Instrumental Interaction

Michel Beaudouin-Lafon
Laboratoire Interdisciplinaire des Sciences du Numérique (LISN)
Université Paris-Saclay / CNRS
mbl@lisn.fr
http://ex-situ.lri.fr

Overview

Analysis of WIMP applications

Power vs. Simplicity

Interaction model

Instrumental Interaction

Design Principles

Analysis of WIMP interfaces

Analysis of WIMP applications

#menus	Menus in menu bar
#cmds	Commands in menus
#dlogs	Commands that lead to a dialog box
#smenus	Sub-menus
#scmds	Commands in sub-menus
#sdlogs	Commands in sub-menus that lead to a dialog box
Tcmds	Total commands: #cmds - #smenus + #scmds
Tdlogs	Total dialog boxes: #dlogs + #sdlogs
Cmds/M	Mean commands per menu: #cmds / #menus
Cmds/SM	Mean commands per sub-menu: #scmds / #smenu
#palettes	Palettes and toolbars
#tools	Widgets in palettes and toolbars
#prefs	Preference pages
#options	Options in preference pages
macros	Whether macros can be defined

Number of commands

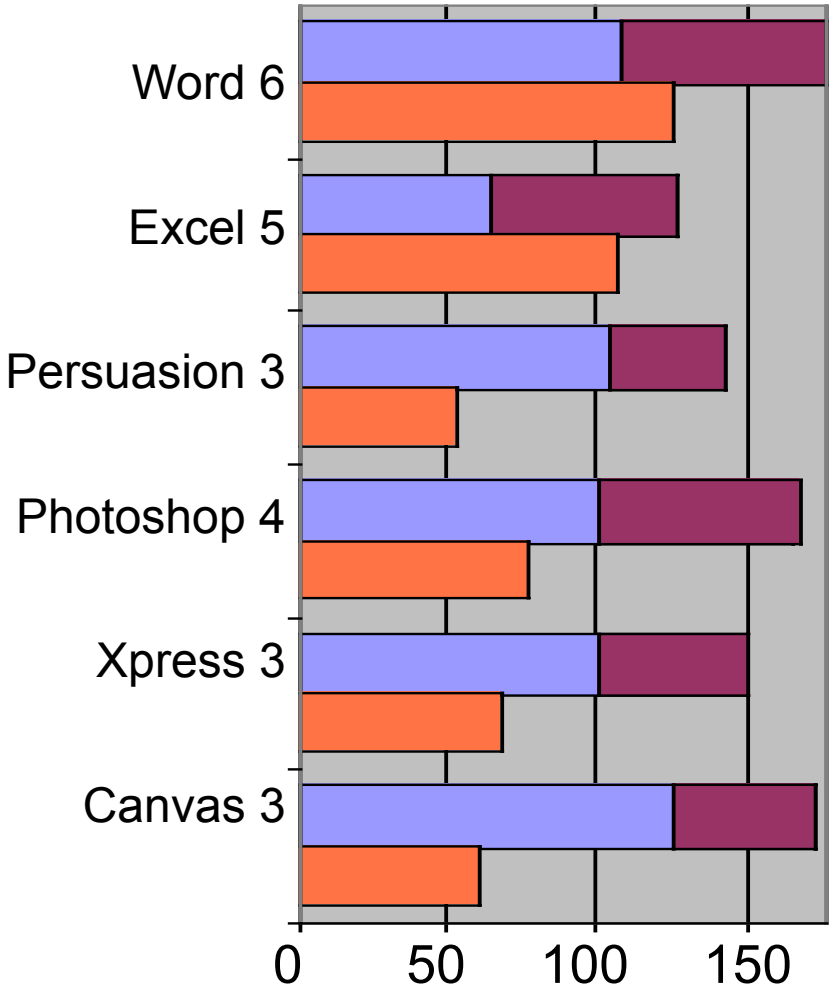
Word6 Excel5 Persuasion3 Photoshop4 Xpress3 Canvas3

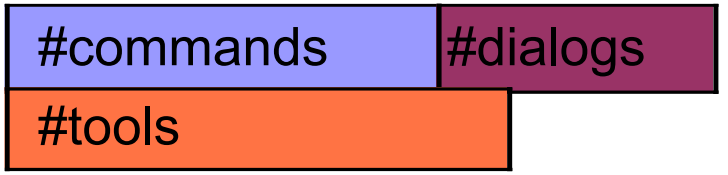
Criteria	W6	E5	Pe3	P4	X3	C3	Avg	s
#menus	8	8	7	8	7	8	7.7	0.5
#cmds	106	84	97	111	99	74	95.2	13.8
#dlog	69	44	20	27	40	21	36.8	18.6
#smenu	1	15	27	26	13	22	17.3	9.8
#scmds	3	58	73	82	65	121	67.0	38.4
#sdlog	0	20	20	40	10	28	19.7	13.9
Tcmds	108	127	143	167	151	173	144.8	24.5
Tdlogs	69	64	40	67	50	49	56.5	11.8
Cmds/M	13.3	10.5	13.9	13.9	14.1	9.3	12.5	2.1
Cmds/SM	3.0	3.9	2.7	3.2	5.0	5.5	3.9	1.1
#palettes	9	13	5	11	6	6	8.3	3.2
#tools	125	106	54	77	68	60	81.7	28.0
#prefs	12	10	1	8	5	11	7.8	4.2
#options	113	76	11	51	82	27	60.0	37.7
macros	yes	yes	no	yes	no	yes		

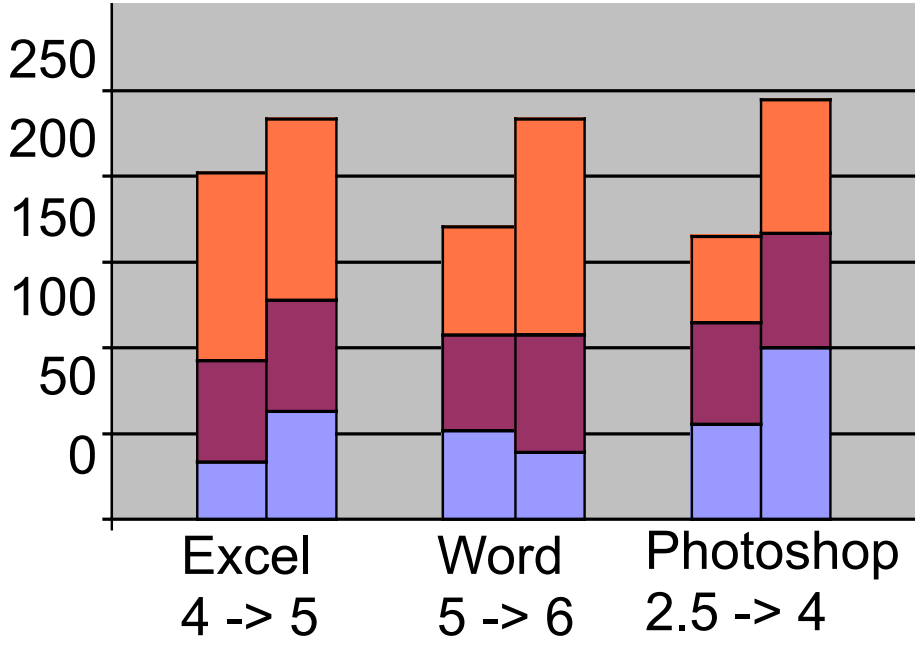
Successive versions

	Excel 4->5			Word 5->6			Photoshop 2.5->4		
Criteria	**E4**	**E5**	**%**	**W5**	**W6**	**%**	**P2**	**P4**	**%**
#menus	8	8	0%	8	8	0%	7	8	+14%
#cmds	93	84	-10%	107	106	-1%	78	111	+42%
#dlog	60	44	-27%	55	69	+25%	21	27	+29%
#smenu	0	15	+ •	0	1	+ •	19	26	+37%
#scmds	0	58	+ •	0	3	+ •	56	82	+46%
#sdlog	0	20	+ •	0	0	+ •	39	40	+3%
Tcmds	93	127	+37%	107	108	+1%	115	167	+45%
Tdlogs	60	64	+7%	55	69	+25%	60	67	+12%
Cmds/M	11.6	10.5	-10%	13.4	13.3	-1%	11.1	13.9	+25%
Cmds/SM	0	3.9	+ •	0	3	+ •	2.9	3.2	+7%
#palettes	8	13	+63%	3	9	+200%	6	11	+83%
#tools	108	106	-2%	63	125	+98%	49	77	+57%
#prefs	0	10	+ •	10	12	+20%	9	8	-11%
#options	0	76	+ •	52	113	+117%	58	51	-12%
macros	yes	yes		no	yes		no	yes	

Analysis of WIMP applications





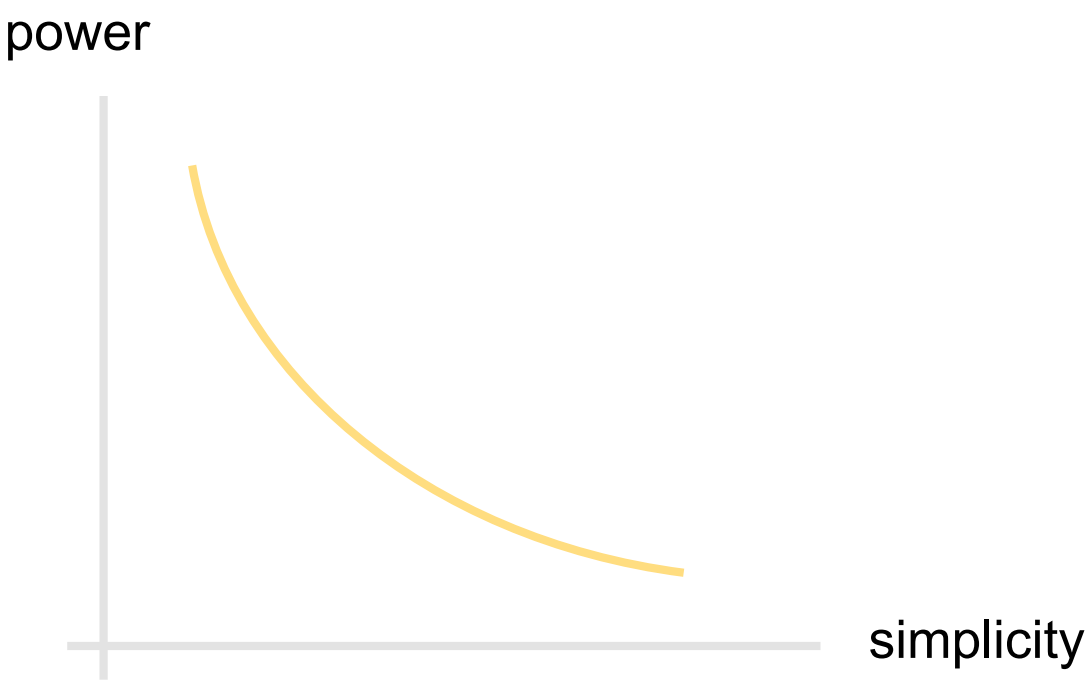


Power vs. Simplicity

Simple things should be simple

Complex things should be possible

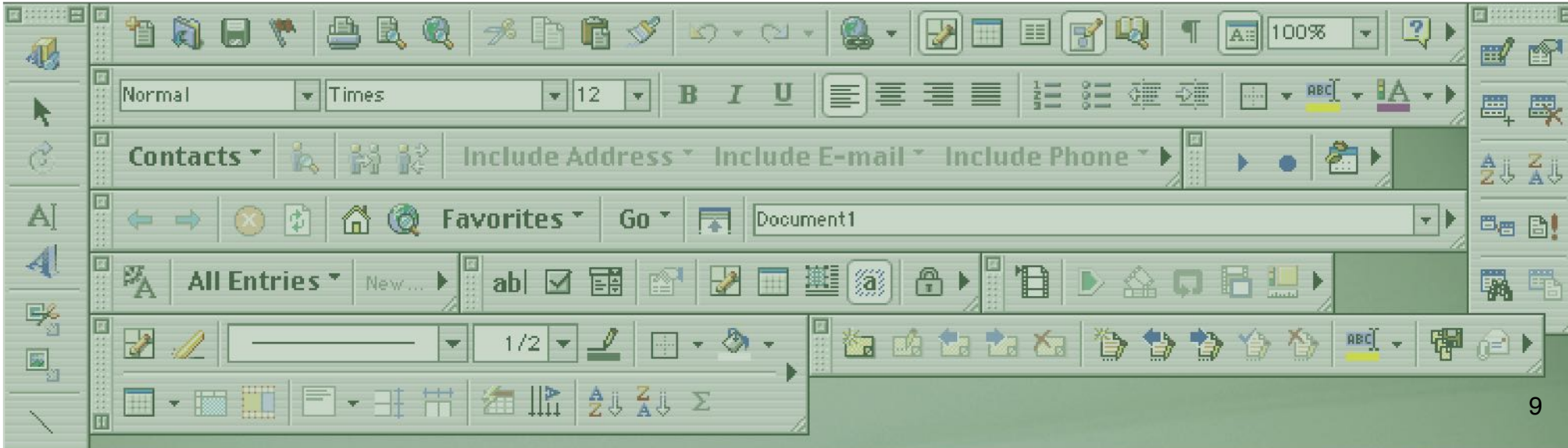
How to combine power & simplicity ?



