

GUI & WIMP Interaction

- WIMP Interfaces
- Window System & Window Manager
- Direct Manipulation
- Instrumental Interaction

Graphical User Interface (GUI)

Hardware

- High resolution, high refresh graphics
- Keyboard e.g. mechanical, touchscreen
- Pointing device e.g. mouse, touchpad



Capabilities

- Display graphics, animation and text
- Manage text entry
- Point-and-click interaction



The system needs to

- **Handle input devices** - keyboard (text) and pointing (mouse/touch)
- **Provide output methods** - drawing primitives, bitmaps, text
- **Wait for user input** and respond to it in a timely manner.

Desktop GUI: WIMP

WIMP (Windows, Icons, Menus, Pointers) is the most common instantiation of a Graphical User Interface.

- Interaction: points + click (drag, double-click).



Advantages of WIMP interface design

- Each application is isolated within its own window(s).
 - Drawing and interacting with an application is restricted.
 - Applications have their own memory, resources.
 - Isolation between applications.
- System supports methods to move, resize, re-order, re-draw windows.
- System supports common presentation of applications/elements.
 - Provide common GUI elements for building apps (e.g. buttons).
 - Emphasizes recognition of interface features over recall of commands.
- Uses metaphor and direct manipulation to facilitate learning

Titlebar

AutoSave OFF

02.gui_interaction

Search in Presentation

Share Comments

Window controls

Toolbar (tools)

Section

Tooltip

Cursor

Slide (object)

Status bar

Notes Comments

Slide 2 of 31 English (United States) 89%

Hardware

- High resolution, high refresh graphics
- Keyboard e.g. mechanical, touchscreen
- Pointing device e.g. mouse, touchpad

Capabilities

- Display graphics, animation and text
- Manage text entry
- Point-and-click interaction

The system needs to

- Handle input devices - keyboard (text) and pointing (mouse/touch)
- Provide output methods - drawing primitives, bitmaps, text
- Wait for user input and respond to it in a timely manner.

Graphical User Interface (GUI)

Macintosh

Desktop GUI: WIMP

WIMP (Windows, Icons, Menus, Pointers) is the most common instantiation of a Graphical User Interface.

- Desktop interface, containing icons, and applications presented as windows.
- The user points and clicks (drags, double-clicks) to interact.

Advantages of WIMP interfaces

- Each application is isolated from other applications, within its own window.
- Drawing and interaction is contained within its own window.
- System supports methods to move, resize, re-order, re-draw windows.
- Common presentation of applications and their elements.
- Provides common GUI elements for building apps (e.g. buttons).
- Emphasizes **recognition** of interface features over recall of commands.
- Uses **metaphor** and **direct manipulation** to facilitate learning.

