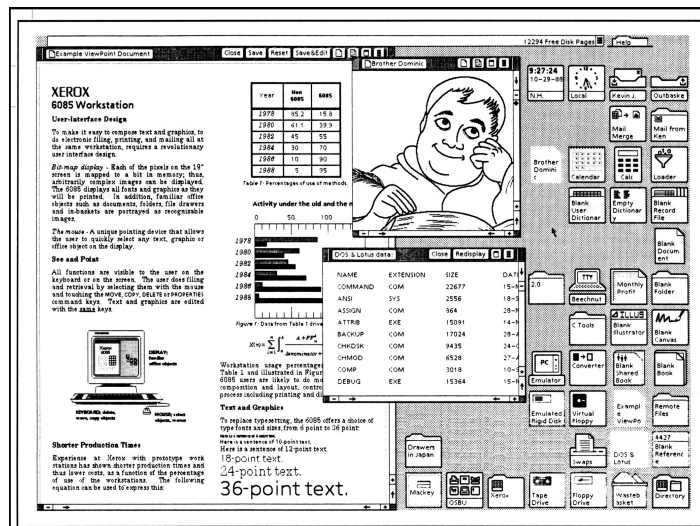


CS477 Reinventing Interactive Systems

Wendy Mackay & Michel Beaudouin-Lafon

INRIA, Université Paris-Sud / Stanford University
mackay@lri.fr, mbl@lri.fr
<http://insitu.lri.fr>

Where do current interfaces come from?



Xerox Star (Xerox PARC, 1981)

What were the underlying assumptions?

Target users: executive secretaries

Application area: office work

One user, one machine: personal computer

What has changed?

Everybody uses computers
for doing many different things
alone or socially

Many people use multiple devices

Everybody is confronted with ever larger amounts of data

... but user interfaces have not changed significantly

How to fix this?

Radical thinking: what if the Xerox Star had not existed?

Break assumptions that are taken for granted:

- windows, applications, files, widgets, ...

Integrate new constraints:

- Massive amounts of data, Diversity of uses and users
- Social use (not just social networks)
- Multiple personal devices, Cloud computing

Create a new interaction model

where interaction is a first-class object

What we will do in this course

Explore the concepts of
instrumental interaction
co-adaptive systems

Read recent work (mostly by us!)

Revisit classic papers

Use design methods:

deconstruction, brainstorming, video prototyping, ...
to create co-adaptive instruments