More is less: the illusion of power

Bloatware

- Too many functions
- More functions with each new version



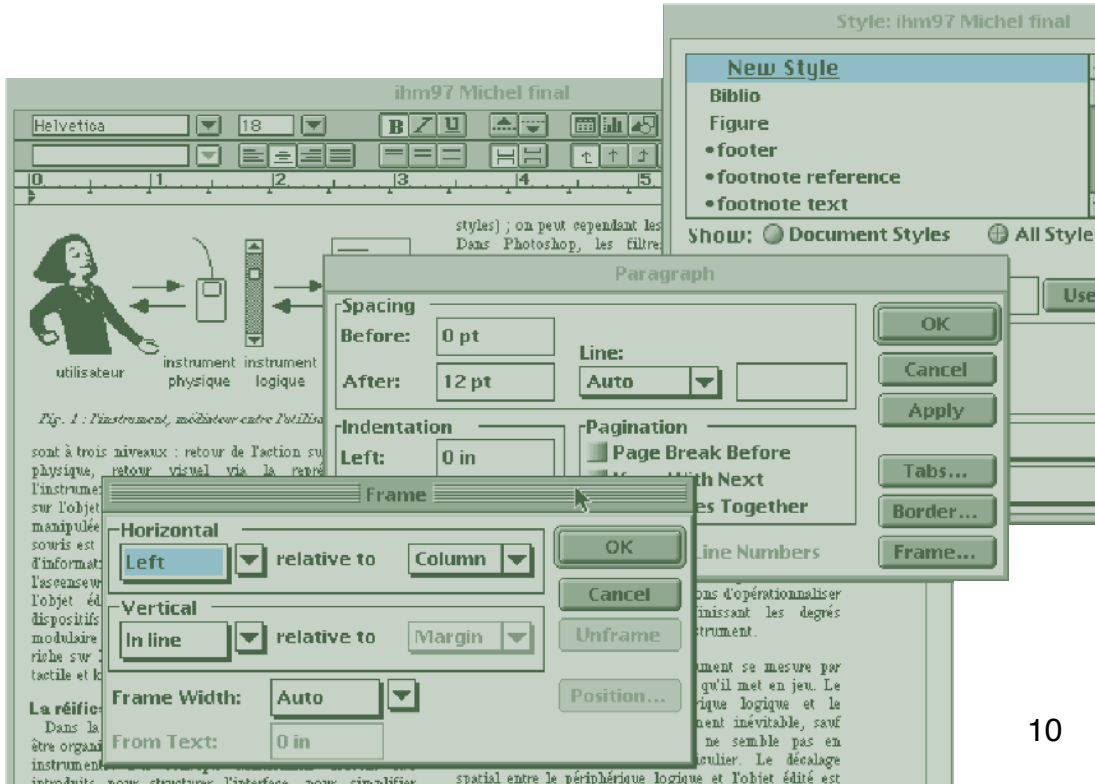
Marketing software : increased power?

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?

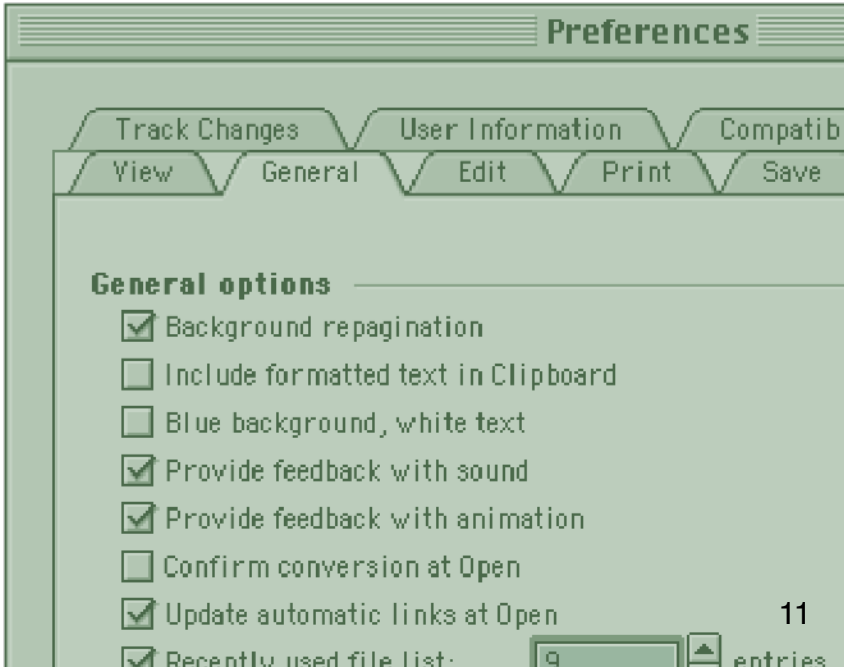
Add wizards

- 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

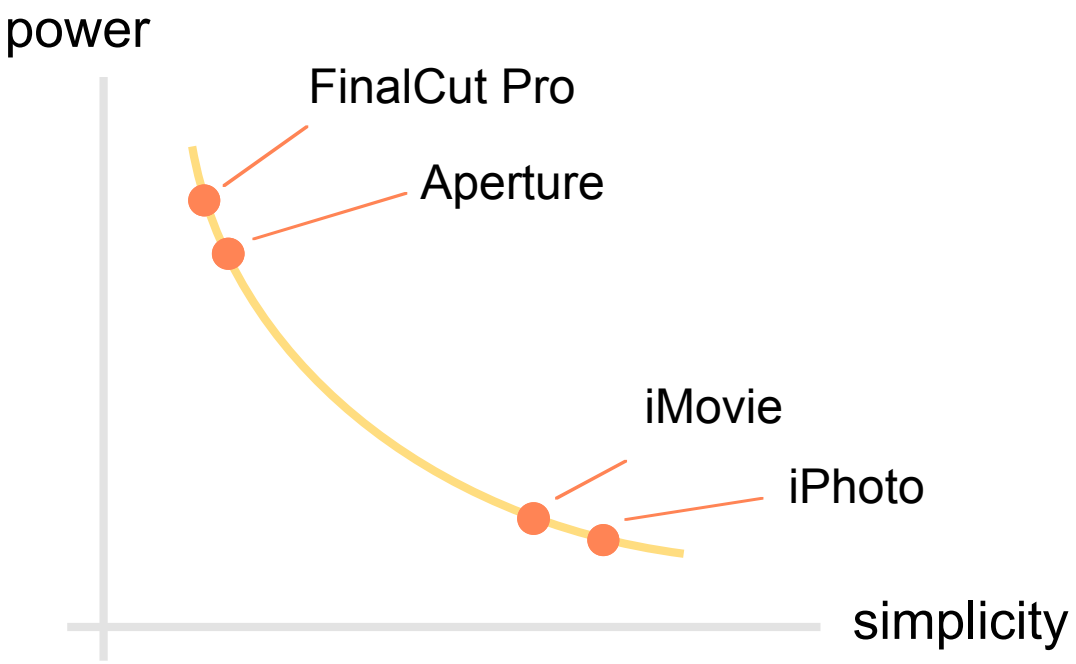
- Cognitive: more decisions
- Sensory-motor: more steps