Before Windowing Systems

Systems typically ran a single "full screen" application

Click to add notes

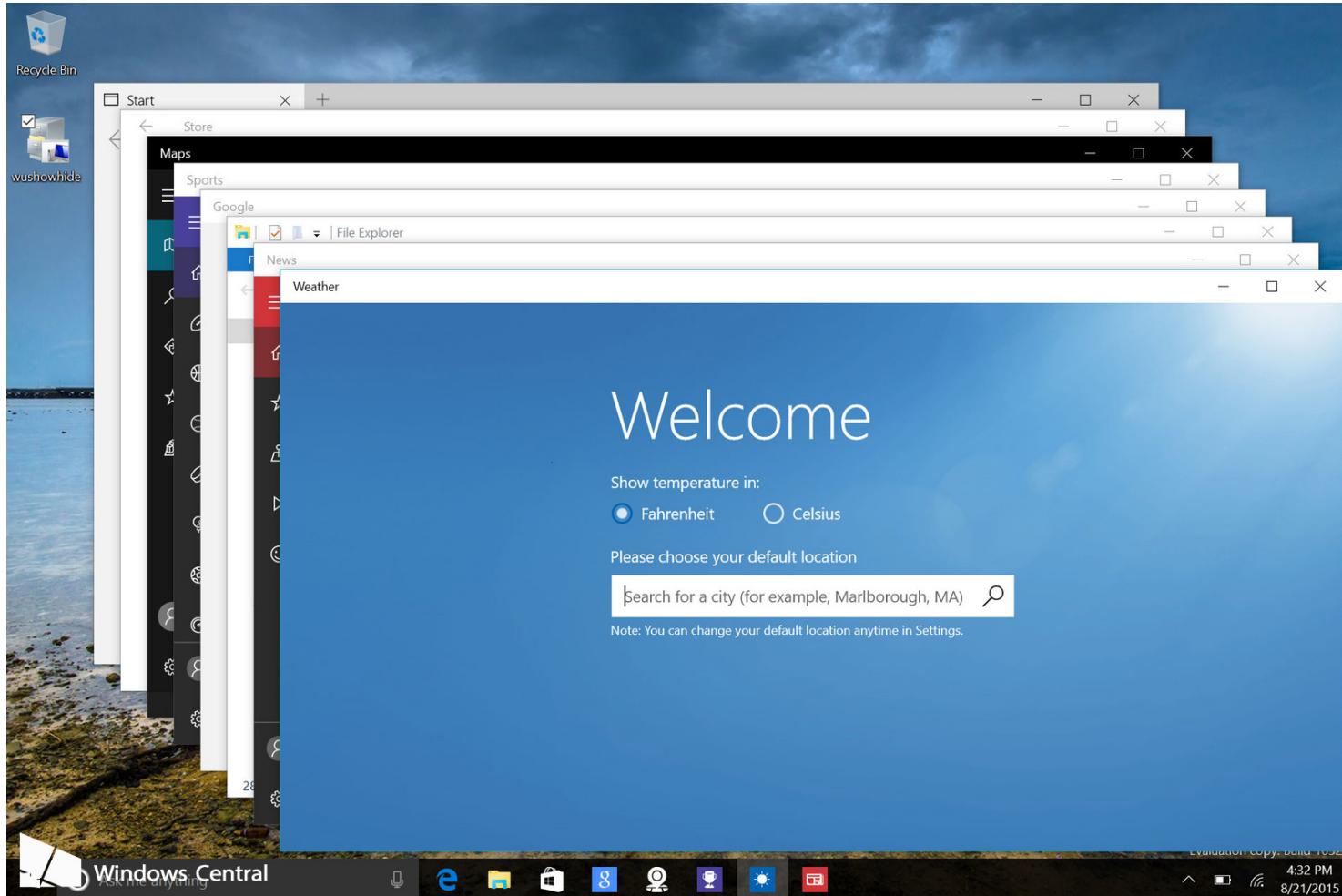
Before Windowing Systems

- Systems typically ran a single “full screen” application



A:A1: 'EMP								
		Worksheet		Range	Copy	Move	File	Print
		Global	Insert	Delete	Column	Erase	Titles	Graph
A	A	B	C	D	E	F	System	Quit
1	EMP	EMP NAME	DEPTNO	JOB	YEARS	SALARY	BONU	
2	1777	Azibad	4000	Sales	2	40000	1	
3	81964	Brown	6000	Sales	3	45000	1	
4	40370	Burns	6000	Mgr	4	75000	2	
5	50706	Caeser	7000	Mgr	3	65000	2	
6	49692	Curly	3000	Mgr	5	65000	2	
7	34791	Dabarrett	7000	Sales	2	45000	1	
8	84984	Daniels	1000	President	8	150000	10	
9	59937	Dempsey	3000	Sales	3	40000	1	
10	51515	Donovan	3000	Sales	2	30000		
11	48338	Fields	4000	Mgr	5	70000	2	
12	91574	Fiklore	1000	Admin	8	35000		
13	64596	Fine	5000	Mgr	3	75000	2	
14	13729	Green	1000	Mgr	5	90000	2	
15	55957	Hermann	4000	Sales	4	50000	1	
16	31619	Hodgedon	5000	Sales	2	40000	1	
17	1773	Howard	2000	Mgr	3	80000	2	
18	2165	Hugh	1000	Admin	5	30000		
19	23907	Johnson	1000	VP	1	100000	5	
20	7166	Laflare	2000	Sales	2	35000		

After Windowing Systems

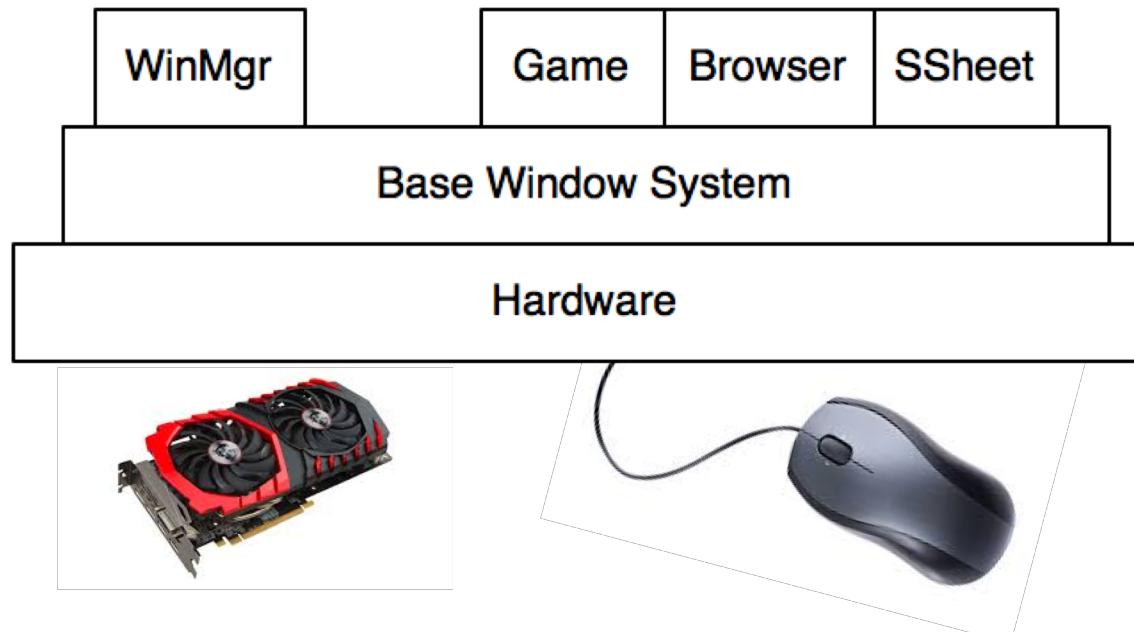
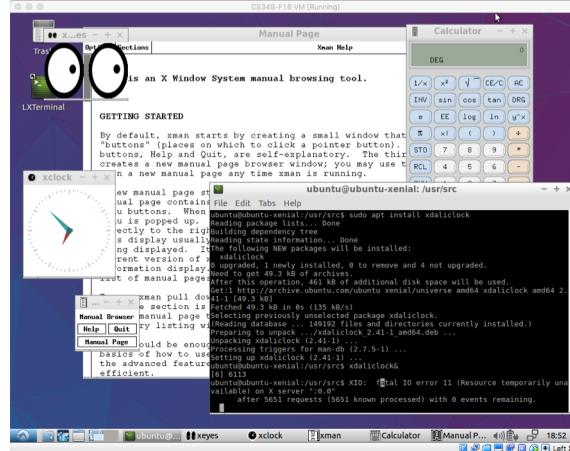


