Information Visualization for Mobile Devices

Benjamin B. Bederson
Computer Science Department
Human-Computer Interaction Lab (HCIL)
University of Maryland

Scaling Information, Scaling Devices

- We always want more: information, pixels, speed, …
- Whether big or small, the problems are similar:
  - Detect patterns and outliers
  - Find details without losing global context
  - Concentrate on task (stay "in the flow")

Information Visualization

Scaling Up

- How to scale up to large information sets?
  - Technical problems
  - Perceptual limitations
  - Design problems
- Solution in just three easy steps:

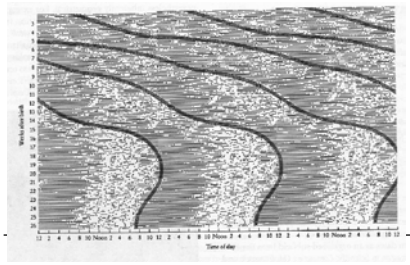
External Cognition

- Step 1: Recognize human limitations

$$\begin{array}{r} 34 \\ \times\ 72 \\ \hline 68 \\ 23^{2}80 \\ \hline 24^{1}48 \end{array}$$

Human Visual Perception

- Step 2: Don't underestimate the human brain



Interaction

- Step 3: Add interaction. If a picture is worth a thousand words, an interactive interface is worth a thousand pictures.

Scaling Up

- How do you show more than fits on the screen?

traditional
- abstract
- link (web)
- scroll (long documents)
- overview+detail (e.g., Photoshop)

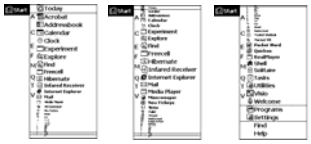
new paradigms
- zoom
- fisheye distortion

Menus, menus, menus

- Problem: Selection from a long list
- Growing importance with data-driven applications
- Traditional approaches:
  - ArrowBars
  - ScrollBars
  - Hierarchies

Fisheye Menus

- Apply fisheye distortion to linear list
- Shows detail in context
- Reduces mouse *presses / taps*



- *Pocket PC version written in embedded C++*

Demo

www.cs.umd.edu/hcil/fisheyemenu

Fisheye Menu Evaluation

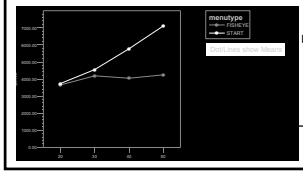
- Fisheye Menu v. Microsoft Pocket PC 2000™
- Goal:
  - Understand scalability of fisheye menus
- Used real applications and icons

Methods

- 22 participants
- Three independent variables
  - Menu type (fisheye vs. regular)
  - Menu length (20, 30, 40, 50)
  - Position of target within menu (first, second, or last third)
- Measured:
  - Time
  - Accuracy
- Total of 1230 trials

Results —Task Times

- Tasks were performed faster using Fisheye Menus, $F(1,1206)=29.4$, $p<0.001$
  - 25% faster (4.0 vs 5.3 secs)
- Difference more pronounced for longer menus



- And more pronounced for items near the end of the menus

Results – Task Accuracy

- But, tasks were completed successfully significantly more often using traditional menu (97% v. 92%), $F(1,1206)=22$, $p<0.001$
- Difference more pronounced for items in middle of menu

Results – User Preference

	Start Fisheye		
Prefer for short:	**82%**	**18%**	**p=.04**
Prefer for long:	**14%**	**86%**	**p<.001**
Perceived faster:	**14%**	**86%**	**p<.001**
Perceived easier:	**68%**	**32%**	**p=.005**

Discussion

- Cool idea, but not a clear winner
- Some older users complained that it made them feel bad about their eyesight
- Other users wanted it immediately for all of their menus…
- Interaction simpler on PDA because of Pen input
- Bottom line: probably a good idea to offer as an option

Photo Browsing

Many tools support image management, annotation, and search.

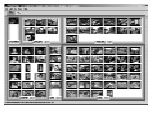
Our goal is to focus on ***browsing***

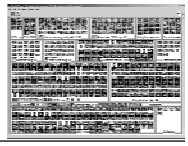
- avoid scrollbars, menus, and window mgmt
- work in a home setting for family use
- support co-present collaborative use
- support serendipitous photo finding

Pocket PhotoMesa

Design:

- Zoomable User Interface
  - Simple navigation interaction
- Quantum Strip Treemap layout
  - Shows multiple directories of images
  - Can group by metadata





Demo

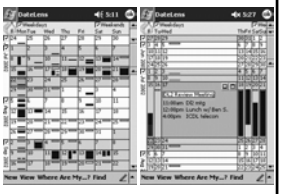
www.cs.umd.edu/hcil/photomesa

Calendar Management

- How to better:
  - support *planning* and *analysis* tasks
  - move between multiple device types
- The answer:
  - more Information Visualization techniques

DateLens

- To scale up and maintain context:
  - Uses 2D fisheye distortion
  - Carefully designed interaction
    - simple or manual control over space
  - Integrated search with or without text entry
- Written in C# with same code running on Tablet PC and desktop