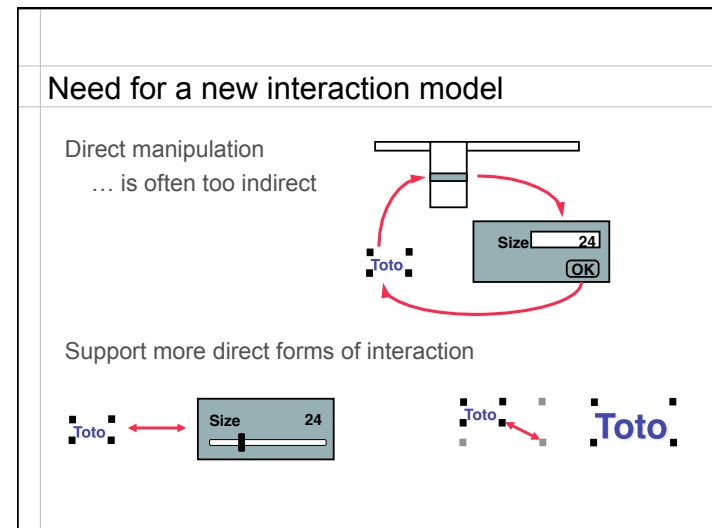
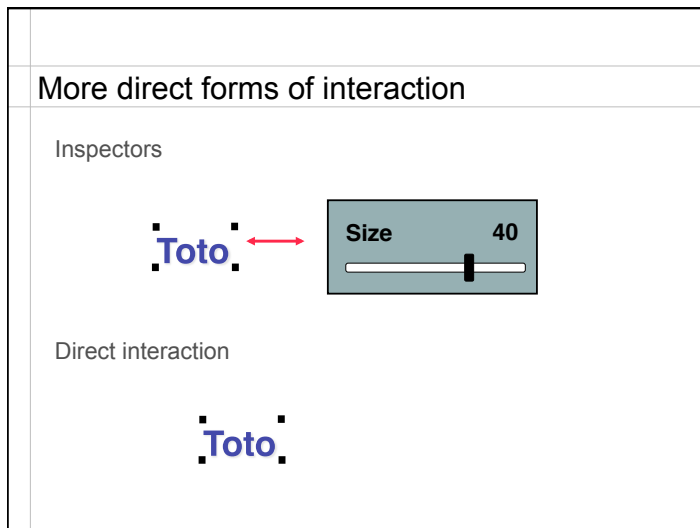
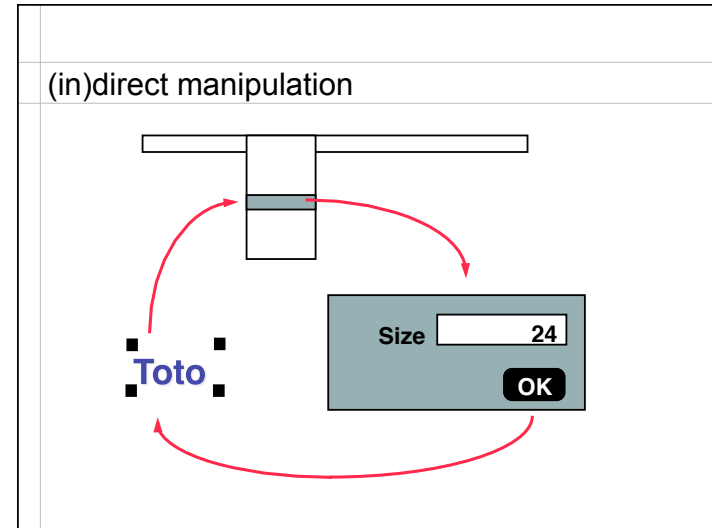
Let's get started: deconstructing interaction

Choosing a color:

- Microsoft Word, Microsoft Excel
- Apple Pages
- Adobe Photoshop, Adobe Illustrator
- Web sites

Class discussion:

- Sharing a document, a photo, a URL, an email address
- Navigating a large document, a list of results, a map



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

Instrumental interaction

Beaudouin-Lafon 97

Inspiration

Interaction with our environment
is mediated by tools and instruments



Two categories of objects

Domain objects

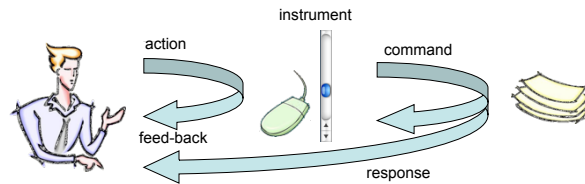


Interaction instruments



Interaction instruments

Conceptual model



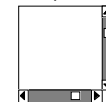
Two levels of interaction: mediation

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



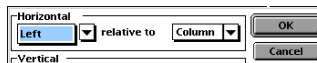
Menus, Toolbars



Scrollbars

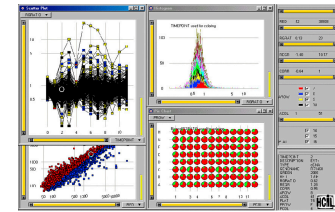


Dialog and Property boxes



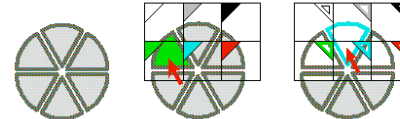
Describing novel interaction techniques

Dynamic Queries



Ahlberg

Toolglasses

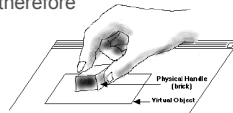


Bier et al.