Power vs. simplicity

A better approach

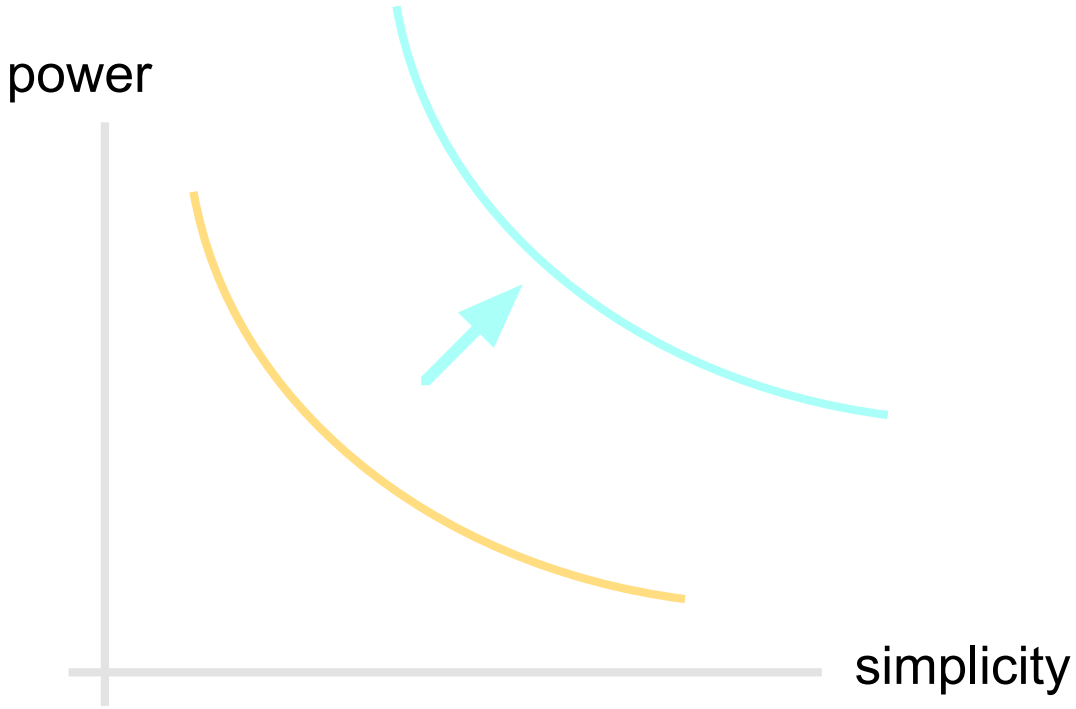
Specializing software

Example: Apple Macintosh

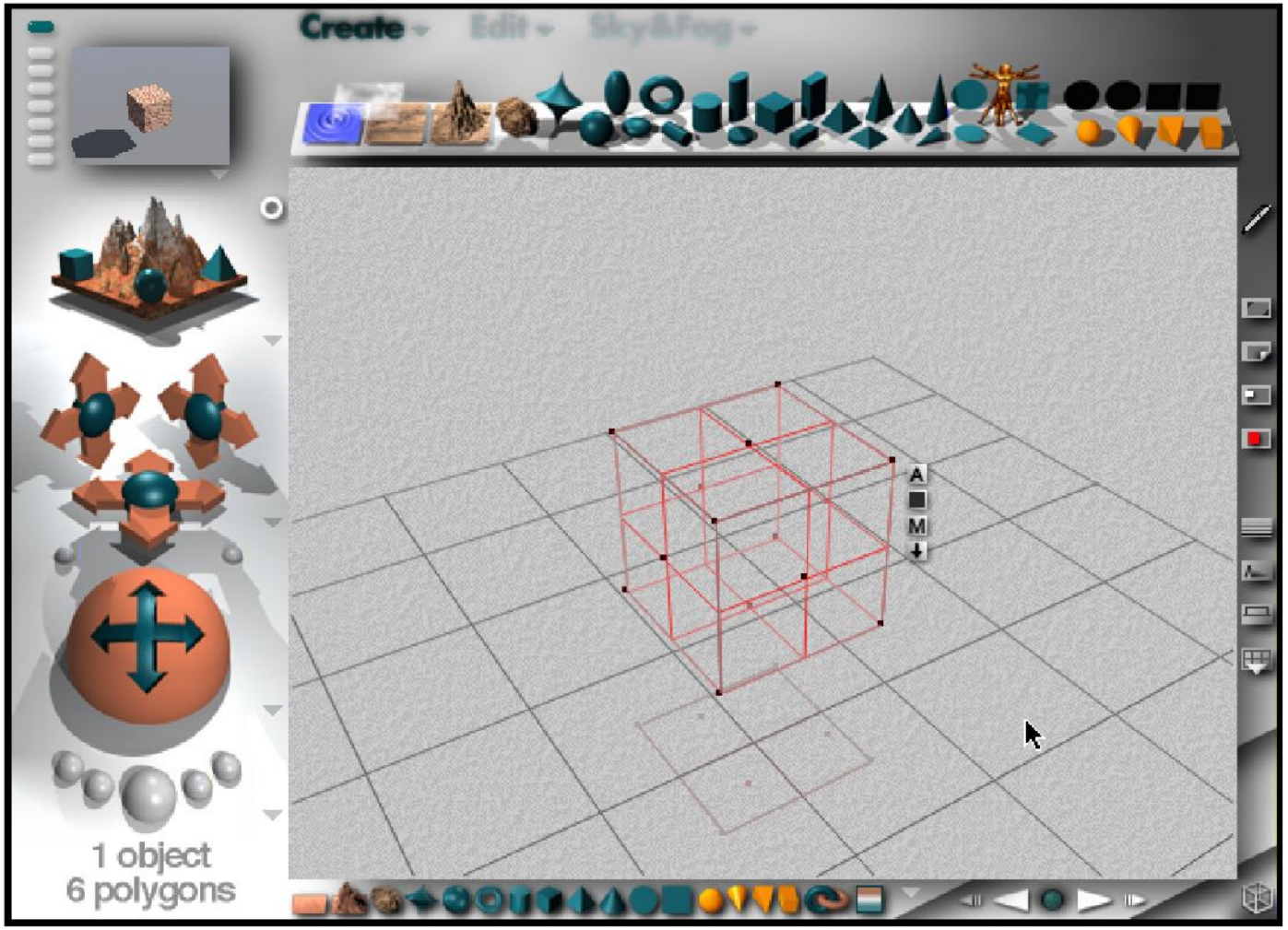


Another approach

Shifting the curve

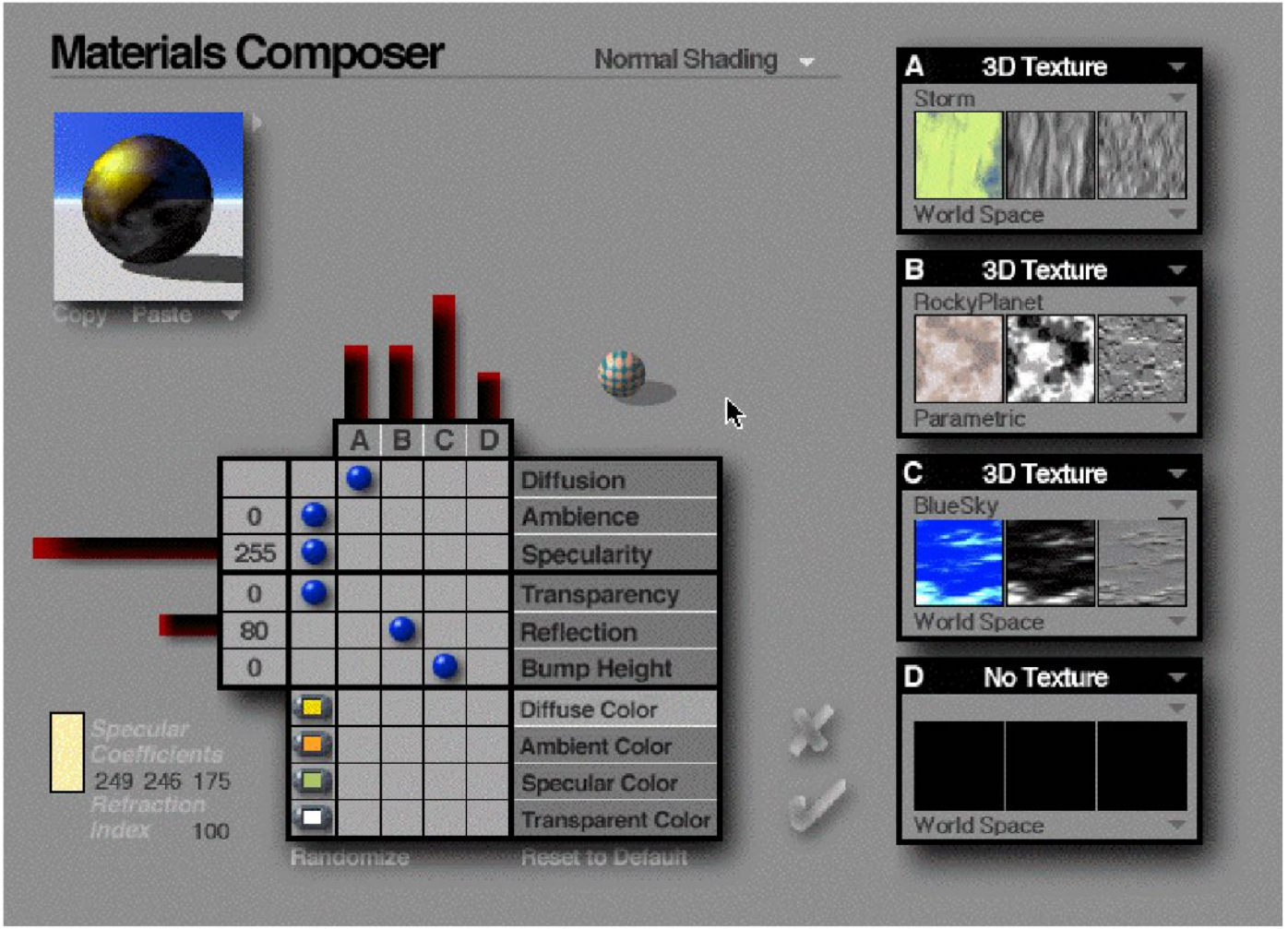


Going beyond WIMP



Bryce2
(Metatools)

Complexity can be simple



Bryce2
(Metatools)

Comparison: Bryce vs WIMP

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

No menus,
No windows,
No dialog boxes

Graphical design
Interaction design
Layered approach

Case study: CPN 2000 Project

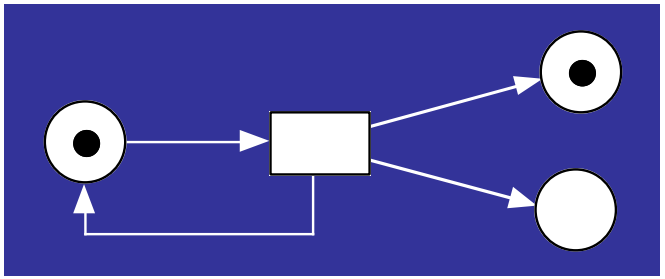
Beaudouin-Lafon & Mackay, 2000

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:

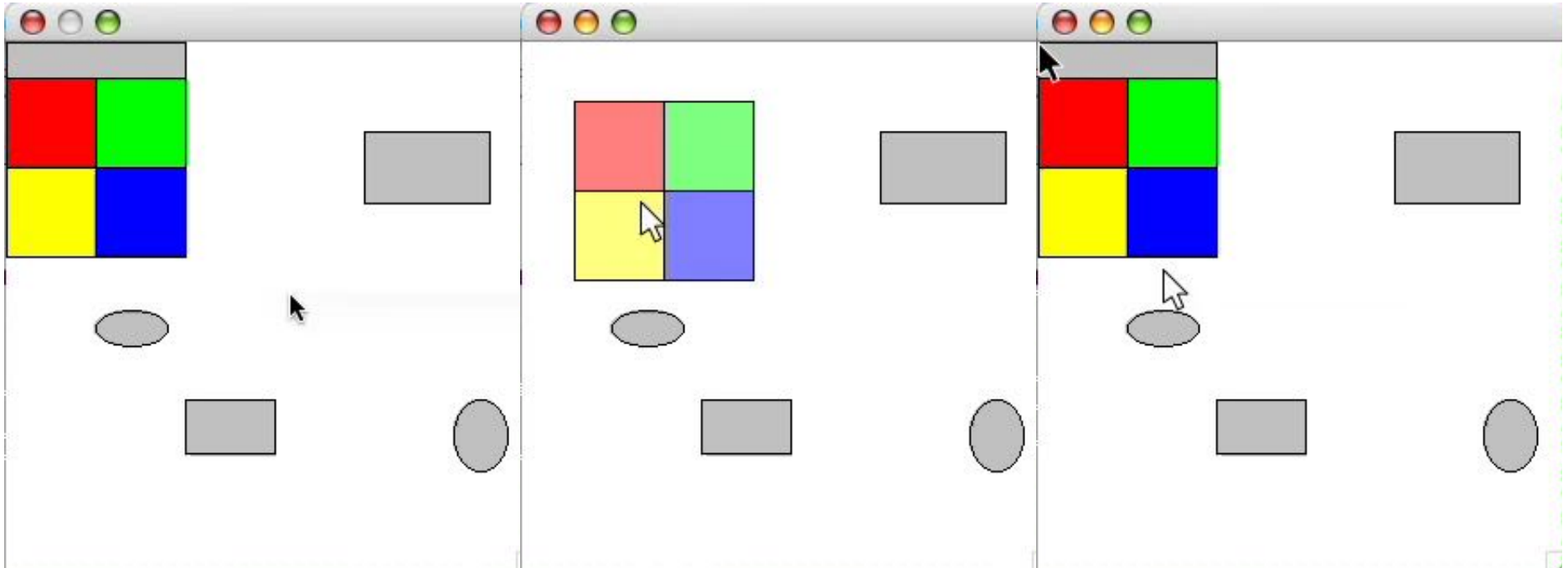
- Toolglasses
- Contextual menus
- Floating palettes
- Bi-manual interaction

