Instrumental Interaction

Michel Beaudouin-Lafon
Laboratoire de Recherche en Informatique
Université Paris-Sud / CNRS
mbl@lri.fr
http://insitu.lri.fr

Overview

Analysis of WIMP applications

Power vs. Simplicity

Interaction model

Instrumental Interaction

Design Principles

Analysis of WIMP applications

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Word6</th>
<th>Excel5</th>
<th>Persuasion3</th>
<th>Photoshop4</th>
<th>Xpress3</th>
<th>Canvas3</th>
</tr>
</thead>
<tbody>
<tr>
<td>W6</td>
<td>108</td>
<td>127</td>
<td>143</td>
<td>167</td>
<td>151</td>
<td>173</td>
</tr>
<tr>
<td>E5</td>
<td>124</td>
<td>120</td>
<td>140</td>
<td>160</td>
<td>155</td>
<td>170</td>
</tr>
<tr>
<td>Pe3</td>
<td>137</td>
<td>135</td>
<td>150</td>
<td>165</td>
<td>157</td>
<td>173</td>
</tr>
<tr>
<td>P4</td>
<td>143</td>
<td>141</td>
<td>160</td>
<td>175</td>
<td>163</td>
<td>183</td>
</tr>
<tr>
<td>X3</td>
<td>151</td>
<td>150</td>
<td>165</td>
<td>180</td>
<td>173</td>
<td>193</td>
</tr>
<tr>
<td>C3</td>
<td>160</td>
<td>160</td>
<td>180</td>
<td>200</td>
<td>193</td>
<td>213</td>
</tr>
<tr>
<td>Avg</td>
<td>144.8</td>
<td>144.8</td>
<td>164.8</td>
<td>184.8</td>
<td>174.8</td>
<td>194.8</td>
</tr>
<tr>
<td>#menus</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>#cmds</td>
<td>106</td>
<td>97</td>
<td>111</td>
<td>99</td>
<td>74</td>
<td>95.2</td>
</tr>
<tr>
<td>#dlogs</td>
<td>49</td>
<td>40</td>
<td>20</td>
<td>27</td>
<td>40</td>
<td>21</td>
</tr>
<tr>
<td>#smenu</td>
<td>1</td>
<td>15</td>
<td>27</td>
<td>26</td>
<td>13</td>
<td>22</td>
</tr>
<tr>
<td>#scmds</td>
<td>5</td>
<td>58</td>
<td>73</td>
<td>82</td>
<td>65</td>
<td>121</td>
</tr>
<tr>
<td>#sdlogs</td>
<td>0</td>
<td>20</td>
<td>20</td>
<td>40</td>
<td>10</td>
<td>28</td>
</tr>
<tr>
<td>Tcmds</td>
<td>108</td>
<td>124</td>
<td>143</td>
<td>167</td>
<td>151</td>
<td>173</td>
</tr>
<tr>
<td>Tdlogs</td>
<td>69</td>
<td>64</td>
<td>40</td>
<td>40</td>
<td>50</td>
<td>49</td>
</tr>
<tr>
<td>Cmds/M</td>
<td>13.3</td>
<td>10.5</td>
<td>13.9</td>
<td>13.9</td>
<td>14.1</td>
<td>9.3</td>
</tr>
<tr>
<td>Cmds/SM</td>
<td>3.0</td>
<td>3.9</td>
<td>2.7</td>
<td>3.2</td>
<td>5.0</td>
<td>5.5</td>
</tr>
<tr>
<td>#palettes</td>
<td>9</td>
<td>13</td>
<td>5</td>
<td>11</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>#tools</td>
<td>125</td>
<td>106</td>
<td>54</td>
<td>77</td>
<td>68</td>
<td>60</td>
</tr>
<tr>
<td>#prefs</td>
<td>12</td>
<td>10</td>
<td>1</td>
<td>8</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>#options</td>
<td>113</td>
<td>76</td>
<td>11</td>
<td>51</td>
<td>82</td>
<td>27</td>
</tr>
<tr>
<td>macros</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

Number of commands

Tcmds = #cmds - #smenu + #scmds
Tdlogs = #dlogs + #sdlogs
Cmds/M = mean commands per menu = #cmds / #menus
Cmds/SM = mean commands per sub-menu = #scmds / #smenu