Windows are independent of one another; they don't need to know where they're located on the screen, or what other apps are running.

Challenges: keeping them separate, sending input to the correct window.

Windowing System (WS)

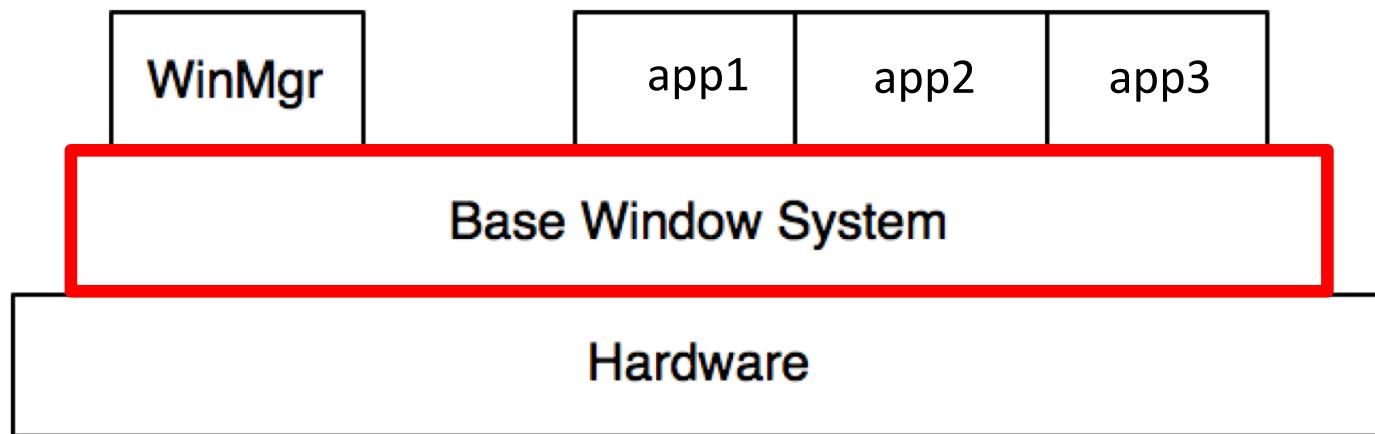
A **windowing system** is a software layer that provides these services - “low-level” input, output and window management capabilities – to the operating system.



Windowing System Functions

A Windowing System provides the following functionality:

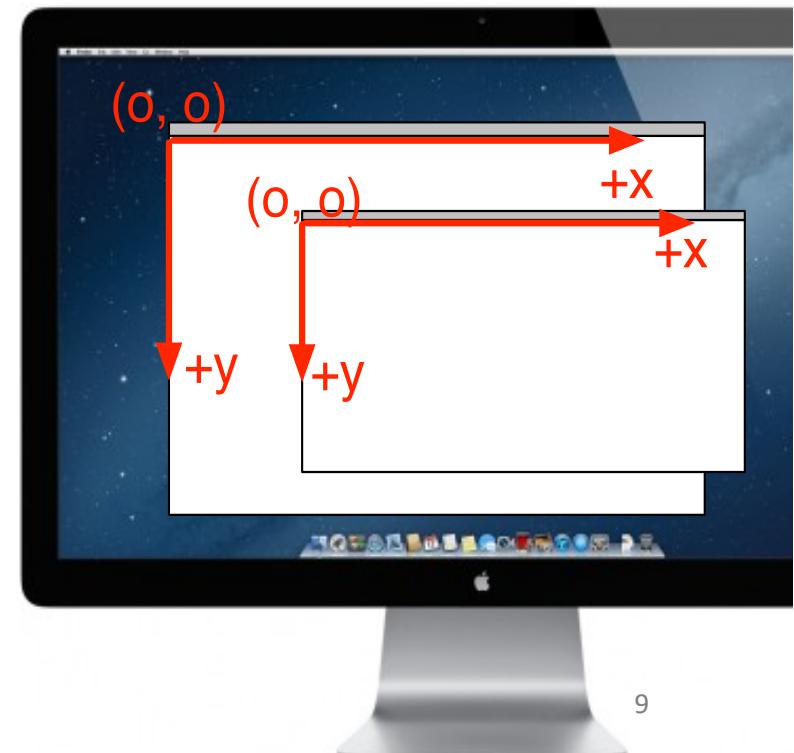
1. Provides OS -level routines for creating, destroying, and managing active application windows. This includes overlapping windows, tiling windows etc.
2. Routes mouse and keyboard input to the correct window. Typically the window that “has focus” receives input.
3. Manages access to underlying graphics hardware. Prevent simultaneous writes to the frame buffer (or video memory).