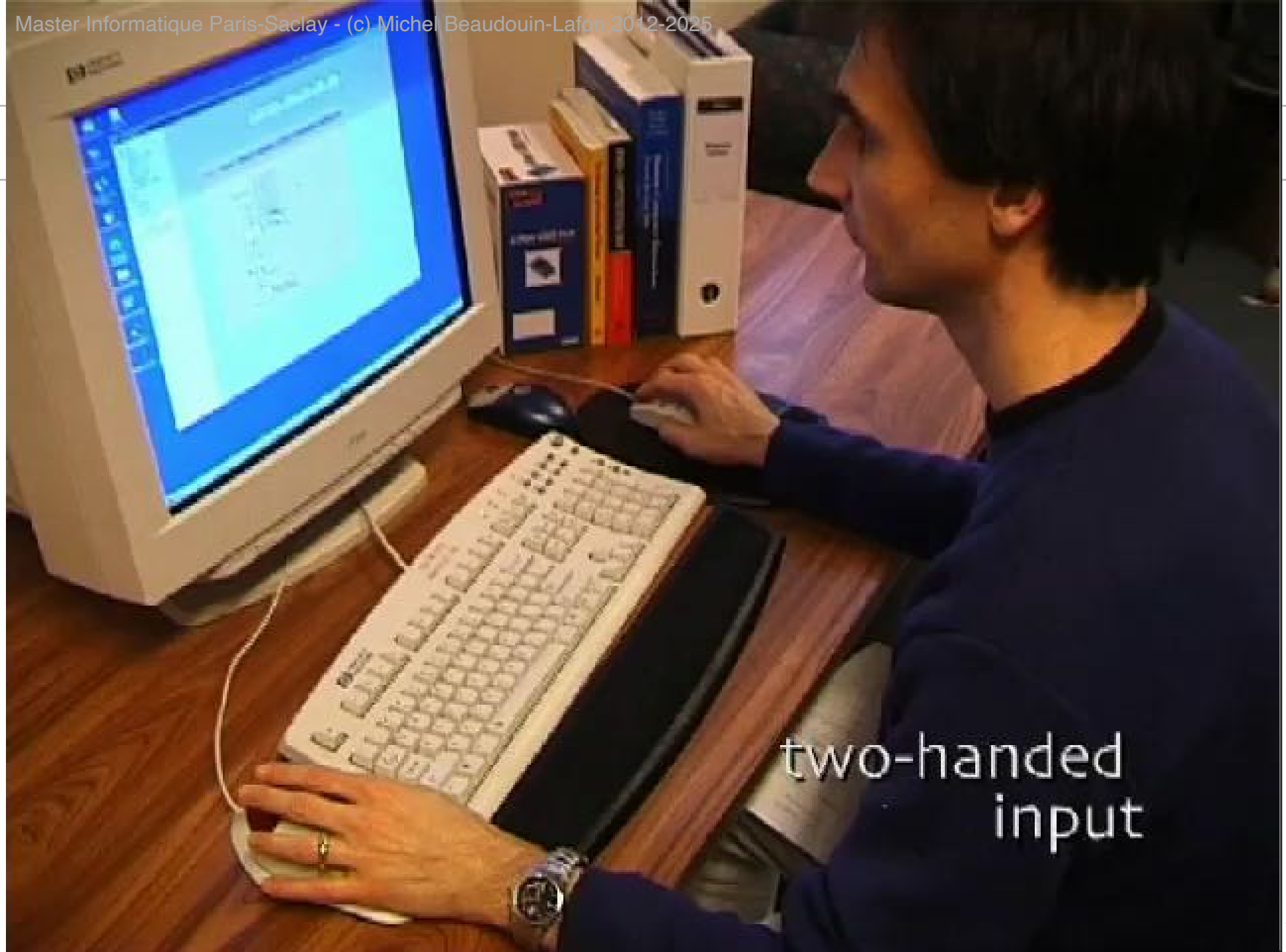
Why these techniques?

- User studies show context affects tool preference
- Palettes: focus on command
- Marking menus: focus on object
- Toolglasses: mixed focus

Three types of palettes

Tool palette	Toolglass	Bimanual palette
Unimanual	Bimanual	Bimanual
Back-and-forth	Click-through	2 cursors




two-handed
input

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

New Interaction Model: Instrumental Interaction

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

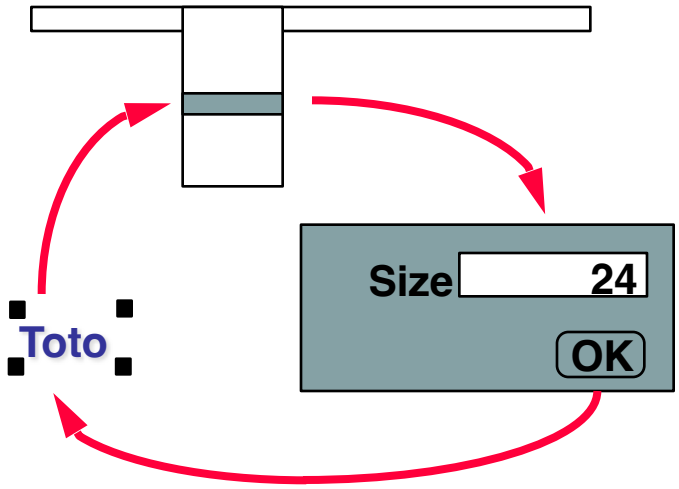
Evaluative:
helps evaluate interactive systems

Generative:
helps create new interaction techniques

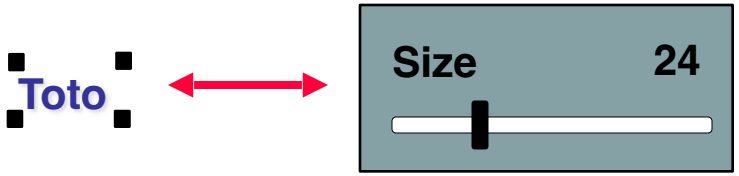
Need for a new interaction model

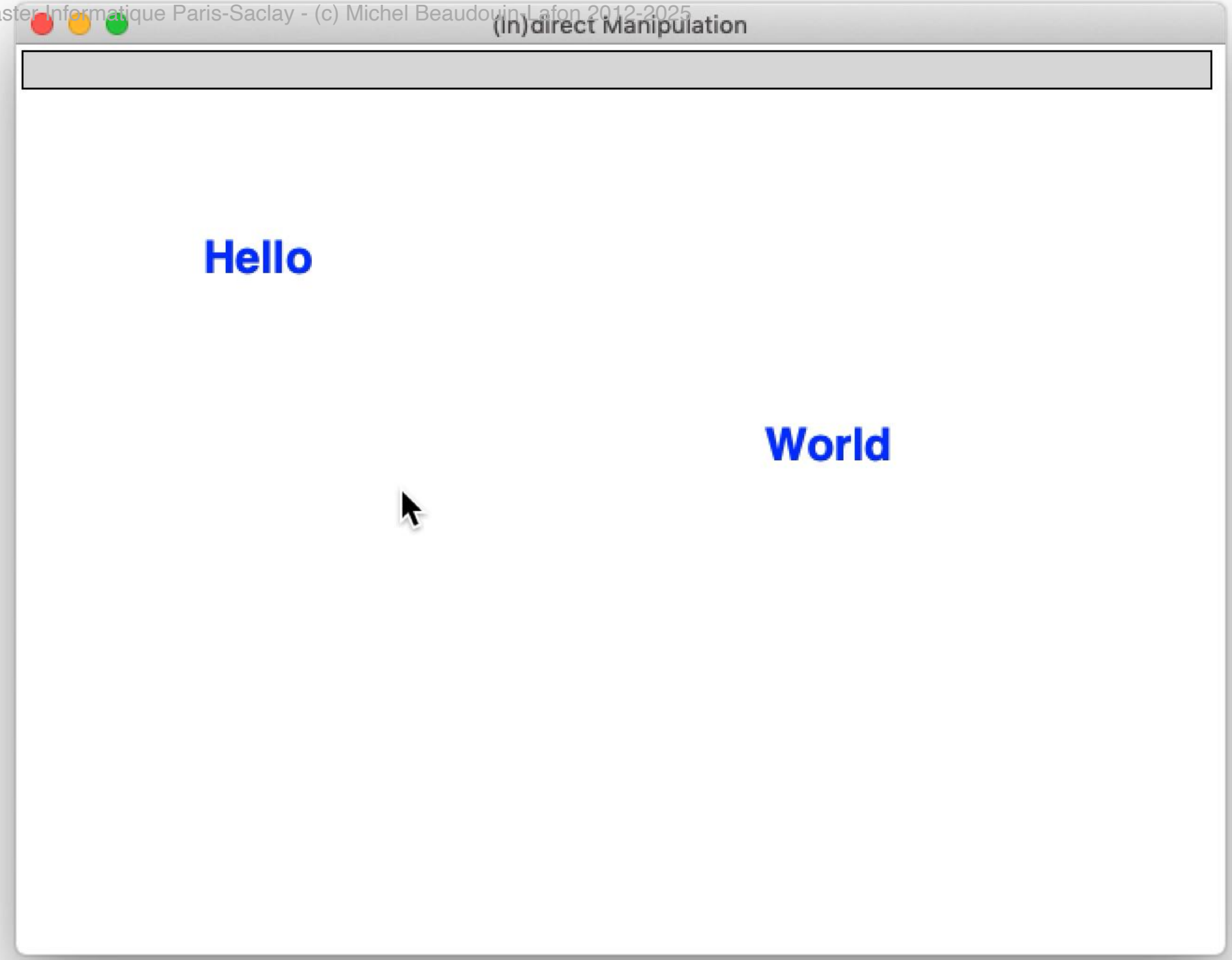
Direct manipulation

… is often too indirect



Support more direct forms of interaction



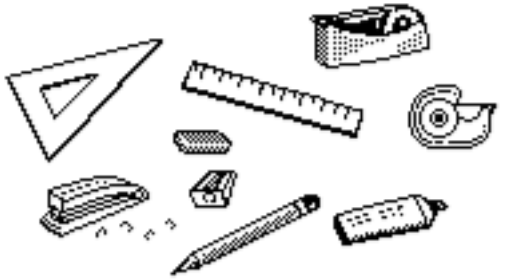

(in)direct Manipulation
Hello
World

Instrumental interaction

Beaudouin-Lafon 97

Inspiration

Interaction with our environment
is mediated by tools and instruments



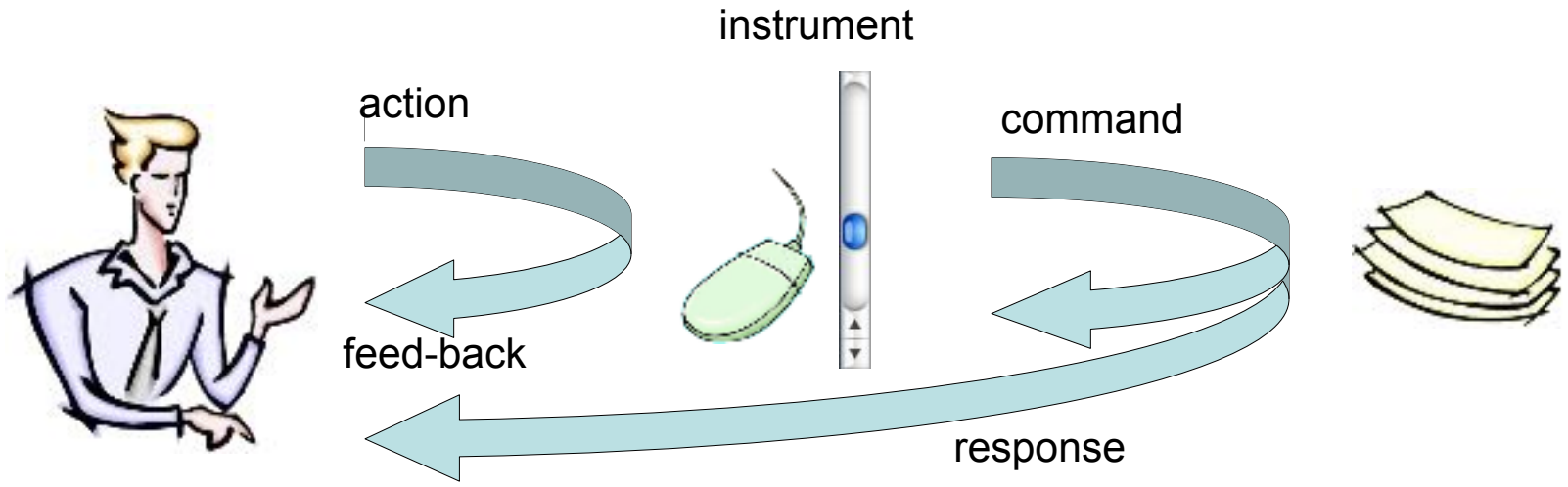
Two categories of objects

Domain objects

Interaction instruments

Interaction instruments

Conceptual model



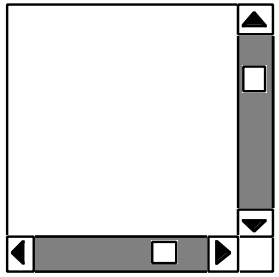
Two levels of interaction: mediation

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

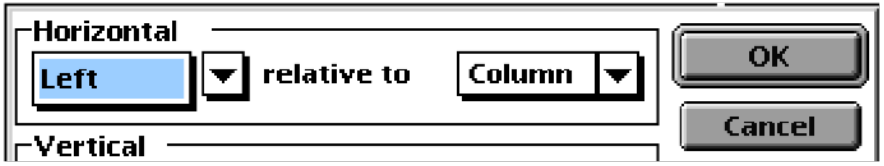
Instruments of (in)direct manipulation

Handles, Title bars

Menus, Toolbars

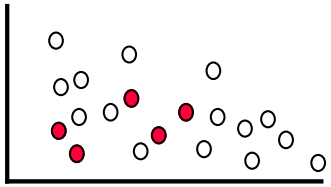
Scrollbars

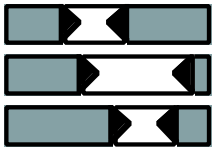
Dialog and Property boxes



Describing novel interaction techniques

Dynamic Queries



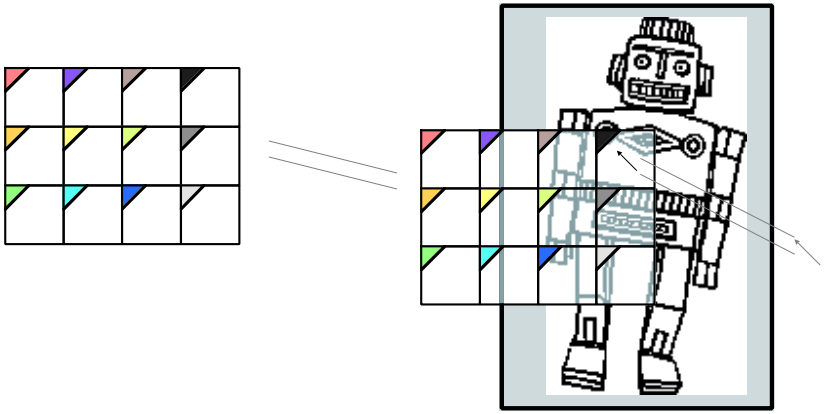


Ahlberg

Dropable Tools

Bederson et al.