Describing novel interaction techniques

Tangible interfaces

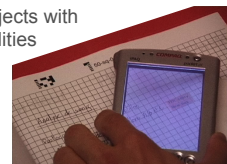
More input devices and therefore more instruments



Fitzmaurice
Ishii
Mackay
Rekimoto
Ullmer

Augmented/Mixed reality

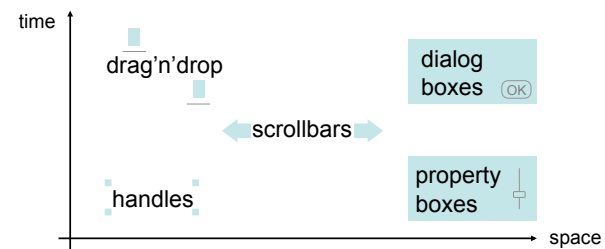
Augmenting physical objects with computational capabilities



Evaluation : Properties of an instrument

Degree of indirection

Spatial offset
Temporal offset



Evaluation : Properties of an instrument

Degree of integration

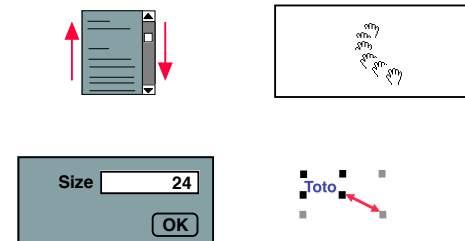
How to use the degrees of freedom of the physical device
Integrity & separability of input devices (Jacob et al., 94)



Evaluation : Properties of an instrument

Degree of conformance

Similarity between physical action and effect on object



Generative power : Three design principles

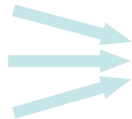
Reification

extends the notion of
what constitutes an object



Polymorphism

extends the power of commands
with respect to these objects



Reuse

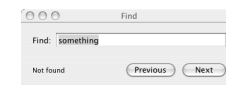
provides a way of capturing and
reusing interaction patterns



Example : text search instrument

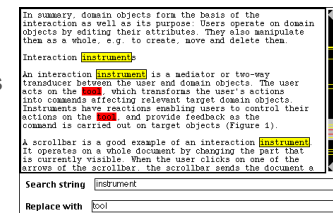
Classic search:

Sequential
Modal

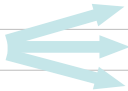


Search instrument:

Show all occurrences
Allow replacing occurrences
in any order




Reification



Turns concepts into (interface) objects

Interaction instrument

Reification of a command into an interface widget



Example :

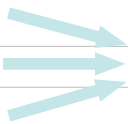
scrolling a document -> scrollbar

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

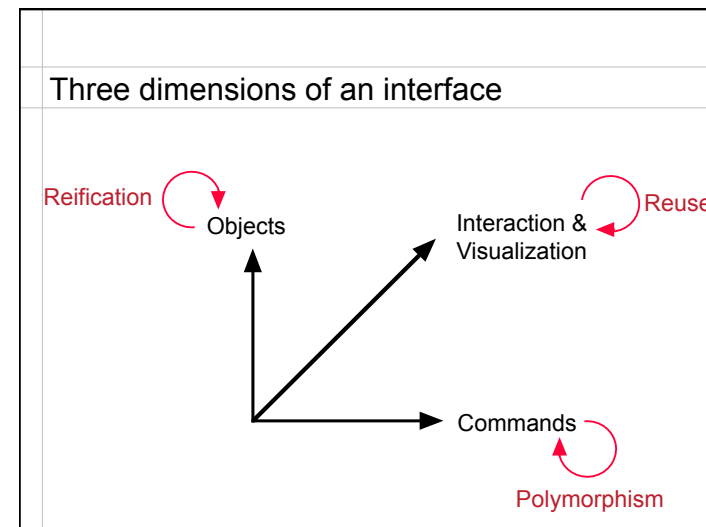
Reuse previously created objects

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

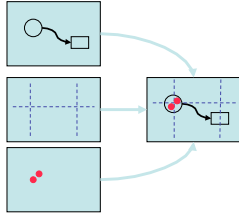
Layers

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

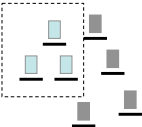
Layer = reification of mode
Turn layer on/off
Guidelines, simulation, annotations...

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