Canvas Abstraction

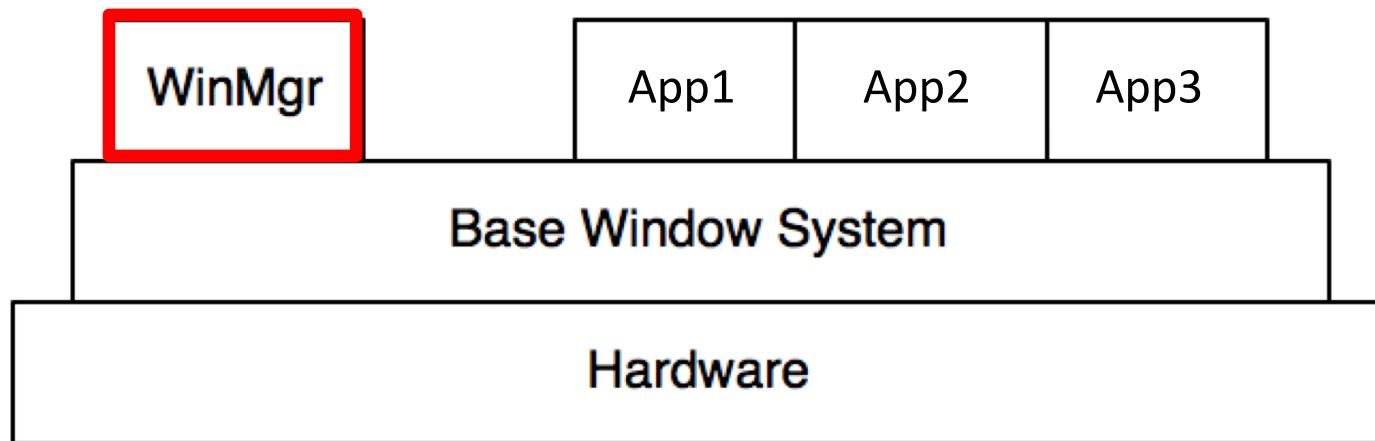
WS controls access to window contents using a “drawing canvas abstraction”

- The application is shielded from details of frame buffer, visibility of window, and all other application windows.
- Each window represents a “canvas” or drawing area for the application.
- Each window is independent and has no-knowledge of other windows
- Each window has its own coordinate system
 - WS transforms between global (screen) and local (window) coordinate systems
 - A window doesn't worry where it is on screen; program assumes its top-left is (0,0)
- WS provides access to graphics routines for drawing



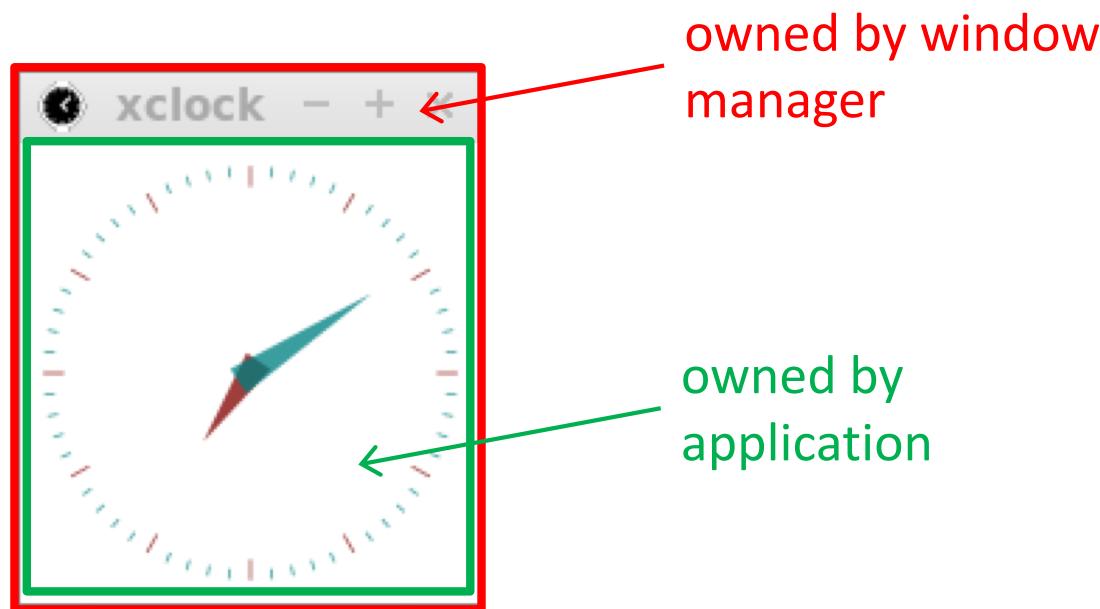
Window Manager (WM)

- Provides conceptually different functionality
 - Can be layered on top of Window System
 - Provides interactive components for windows (close button, resize handle)
 - Creates the “look and feel” of each window
- Application “owns” the contents of the window, but the WM and BWS “own” the application window itself!