Toolglasses

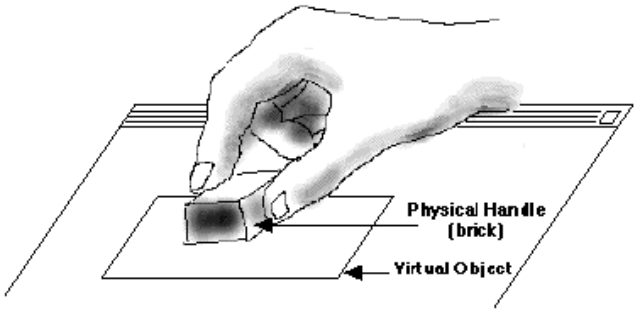


Bier et al.

Describing novel interaction techniques

Tangible interfaces

More input devices and therefore more instruments

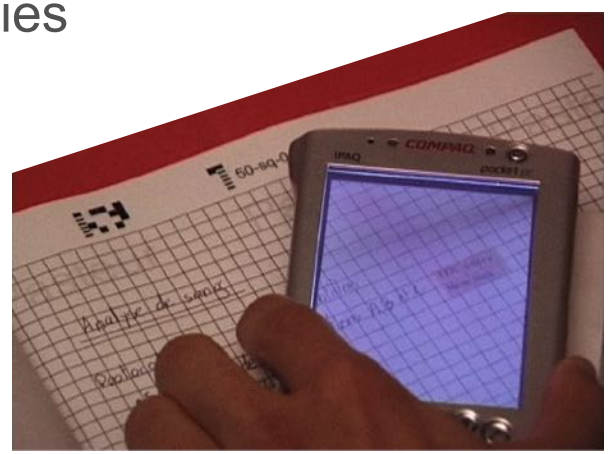


Fitzmaurice
Ishii

Augmented/Mixed reality

Augmenting physical objects with computational capabilities

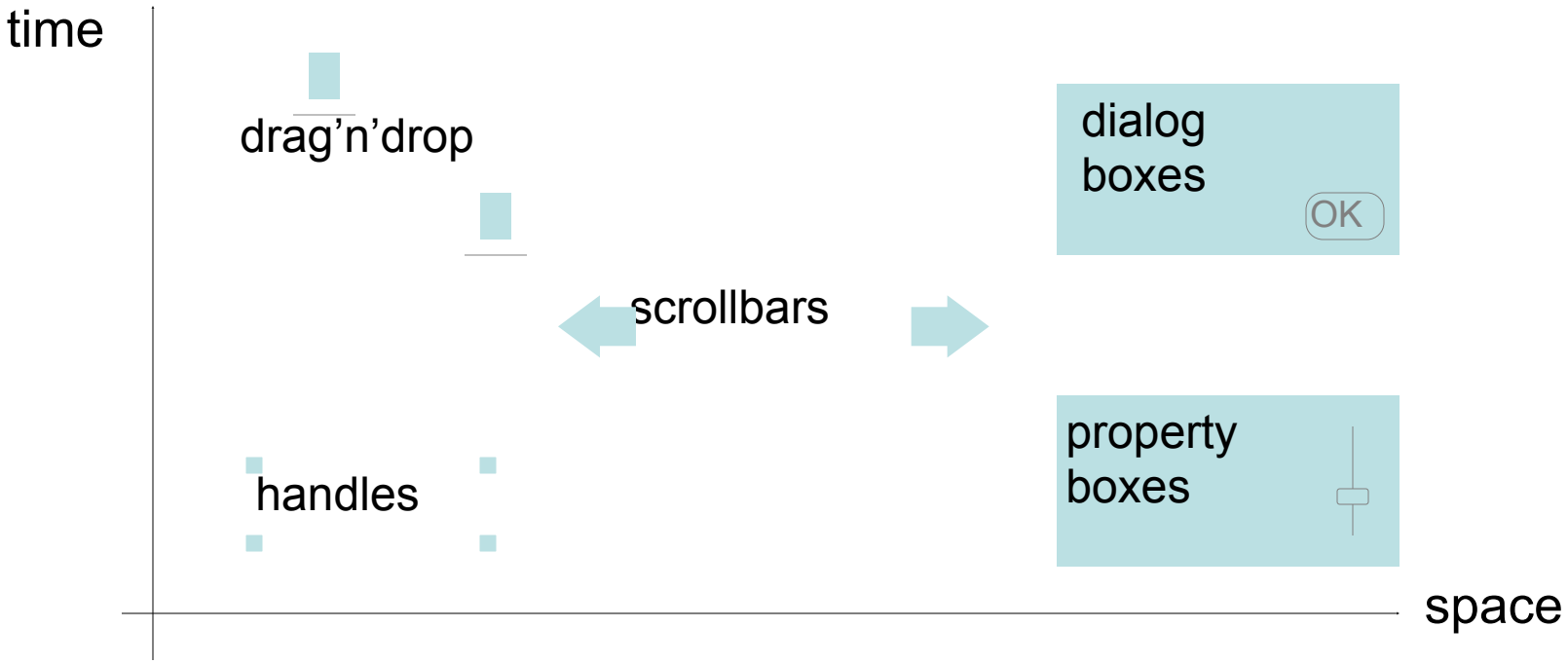
Mackay
Rekimoto
Ullmer



Evaluation : Properties of an instrument

Degree of indirection

- Spatial offset
- Temporal offset



Evaluation : Properties of an instrument

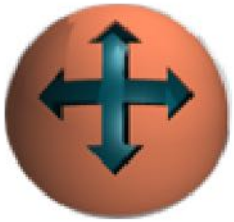
Degree of integration

How to use the degrees of freedom of the physical device

Integrality & separability of input devices (Jacob et al., 94)

2->1

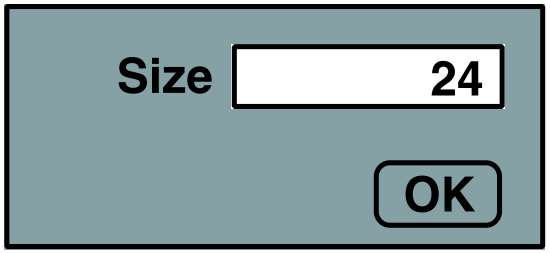
2->3

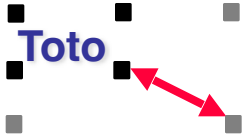


Evaluation : Properties of an instrument

Degree of conformance

Similarity between physical action and effect on object





Design Principles

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

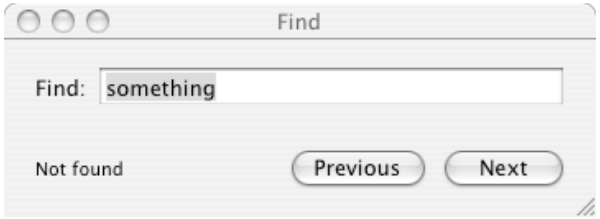
Reuse

provides a way of capturing and
reusing interaction patterns

Example : text search instrument

Classic search:

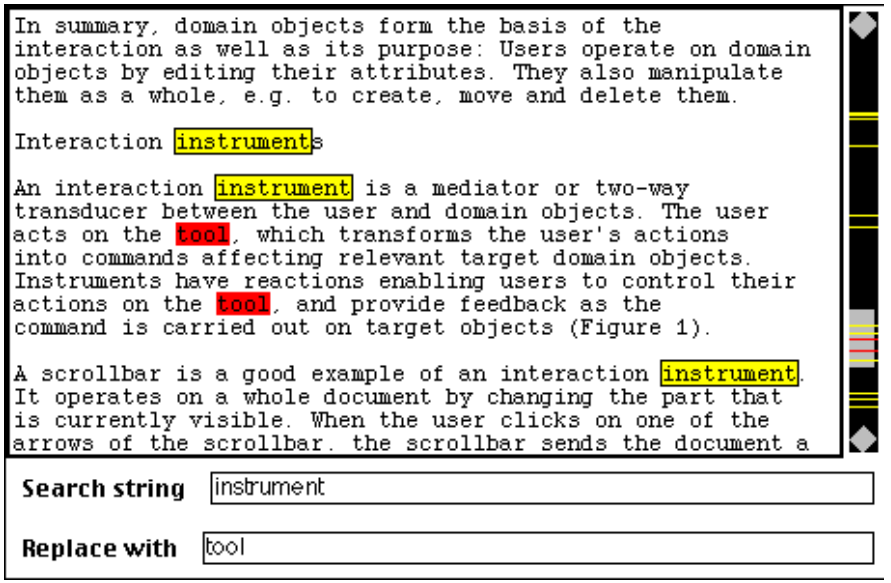
- Sequential
- Modal



Search instrument:

- Show all occurences
- Allow replacing occurences in any order

Augmented scrollbar



Editor

Instrumental Interaction: An Interaction Model for
Designing Post-WIMP User Interfaces

Michel Beaudouin-Lafon

Dept of Computer Science
University of Aarhus

Aabogade 34
DK-8200 Aarhus N - Denmark
mbl@daimi.au.dk

ABSTRACT
This article introduces a new interaction model called
Instrumental Interaction that extends and generalizes the
principles of direct manipulation. It covers existing
interaction styles, including traditional WIMP interfaces, as
well as new interaction styles such as two-handed input and
augmented reality. It defines a design space for new
interaction techniques and a set of properties for comparing
them. Instrumental Interaction describes graphical user
interfaces in terms of domain objects and interaction
instruments. Interaction between users and domain objects
is mediated by interaction instruments, similar to the tools
and instruments we use in the real world to interact with
physical objects. The article presents the model, applies it
to describe and compare a number of interaction techniques,
and shows how it was used to create a new interface for
searching and replacing text.

