -(d_{\text{tree}}(x,y) + d_{\text{tree}}(x,\text{root}))$
Fisheye Tree

(a) Distance from y:
\[ d_{\text{tree}}(x,y) \]

(b) A Priori Importance in the tree:
\[ \text{Imp}(x) = -d_{\text{tree}}(x, \text{root}) \]

(c) The Fisheye DOI:
\[ \text{DOI}_{\text{fisheye(tree)}}(x, y) = \text{API}(x) - D(x, y) = -(d_{\text{tree}}(x, y) + d_{\text{tree}}(x, \text{root})) \]

An arithmetically larger number means that the node is more interesting for interactions focused on y.
Fisheye Tree

- To obtain fisheye views of different sizes, set a DOI threshold $k$ with $\text{DOI}(x) > k$

$k = -3$; direct ancestral lineage
(a) Zero-order tree fisheye:

$k = -5$; siblings are added
(b) First-order tree fisheye:

$k = -7$; cousins are added
(c) Second-order tree fisheye:
Fisheye Tree Applied

- Full view of the program
  - Box: lines in default view
  - Underlines: lines in fisheye view
Fisheye Tree Applied

• Working on line marked with „>>“

```c
#define DIG 40
#include <stdio.h>

int main()
{
    int c, i, x[DIG/4], t[DIG/4], k = DIG/4, noprint = 0;
    while((c = getchar()) != EOF){
        if(c == '0' && c <= '9'){
            for(i = 0; i < k; i++) t[i] = x[i];
            break;
        } else {
            switch(c){
                case '+':
                    case '-':
                    case 'e':
                        for(i = 0; i < k; i++) t[i] = x[i];
                        break;
                    case 'q':
                        exit(0);
                    default:
                        switch(c){
                            case '+':
                                case '-':
                                case 'e':
                                    for(i = 0; i < k; i++) t[i] = x[i];
                                    break;
                                case 'q':
                                    exit(0);
                                default: noprint = 1;
                                    break;
                                } if(!noprint){
                                    for(i = 0; i < k; i++) t[i] = 0; // This line is marked with ">>"
                                    printf("%d", t[i]);
                                if(i > 0) {
        ...
```

Figure 3. Standard 'flat-window' view of a C program. Line numbers are in the left margin.

Figure 4. A fisheye view of the C program. Line numbers are in the left margin. "..." indicates missing lines.
Graphical Fisheye Views

- Applied rather to layouts than to logical structure
- Furnas fisheye: items are either present in full detail or absent from the view
- Objective: continuous distortion of items and item representation
Bifocal Display

- Spence & Apperley 1982
- Office environment of the future
- Virtual workspace showing documents on a horizontal strip
- Centered detail region and two compressed context regions
- Scroll compressed documents in the detail region to decompress
- Distortion increases the amount of information that can be displayed
Perspective Wall

- Robertson et al. 1991
- Same approach as the bifocal lens but using perspective
- Detail information about objects recedes into the distance
Document Lens

• Robertson & Macklnlay 1993
Distortion Approaches Used

- Overview of the different distortion techniques

Bifocal display

Perspective wall

Document lens

Magnification

Transfer function
Graph Fisheye

• Sarkar & Brown 1994
• Fisheye lens for viewing and browsing large graphs
• Present focus vertex in high detail but preserve context
• Recap node-link representation
  – Vertex (node)
  – Edges (links)
How did they do that…?