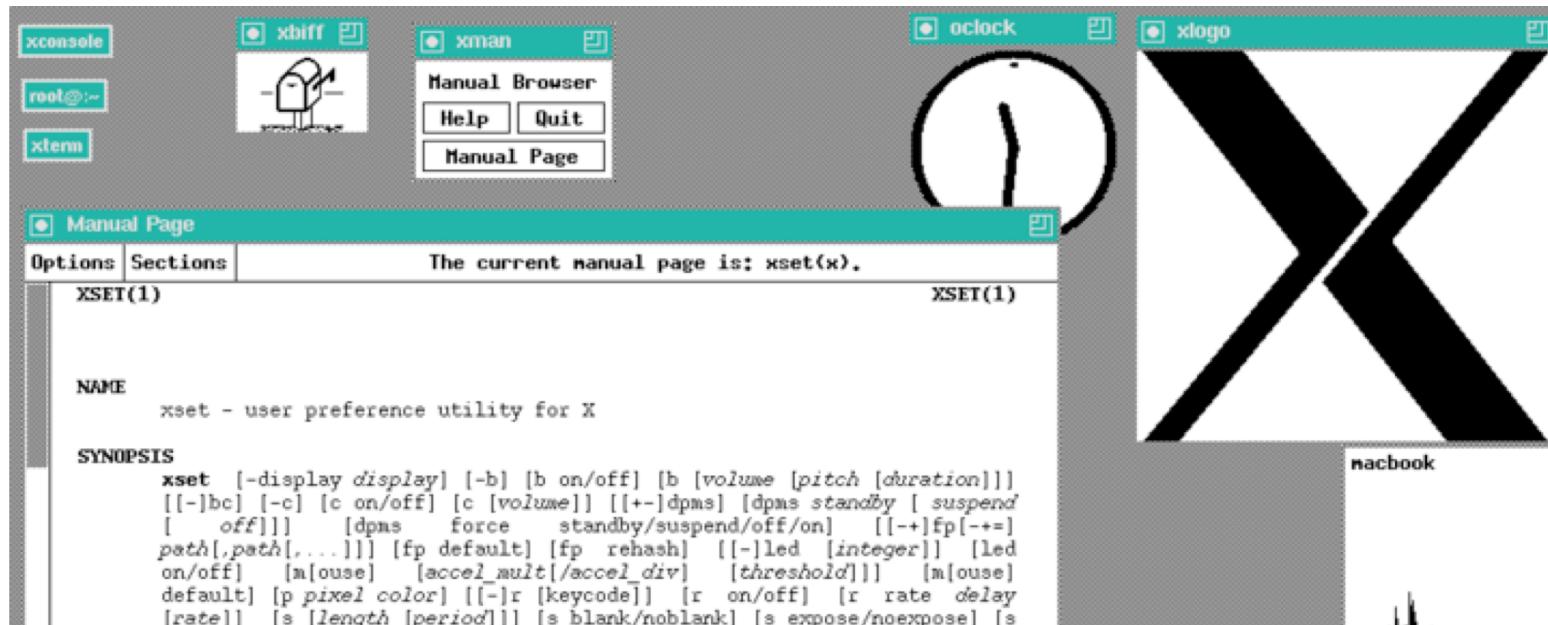
Roles in Managing the Window

- Window vs. Canvas
 - the Windowing System creates the window
 - the Window Manager manages interaction with the window (including its min/max/restore controls, and titlebar)
 - the application manages the contents of the canvas



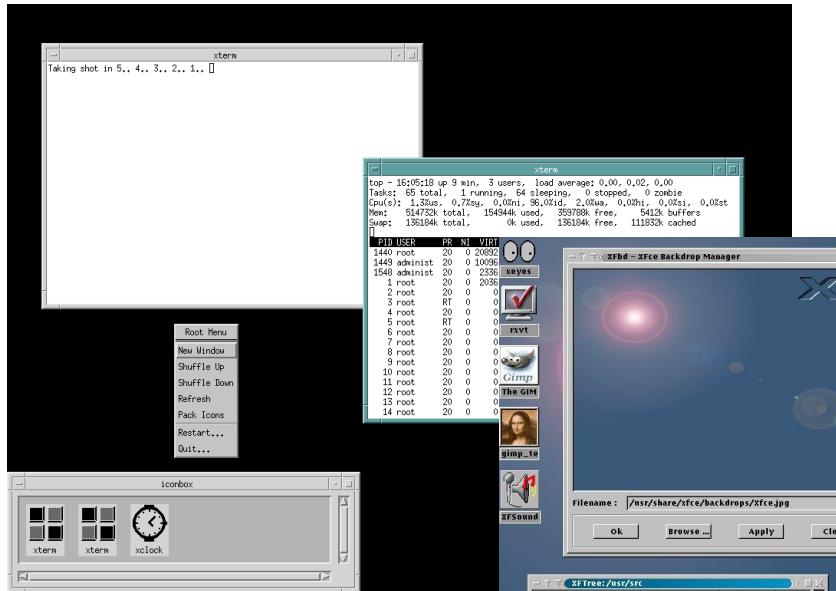
Example: X Windows as a Windowing System