(c) 2011, Michel Beaudouin-Lafon, mbl@lri.fr
Successive versions

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Excel 4-&gt;5</th>
<th>Word 5-&gt;6</th>
<th>Photoshop 2.5-&gt;4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E4</td>
<td>E5</td>
<td>%</td>
</tr>
<tr>
<td>#menus</td>
<td>8</td>
<td>8</td>
<td>0%</td>
</tr>
<tr>
<td>#cmds</td>
<td>93</td>
<td>84</td>
<td>-10%</td>
</tr>
<tr>
<td>#dlog</td>
<td>60</td>
<td>44</td>
<td>-27%</td>
</tr>
<tr>
<td>#smenu</td>
<td>0</td>
<td>15</td>
<td>+5%</td>
</tr>
<tr>
<td>#scmds</td>
<td>0</td>
<td>58</td>
<td>+14%</td>
</tr>
<tr>
<td>#sdlog</td>
<td>0</td>
<td>20</td>
<td>+14%</td>
</tr>
<tr>
<td>Tmenus</td>
<td>93</td>
<td>127</td>
<td>+37%</td>
</tr>
<tr>
<td>Tdlogs</td>
<td>60</td>
<td>64</td>
<td>+7%</td>
</tr>
<tr>
<td>Cmds/M</td>
<td>11.6</td>
<td>10.5</td>
<td>-10%</td>
</tr>
<tr>
<td>Cmds/SM</td>
<td>0</td>
<td>3.9</td>
<td>+14%</td>
</tr>
<tr>
<td>#palettes</td>
<td>8</td>
<td>13</td>
<td>+63%</td>
</tr>
<tr>
<td>#tools</td>
<td>108</td>
<td>106</td>
<td>-2%</td>
</tr>
<tr>
<td>#prefs</td>
<td>0</td>
<td>10</td>
<td>+100%</td>
</tr>
<tr>
<td>#options</td>
<td>0</td>
<td>76</td>
<td>+14%</td>
</tr>
<tr>
<td>macros</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

Analysis of WIMP applications

Power vs. Simplicity

Simple things should be simple
Complex things should be possible
How to combine power & simplicity?

More is less: the illusion of power

Bloatware
Too many functions
More functions with each new version
Marketing software: increased power?

Add features
- More menu items - Each is harder to find
- More commands - Each is harder to learn
- More dialog boxes - More steps to the goal

Add programming
- Macros
- Scripting languages
- Require users to understand programming concepts

Marketing software: increased simplicity?

Add wizards
- Hard to understand: What did the wizard do?
- Lose control: Wizard may do the wrong thing
- Waste time: Must fix the wizard’s mistakes

Add Customization:
- Preferences menus
  - Hard to navigate
  - Hard to translate into user’s terms
  - Hard to choose relevant settings
  - Rarely sharable
  - Most users don’t bother

Costs vs. benefits

Simple things are harder
Complex things are not used

Cost of learning
- Learned skills made obsolete
- No path from novice to expert

Cost of making choices
- Cognitive: more decisions
- Sensory-motor: more steps

A better approach

Specializing software
Example: Apple Macintosh

Final Cut Pro
Aperture
iMovie
iPhoto

power

simplicity
Another approach

Shifting the curve

\[ \text{power} \rightarrow \text{simplicity} \]

Going beyond WIMP

Complexity can be simple

Comparison: Bryce vs WIMP

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Avg</th>
<th>Bryce2</th>
<th>% of Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>#menus</td>
<td>7.7</td>
<td>3</td>
<td>38.9%</td>
</tr>
<tr>
<td>#cmds</td>
<td>95.2</td>
<td>45</td>
<td>47.3%</td>
</tr>
<tr>
<td>#dlog</td>
<td>36.8</td>
<td>18</td>
<td>48.9%</td>
</tr>
<tr>
<td>#smenu</td>
<td>17.3</td>
<td>0</td>
<td>0.0%</td>
</tr>
<tr>
<td>#scmds</td>
<td>67.0</td>
<td>0</td>
<td>0.0%</td>
</tr>
<tr>
<td>#tdlog</td>
<td>19.7</td>
<td>0</td>
<td>0.0%</td>
</tr>
<tr>
<td>Tcmds</td>
<td>144.8</td>
<td>45</td>
<td>31.1%</td>
</tr>
<tr>
<td>Tdlogs</td>
<td>56.5</td>
<td>18</td>
<td>31.8%</td>
</tr>
<tr>
<td>Cmds/M</td>
<td>12.5</td>
<td>15.0</td>
<td>120.0%</td>
</tr>
<tr>
<td>Cmds/SM</td>
<td>3.9</td>
<td>0.0</td>
<td>0.0%</td>
</tr>
<tr>
<td>#palettes</td>
<td>8.3</td>
<td>9</td>
<td>108.4%</td>
</tr>
<tr>
<td>#tools</td>
<td>81.7</td>
<td>71</td>
<td>86.9%</td>
</tr>
<tr>
<td>#prefs</td>
<td>7.8</td>
<td>1</td>
<td>12.8%</td>
</tr>
<tr>
<td>#options</td>
<td>60.0</td>
<td>5</td>
<td>8.3%</td>
</tr>
</tbody>
</table>

No menus, No windows, No dialog boxes

Graphical design Interaction design Layered approach

(c) 2011, Michel Beaudouin-Lafon, mbl@lri.fr
Case study: CPN 2000 Project

Redesign of Design/CPN
Current use world-wide: 600+ organizations

Purpose:
Edit and simulate coloured Petri Nets

Opportunity:
Explore research questions with a real-world application

Two key design decisions

Support two-handed input
Dominant and non-dominant hands

Integrate four interaction techniques:
Toolglasses Floating palettes
Contextual menus Bi-manual interaction