Keywords
Interaction model, WIMP interfaces, direct manipulation,
post-WIMP interfaces, instrumental interaction

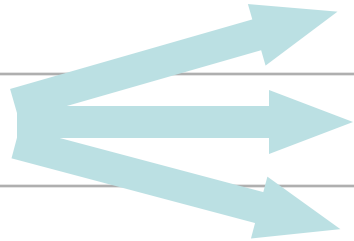
INTRODUCTION

In the early eighties, the Xerox Star user interface [27] and
the principles of direct manipulation [26] led to a powerful
graphical user interface model, referred to as WIMP
(Windows, Icons, Menus and Pointing). WIMP interfaces

Search string

Replace with

Reification



Turns concepts into (interface) objects

Interaction instrument

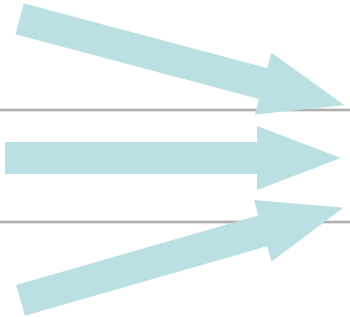
- Reification of a command into an interface widget

- 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

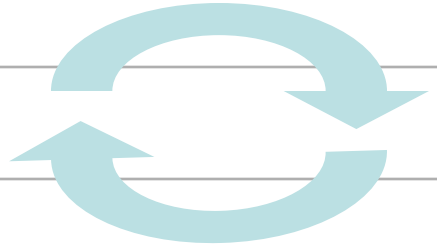
Homogenous groups

- 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

Examples

Magnetic guidelines

Reification of the alignment command

Power and simplicity

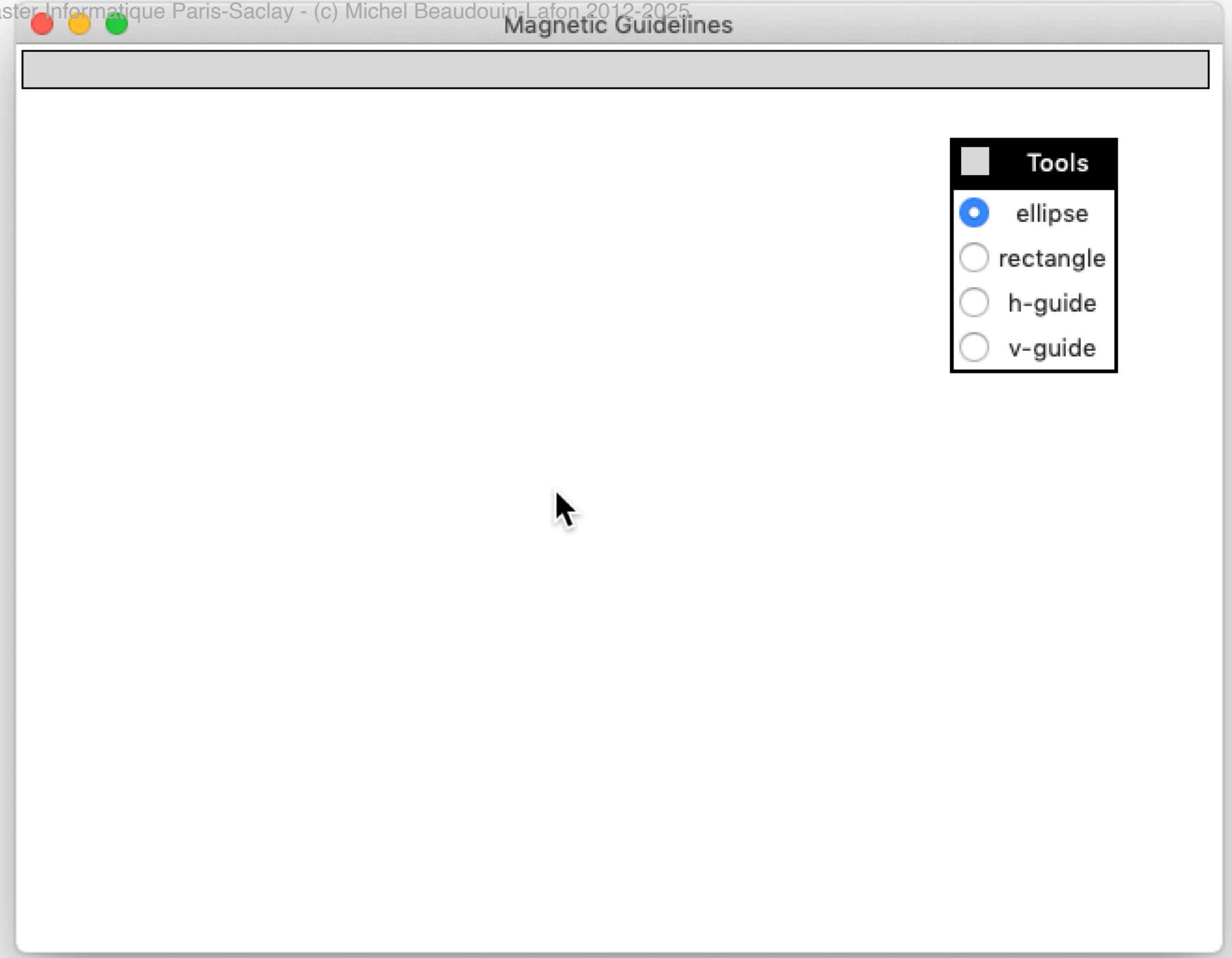
- Align command vs Align object:
- Align (now) vs Align (and keep aligned)

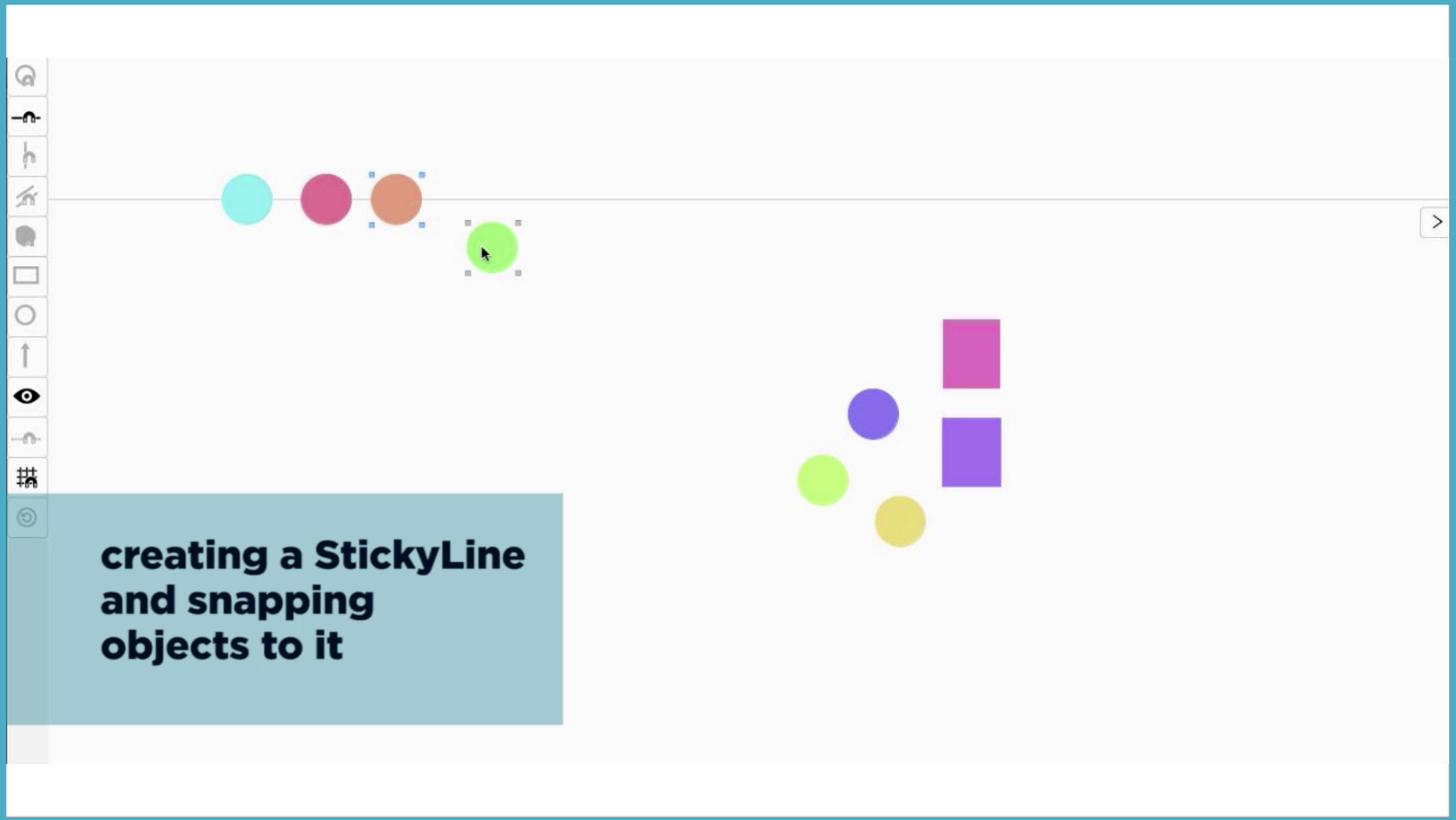
Multiple shapes

- Horizontal, vertical, diagonal, circular, rectangular
- Distribute objects

Decomposition

- Create / Move / Add object / Remove object


Magnetic Guidelines
Tools
ellipse
rectangle
h-guide
v-guide



StickyLines - Ciolfi, Maudet, Mackay & Beaudouin-Lafon, 2016

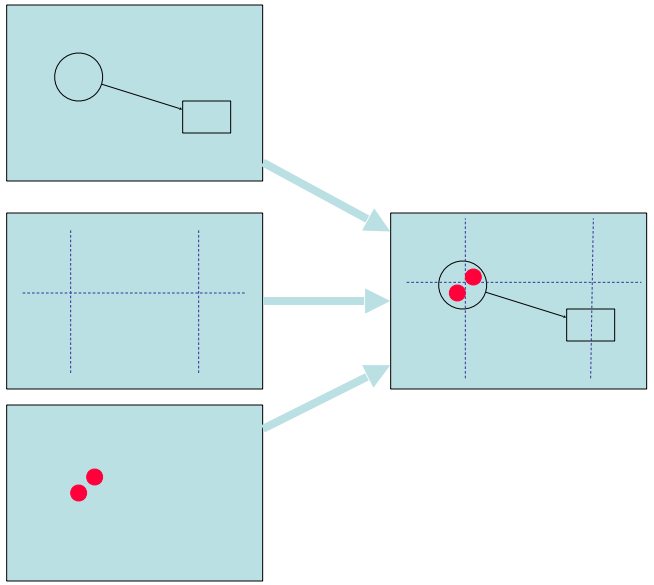
Layers

A mode defines:

- Which objects are visible
- Which commands are available

Layer = reification of mode

- Turn layer on/off
- Guidelines, simulation, annotations...