- Developed in 1984 at MIT, extended by a consortium (HP, Sun)
- XWindows is the Unix standard **Windowing System**
 - Provides “low-level” input, output and window management capabilities to the underlying operating system.
 - Separate layer from operating system
- Essentially a protocol (X11) to manage input, output/graphics, windows
- Modern alternatives for Linux include Wayland, Metisse

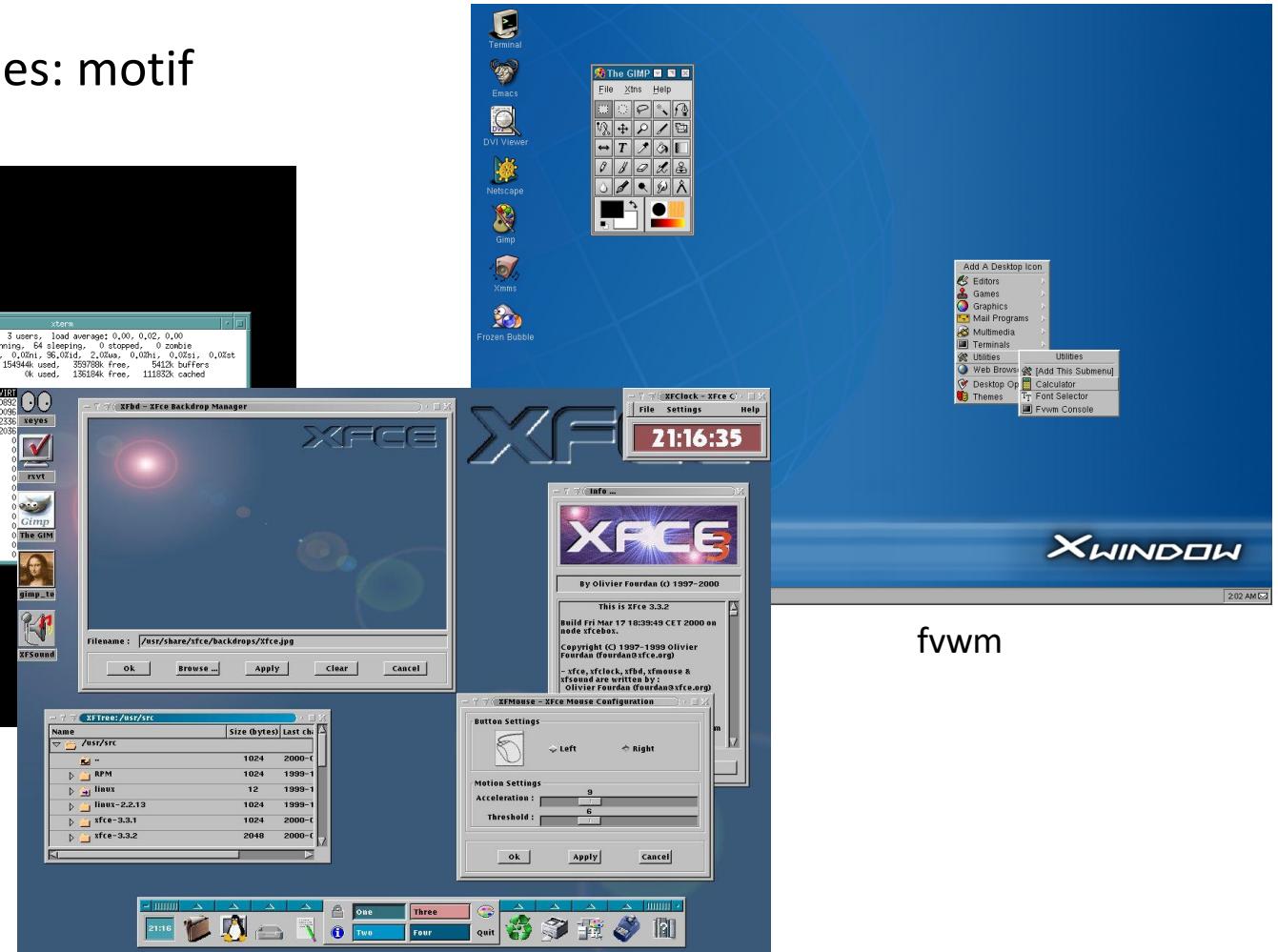


X Window Managers

- X Windows supports a pluggable **Window Manager**
- The window manager defines the look-and-feel of the application (e.g. colors, border width, font used).
- Commercial examples: motif