Why these techniques?
User studies show context affects tool preference
Palettes: focus on command
Marking menus: focus on object
Toolglasses: mixed focus

Less is more: the power of simplicity

CPN2000 case study
New version has more power but
no menu bar
no title bars
no scrollbars
no dialog boxes
no selection

This required
Participatory design process
Interaction model
Implementation from scratch

(c) 2011, Michel Beaudouin-Lafon, mbl@lri.fr
### Interaction model

**Definition**
- Set of principles, rules and properties that guide the design of an interactive system
- Helps combine interaction techniques in a consistent way

**Properties**
- **Descriptive:** describes a range of existing interactive systems
- **Evaluative:** helps evaluate interactive systems
- **Generative:** helps create new interaction techniques

---

### Need for a new interaction model

**Direct manipulation**
- ... is often too indirect

**Support more direct forms of interaction**

---

### Instrumental interaction

**Inspiration**
- Interaction with our environment is mediated by tools and instruments

**Two categories of objects**
- Domain objects
- Interaction instruments

---

### Interaction instruments

**Conceptual model**

**Two levels of interaction: mediation**
Instruments and modes

An instrument turns a mode into an object

Activating a mode = activating an instrument
Spatial mode: pointing

Temporal mode: selection

Cost of activation

Describing current WIMP interfaces

WIMP interfaces are based on widgets

Instruments of (in)direct manipulation

Handles, Title bars
Menus, Toolbars
Scrollbars
Dialog and Property boxes

Describing novel interaction techniques

Dynamic Queries
Dropable Tools
Toolglasses

Describing novel interaction techniques

Tangible interfaces
More input devices and therefore more instruments

Augmented/Mixed reality
Augmenting physical objects with computational capabilities
Evaluation: Properties of an instrument

Degree of indirection
- Spatial offset
- Temporal offset

Degree of integration
- How to use the degrees of freedom of the physical device
  - Integrality & separability of input devices (Jacob et al., 94)

Degree of conformance
- Similarity between physical action and effect on object

Generative power: Three design principles
- Reification
  - Extends the notion of what constitutes an object
- Polymorphism
  - Extends the power of commands with respect to these objects
- Reuse
  - Provides a way of capturing and reusing interaction patterns
Example: text search instrument

- Classic search:
  - Sequential
  - Modal

- Search instrument:
  - Show all occurrences
  - Allow replacing occurrences in any order

- Augmented scrollbar

Reification

- Turns concepts into (interface) objects

- Interaction instrument
  - Reification of a command into an interface widget
    - Example: scrolling a document -> scrollbar

- Examples
  - Guidelines: reification of alignment
  - Layers: reification of mode

Polymorphism

- Extends commands to multiple object types

- Common examples:
  - Cut, paste, delete, move

- Context-dependent commands

- Homogenous groups
  - If applicable to one object, then applicable to a group of same-type objects

- Heterogeneous groups
  - Applicable to a heterogeneous group if it has meaning for individual object types

Reuse

- Captures interaction patterns for later reuse

- Output reuse
  - Reuse previously created objects
    - Example: duplicate, copy/paste

- Input reuse
  - Reuse previous commands
    - Example: redo, history, macros
Magnetic guidelines

Reification of the alignment command

Power and simplicity
Align command vs Align object:
Align (now) vs Align (and keep aligned)

Multiple shapes
Horizontal, vertical, diagonal, circular, rectangular
Distribute objects

Decomposition
Create / Move / Add object / Remove object

Layers

A mode defines:
Which objects are visible
Which commands are available

Layer = reification of mode
Turn layer on/off
Guidelines, simulation, annotations...

Increased power
Combine layers

Example in CPN2000: debug mode, simulation mode

Groups

Reification + Polymorphism

Group = reification of a selection
Polymorphism:
Apply a command to a group = apply it to each object in the group
Generic commands: Open, Edit, Cut-Copy-Paste

Examples in CPN2000
Folders = Groups of pages
Index = Hierarchy of documents and palettes
Magnetic guidelines = Groups of layout-constrained objects
Styles = Objects that share graphical attributes

Styles

Reification + Output reuse

Style object
Reification of a collection of attributes
Objects that share a style = group
Editing style affects all objects in group

Style picker
Copies any object's current attributes

Style dropper
Applies style to any object
Macros

Input reuse + Reification + Polymorphism

Reuse
Record a sequence of commands as a macro

Polymorphism:
Apply macro as a command in new contexts

Reification:
Edit macro as first class object

Integrating the principles

Reification and polymorphism
More objects and fewer commands

Reification facilitates output reuse
More first-class objects can be reused

Polymorphism facilitates input reuse
Increases the scope of commands

Design principles

Increase simplicity
Reification: direct instruments not indirect commands
Polymorphism: fewer commands
Reuse: copy/redo rather than re-create from scratch

Increase power
Reification: commands as first-class objects
Polymorphism: same command works in multiple contexts
Reuse: path to programming/scripting

Conclusion

Instrumental Interaction makes explicit the artifacts involved in the mediation between user and objects of interest

Descriptive, evaluative and generative model

Design principles help combine power and simplicity