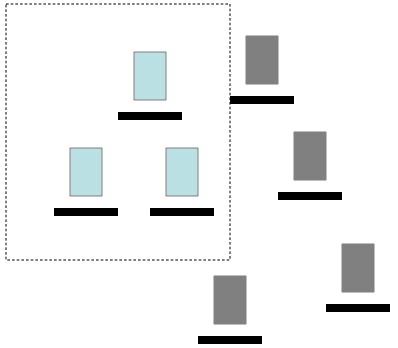
Increased power

- 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

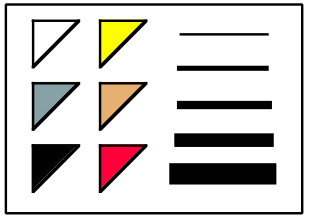
- Folders = Groups of pages
- Index = Hierarchy of documents and palettes
- Magnetic guidelines = Groups of layout-constrained objects
- Styles = Objects that share graphical attributes

Styles

Reification + Output reuse

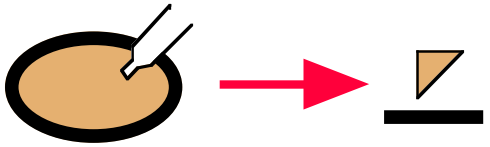
Style object

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



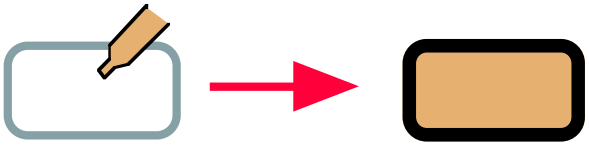
Style picker

- 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

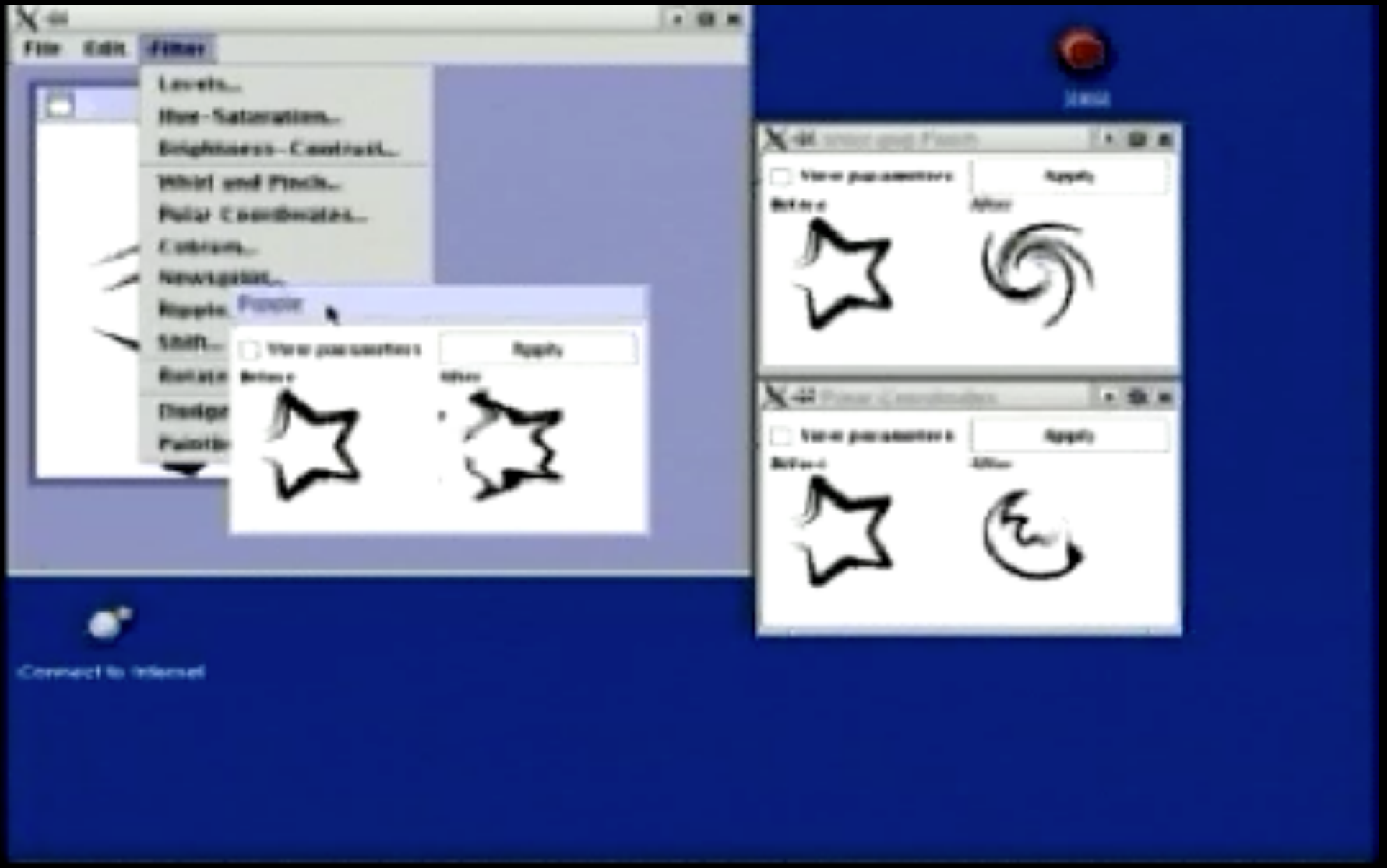
More objects and fewer commands

Reification facilitates output reuse

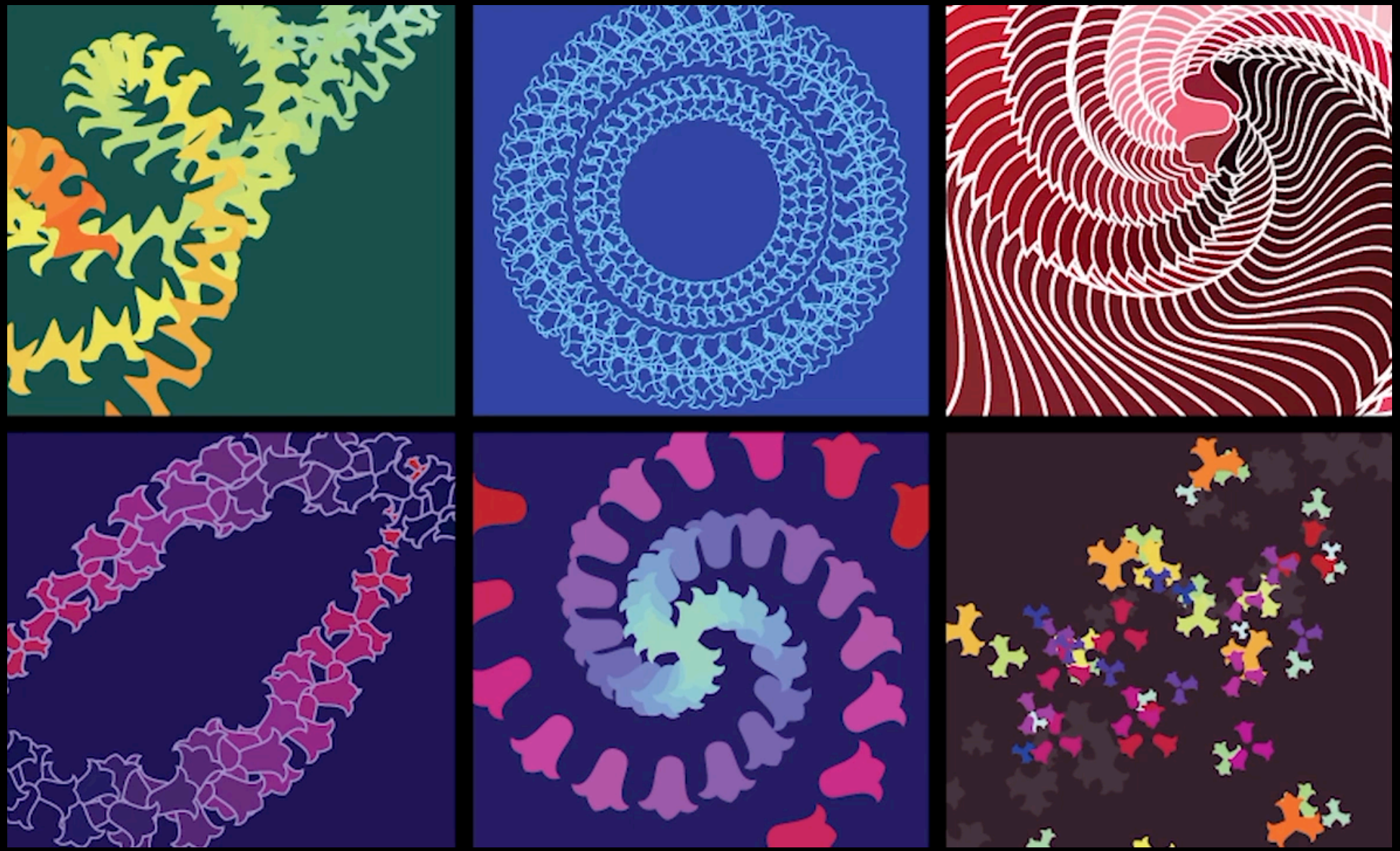
More first-class objects can be reused