- Focus: viewer’s point of interest
- Coordinates in the initial layout: regular geographic coordinates
- Coordinates in the fisheye view: fisheye coordinates
- Each vertex has
  - A normal position specified by geographic coordinates
  - Size (Length of the square-shaped bounding box)
  - A priori importance (API)
  - Edge
    - Straight line from one vertex to another OR
    - For bent edges: set of intermediate bend points
- Apart from the distortion, the systems calculates for each vertex:
  - Amount of detail (content) to be displayed
  - Visual weight: shall the vertex be displayed? - display threshold
Implementation

- Two step process
  - Apply geometric transformation to the normal view to reposition vertices and magnify / demagnify the bounding boxes
  - Use the API of vertices to determine their final size, detail, and visual weight
- Slides will only present the repositioning of vertices - for the remaining algorithm see the paper at ftp://ftp.cs.brown.edu/pub/techreports/93/cs93-40.pdf
Cartesian Transformation

• Compute the position of a point $P_{\text{norm}}$ from normal coordinates to fisheye coordinates

$$P_{\text{feye}} = \langle g \left( \frac{D_{\text{norm}_x}}{D_{\text{max}_x}} \right) D_{\text{max}_x} + P_{\text{focus}_x},$$

$$g \left( \frac{D_{\text{norm}_y}}{D_{\text{max}_y}} \right) D_{\text{max}_y} + P_{\text{focus}_y} \rangle$$

• where

$$g(x) = \frac{(d + 1)x}{dx + 1}$$

• $D_{\text{max}}$: the horizontal / vertical distance between the boundary of the screen and the focus in normal coordinates

• $D_{\text{norm}}$: horizontal / vertical distance between the point being transformed and the focus in normal coordinates

• $d$: distortion factor, see graphs
Distortion Factor

• Example: distortion of a nearly symmetric graph
• Focus in the southeast
Polar Transformation

- With cartesian transformation all vertical and horizontal lines remain vertical and horizontal in the fisheye view.
- Makes this approach well suited for abstract orthogonal layouts of information spaces (e.g. circuit design, UML diagrams, etc.)
- Problem: does not seem very natural.
- Alternative approach: distorting the map onto a hemisphere using polar coordinates (origin = focus).
- Point with normal coordinates \((r_{\text{norm}}, \theta)\) is mapped to fisheye coordinates \((r_{\text{feye}}, \theta)\), where
  
  \[ r_{\text{feye}} = r_{\text{max}} \frac{(d + 1)r_{\text{norm}}}{d r_{\text{norm}} + 1} \]

  - \(r_{\text{max}}\): maximum possible value of \(r\) in the same direction as \(\theta\).
  - Note: \(\theta\) remains unchanged, origin of polar coordinates is the focus.
  - Distortion forms a pyramid lens.
  - Users know this effect from lenses and elastic materials in the real world, often find it fascinating.
Cartesian vs Polar Transformation

Cartesian

Polar
Discussion break

• What do you think of this? Ideas?
More Fisheye Lenses

• Gutwin & Fedak 2004

Original pyramid lens (polar transformation, full screen)

Constrained hemispherical lens: constrain polar algorithm to a fixed radius

Constrained flat-hemispherical lens: insert a region of constant magnification
Victor Vasarely (1906-1997)
Multiple Foci

- Keahey & Robertson 1996
- Also multiple foci in a single domain are possible
- Interesting question: how to handle overlap?
Problem: Focus Targeting

- Gutwin 2002
- Move the fisheye lens to a target
- Problem: targets appear to move and thus are more difficult to hit directly (same effect as with a simple magnifying lens)
- Movement is in the opposite direction to the motion of the fisheye lens: focus target will move towards the approaching lens and vice versa
Focus Targeting

• Even worse: with the fisheye lens, targets move towards the focus more and more rapidly as the focus approaches them
• Depending on the distortion factor, the targets may move several times faster than the focus
• Leads to overshooting
• Approach to reduce problem: speed-coupled flattening
  – Detecting a target acquisition, the system automatically reduces the distortion
  – Distortion is automatically restored when the target action is completed
  – Algorithm is based on pointer velocity and acceleration thresholds
Speed-Coupled Flattening

- Found to significantly reduce targeting time and errors

Figure 4. Speed-coupled flattening. Top row shows the fisheye view and pointer path. Bottom row shows a stylized plot of pointer velocity and distortion level. The dotted line indicates the point in time that the corresponding screen was captured.
Discussion: Mac OS X Dock