motif

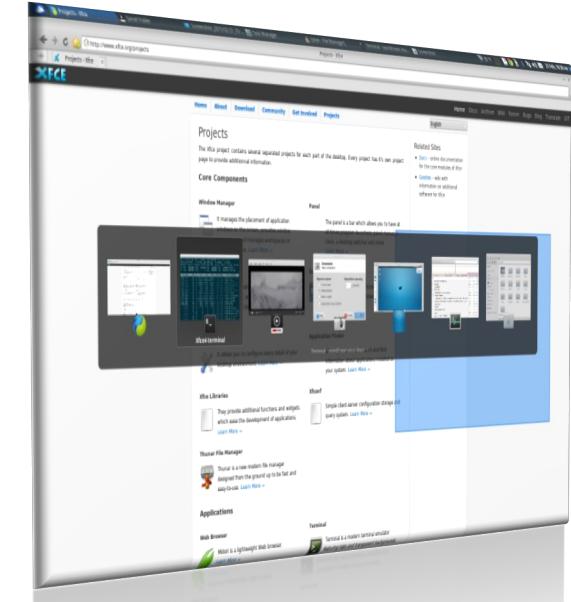
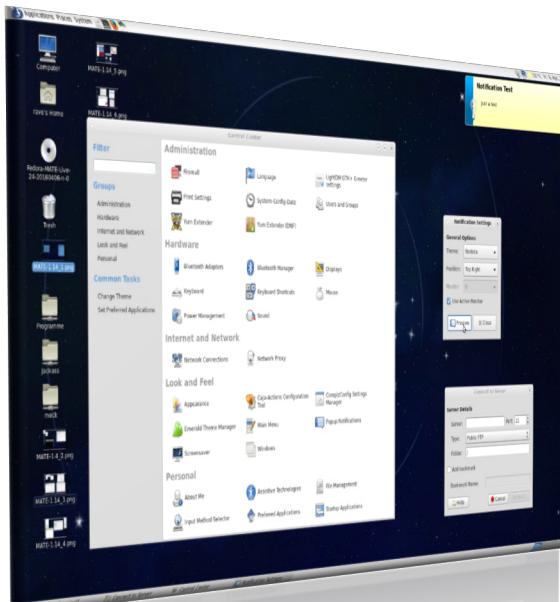


xfce

fvwm

Why Separate Window Manager from WS?

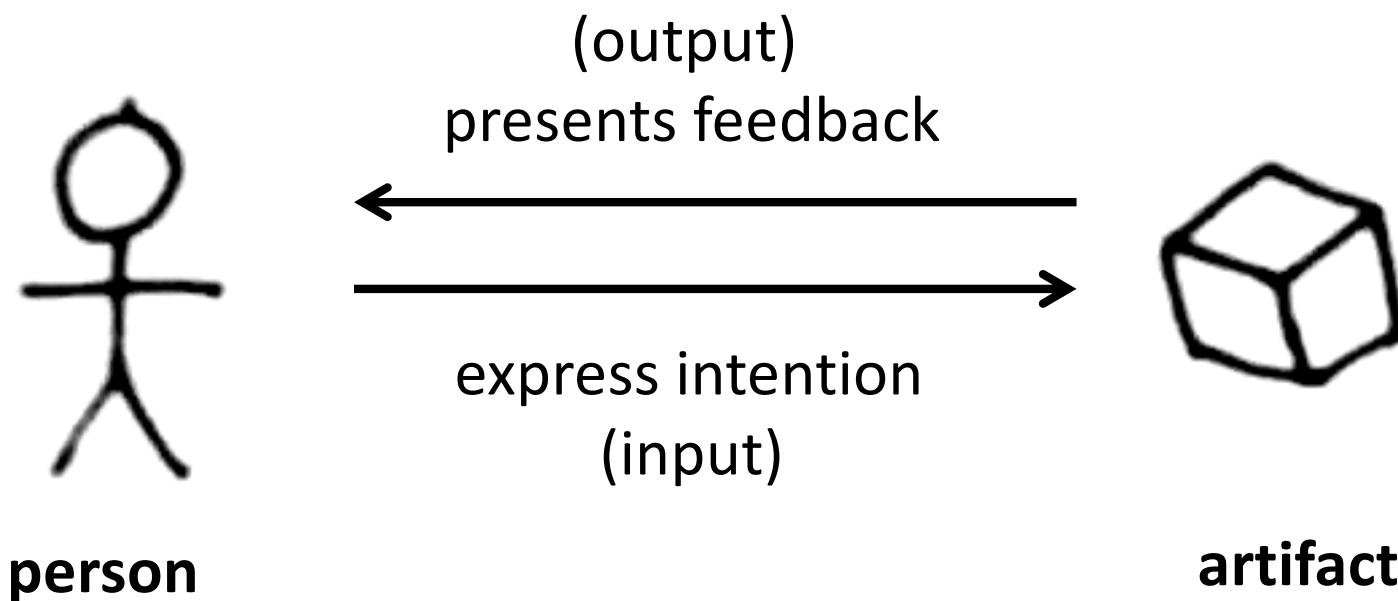
- Enables alternative “look and feels” for windowing system
(i.e. “Desktops” like Unity, GNOME, KDE, Xfce...)
- Enables different windowing paradigms
(i.e. Xmonad for tiled windows)
- macOS and Windows don’t expose a Window Manager (why?)