Demo

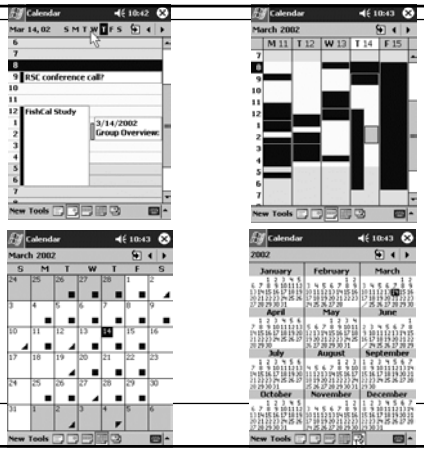
www.cs.umd.edu/hcil/datelens

DateLens Performance

- Careful coding
- Custom renderer and picker
- Space-time trade-off
- Minimal rendering during transitions
- Careful evaluation of how to render

DateLens Benchmark Study

- DateLens v. Microsoft Pocket PC 2002™
- Goals
  - 1st iteration of UI with potential users
  - to compare its overall usability against an existing product
- Used Mary's calendar, seeded with artificial calendar events



Methods

- 11 knowledge workers (5 F)
  - All experienced PC (not PDA) users
- 11 isomorphic browsing tasks on each calendar
  - All conditions counterbalanced
  - All tasks had deadline of 2 minutes
- Measured:
  - task times
  - success rate
  - verbal protocols
  - user satisfaction and preference

Results —Task Times

- Tasks were performed faster using DateLens, $F(1,8)=3.5$, $p=.08$
  - Avg=49 v. 55.8 sec's for the Pocket PC
  - Complex tasks significantly harder, $p<.01$, but handled reliably better by DateLens (task x calendar interaction), $p=.04$

Task Success

- Tasks were completed successfully significantly more often using DateLens (88.2% v. 76.3%), p<.001.
- In addition, there was a significant main effect of task, p<.001.
- For the most difficult task (#11), no participant using the Pocket PC completed the task successfully.

Usability Issues

- Many users disliked the view when more than 6 months were shown
- Concerns about the readability of text, needs to be customizable
- Wanted more control about how weeks were viewed (e.g., start with Sunday or Monday?)
- Needed better visual indicators of conflicts for both calendars, e.g., red highlights and/or a "conflicts" filter

Discussion

- DateLens performed well despite its novelty and its first iteration of user testing
- DateLens Positive Features:
  - Responsivity to direct user manipulation
  - Ability to create custom views easily
  - Clearer presentation of conflicts
  - Integrated search utility
- The PPC calendar was seen to be consistent with other MS calendar products (good)
- A combination of the 2 would be a great final product

Piccolo – A ZUI/Structured Canvas Toolkit

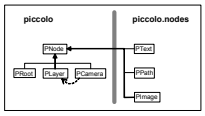
- Building Info Vis applications is hard
- Common features not typically supported:
  - Custom visual objects
  - Non-rectangular/non-opaque
  - Transforms
  - Lightweight
  - Picking
  - Region management
  - Multiple views
  - Zooming
  - Animation
  - Event Handlers

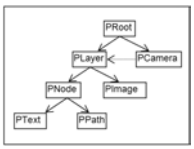
Example potential applications:
- Drawing programs
- Presentation programs
- Visualizations
- Excel…

www.cs.umd.edu/hcil/piccolo

Piccolo – What is it?

- A toolkit to support custom visual apps. First written in Java, being ported to C#, and compact framework





Piccolo – What does it look like?

```
import javax.swing.*;
import edu.umd.cs.piccolo.*;
import edu.umd.cs.piccolo.nodes.*;
import edu.umd.cs.piccolox.*;

public class PHelloWorld extends PFrame {
    public void initialize() {
        PText text = new PText("Hello World!");
        getCanvas().getLayer().addChild(text);
    }

    public static void main(String args[]) {
        new PHelloWorld();
    }
}
```

Java

C#

```
using Piccolo;
using Piccolo.Nodes;
using Piccolox;

public class PHelloWorld : PForm {
    public void Initialize() {
        PText text = new PText("Hello World!");
        Canvas.Layer.AddChild(text);
    }

    public static void Main() {
        new PHelloWorld();
    }
}
```

Demo

Porting – Porting to C# ☺

- Amount better than win32: ∞
- Amount better than embedded C++: ∞ ∞
- Competitive with Java
  - Fast launch
  - Simple and clean widget layout mechanism
  - Properties and events make code cleaner
  - Visual Studio and Eclipse both good (but…)

Piccolo – Porting to C# ☹

- Issues:
  - Extending core classes impossible because they are sealed and there is no interface (e.g., Graphics, Matrix, Pen)
    - ⇒ Can't create custom version (i.e., OpenGL)
  - Extending structs is impossible
    - ⇒ Can't extend Rectangle class w/ empty bit
    - ⇒ Thus can't pass our custom rect into existing method that expects Rectangle (surprisingly major issue)
  - No non-static or anonymous inner classes
    - ⇒ Fills namespace
  - Graphics support similar and generally excellent, but
    - ⇒ Java scaled image rendering ~50% faster (w/out hardware accel)
    - ⇒ Java graphics uses hardware acceleration when possible
  - Properties don't scale to setting w/ multiple params

Piccolo – Porting to Compact Framework

- Has floats, and very easy to port from C#
- Main missing features:
  - Matrix class and application to Graphics
  - Path
- Uses same API as full framework
  - Core DateLens on Pocket PC in ~2 hours (seriously!)
- But… as w/ J2SE, more limited graphics model
- Floats are slow, so we wrote a FixedPoint class and used operator overloading (hardware issue, not CF)

Conclusion

- Writing mobile apps is no longer an extreme sport
- Porting from C# is straight forward
- PDAs are now powerful enough to support rich visual applications, and current tools support this well
- Designing scalable interfaces are even more important on small devices