Polymorphism facilitates input reuse

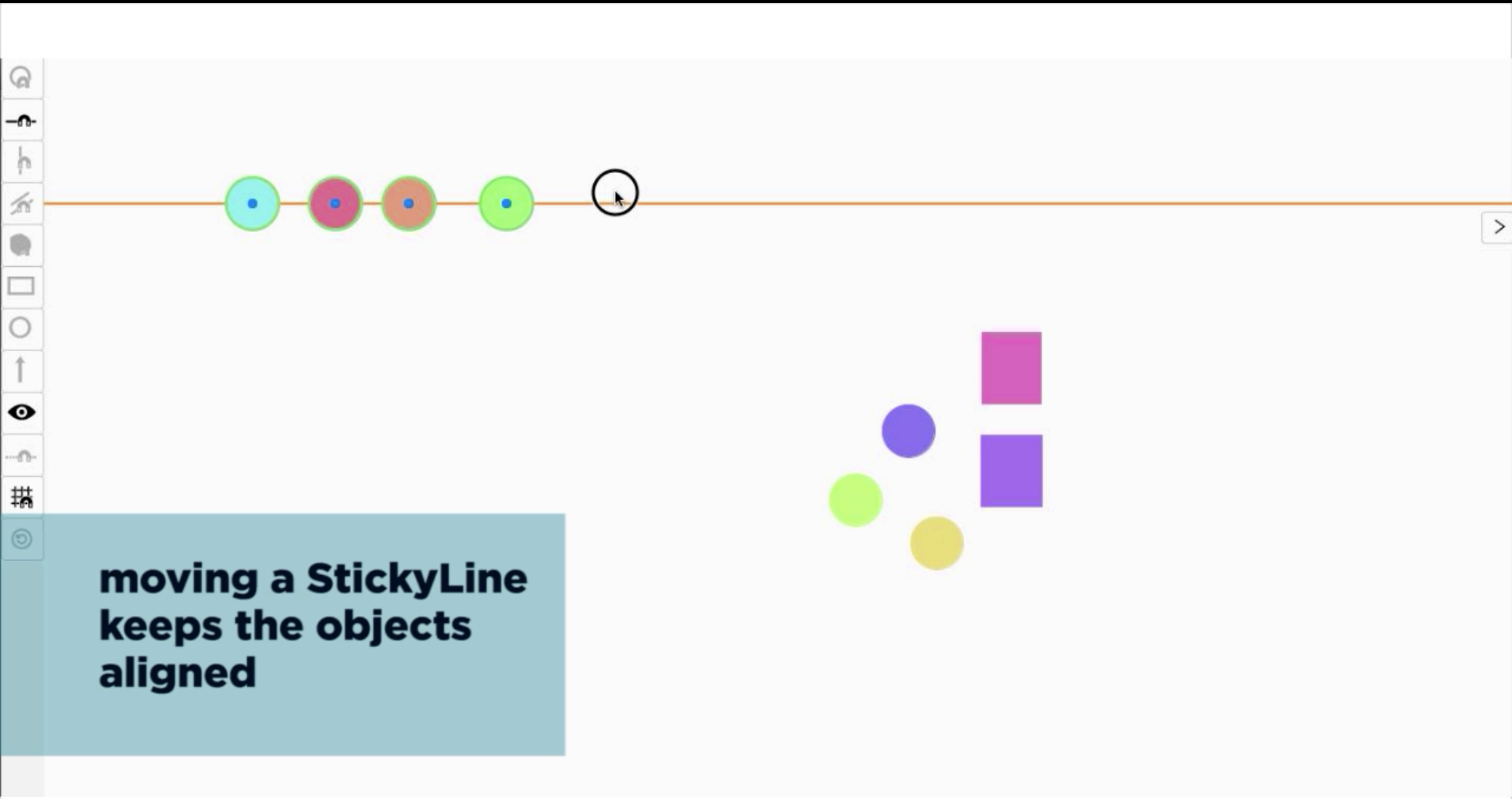
Increases the scope of commands



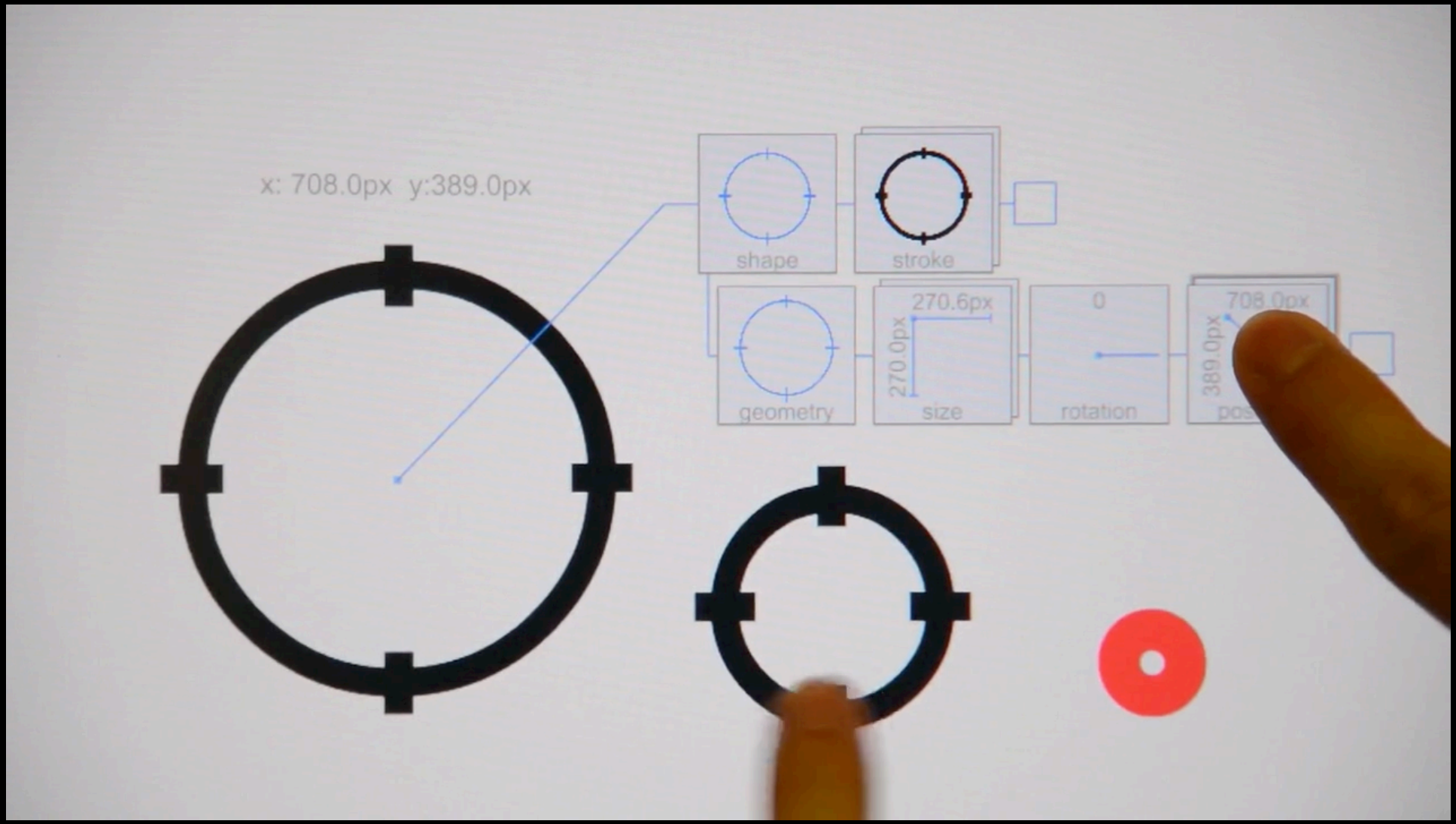
SideViews - Terry & Mynatt, 2002



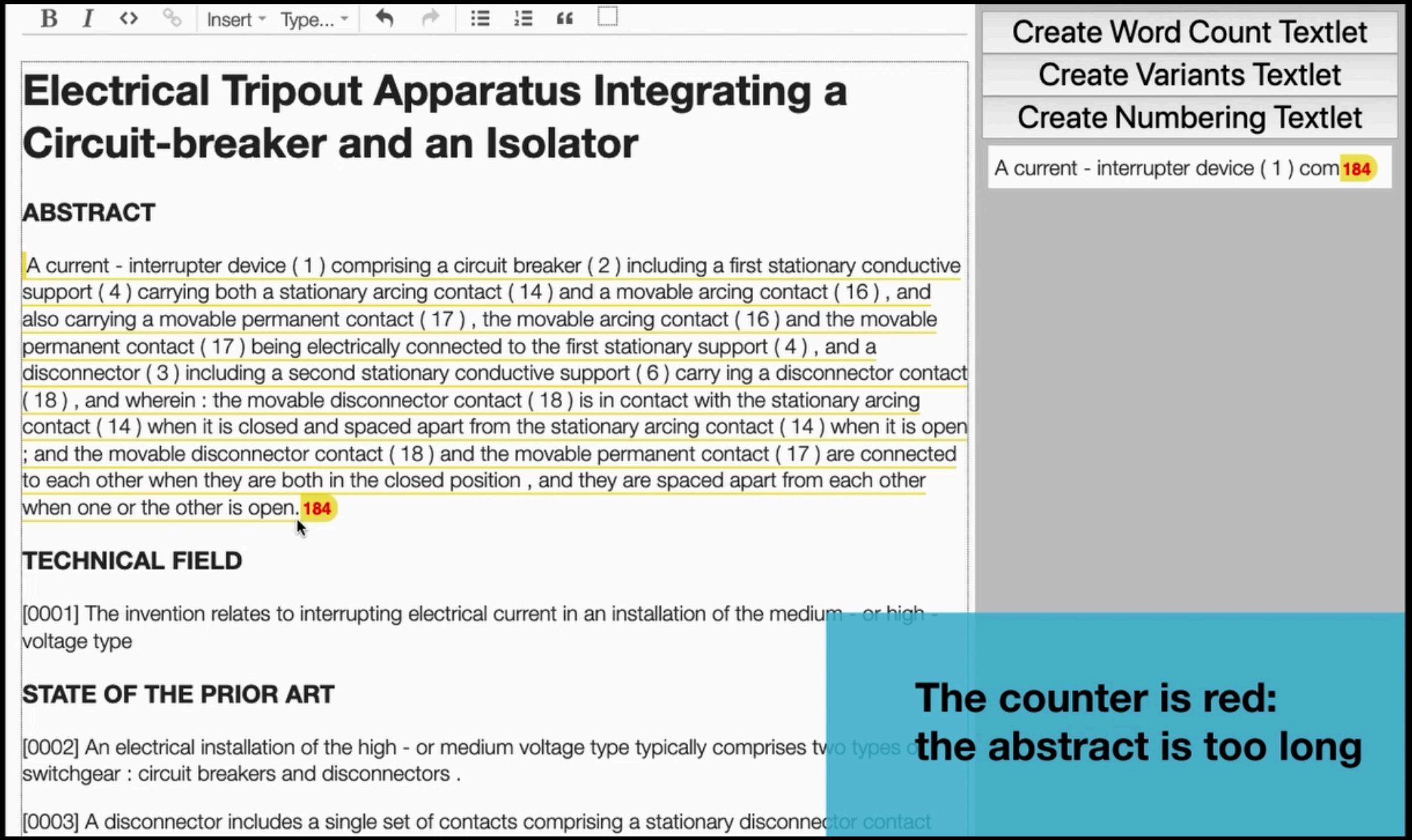
Procedural Drawing - Jacobs, 2015



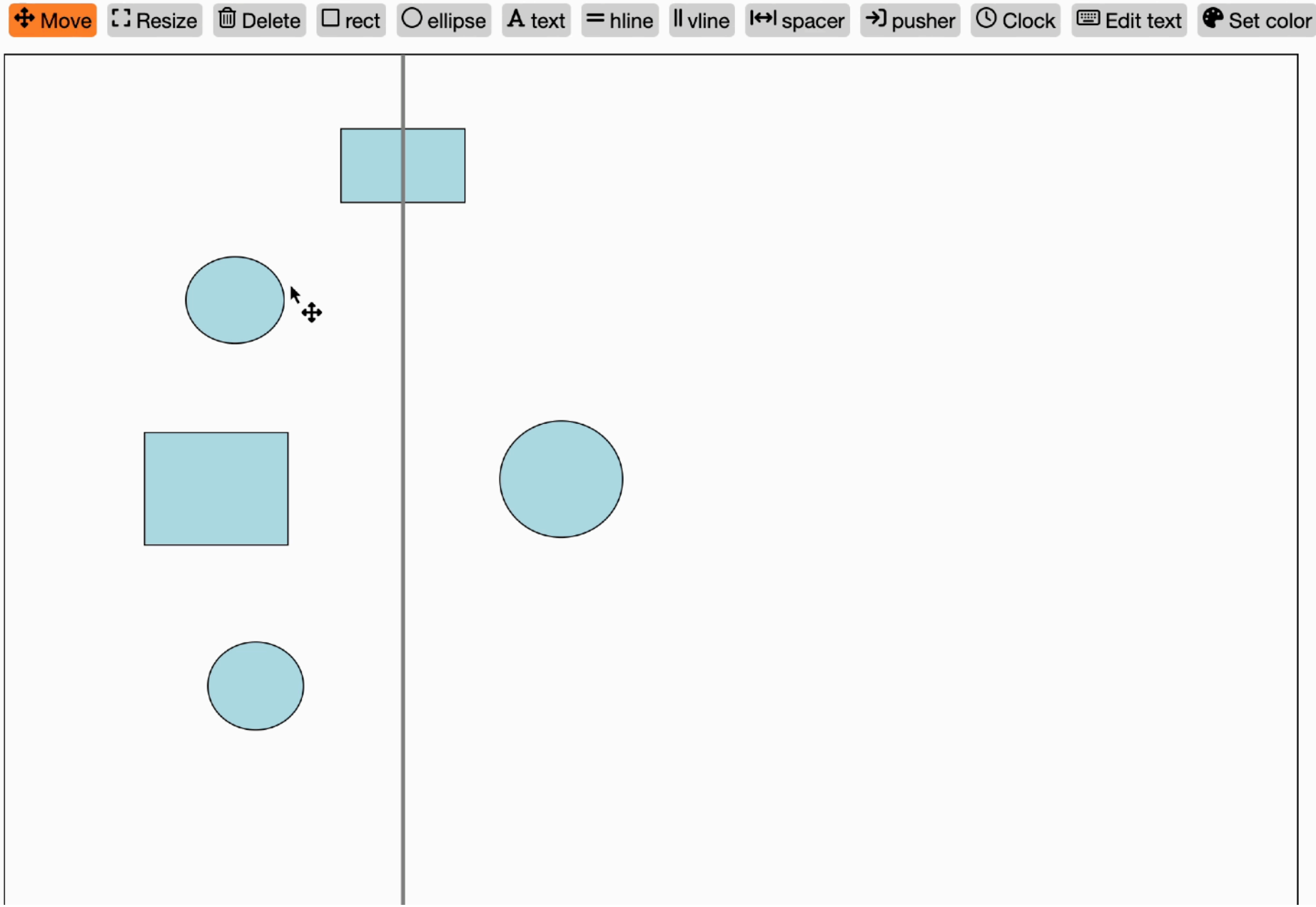
StickyLines - Ciolfi Felice et al, 2016



Object-Oriented Drawing - Xia et al, 2017

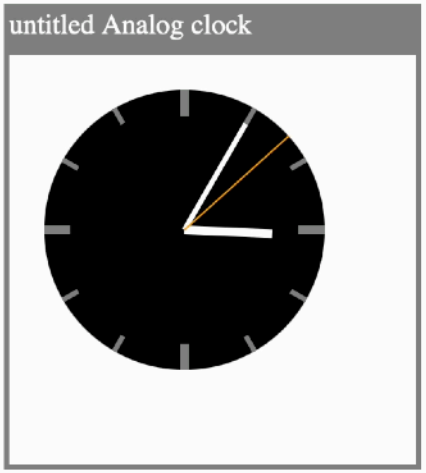


Textlets - Han et al, 2020


Move
Resize
Delete
rect
ellipse
text
hline
vline
spacer
pusher
Clock
Edit text
Set color

New window | Delete | Move | Resize | Edit text | Set color | Analog clock | WebPage

Canvas | rect | ellipse | hline | vline | spacer | pusher | Load



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