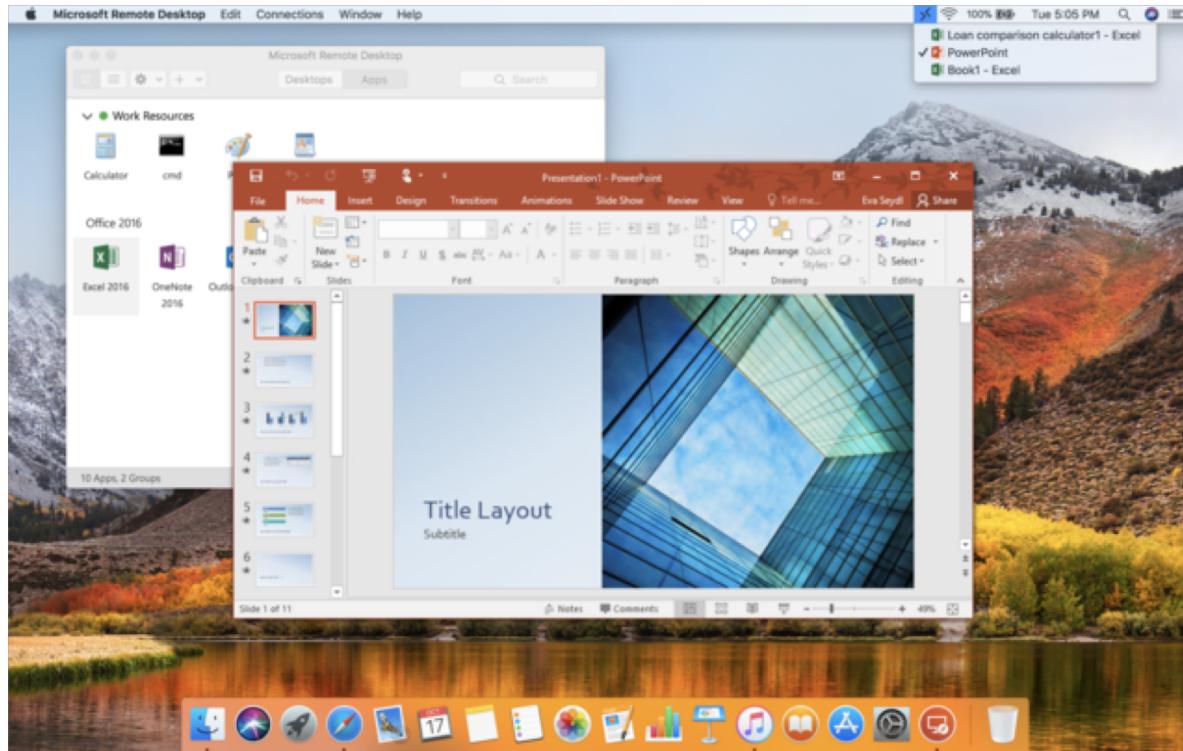
User Interaction

People typically interact with technology to perform a task.

They determine what they want to do, and provide guidance to the system, which gives them feedback.



WIMP Interaction



- Each application is represented by one or more windows
- Each window can be moved and resized, and overlap other windows
- The background (desktop) can hold icons, representing documents, folders or applications.
- Interaction consists of moving the mouse over a graphical element or **widget**, and clicking to interact.

Direct Manipulation

- Direct manipulation is when a *virtual representation of an object* is manipulated in a similar way to a real-world object.
- traced back to Ivan Sutherland's Sketchpad (see History lecture)
- proposed in 1983 by Ben Schneiderman

Indirect Manipulation

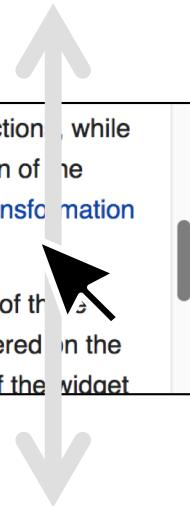
easily modify an object in the most common directions, while also attempting to be as intuitive as to the function of the widget as possible. The three most ubiquitous **transformation** widgets are mostly standardized and are:

- the **translation** widget, which usually consists of three arrows aligned with the orthogonal axes centered on the

Direct Manipulation

easily modify an object in the most common directions, while also attempting to be as intuitive as to the function of the widget as possible. The three most ubiquitous **transformation** widgets are mostly standardized and are:

- the **translation** widget, which usually consists of three arrows aligned with the orthogonal axes centered on the object to be translated. Dragging the center of the



```
>scroll -down 500
```

“Direct manipulation allows people to feel that they are directly controlling the objects represented by the computer.” (Apple)

Direct Manipulation Principles

Goal of Direct Manipulation is to make the interaction feel as-if the user was manipulating a “real object” instead of working through an intermediary.

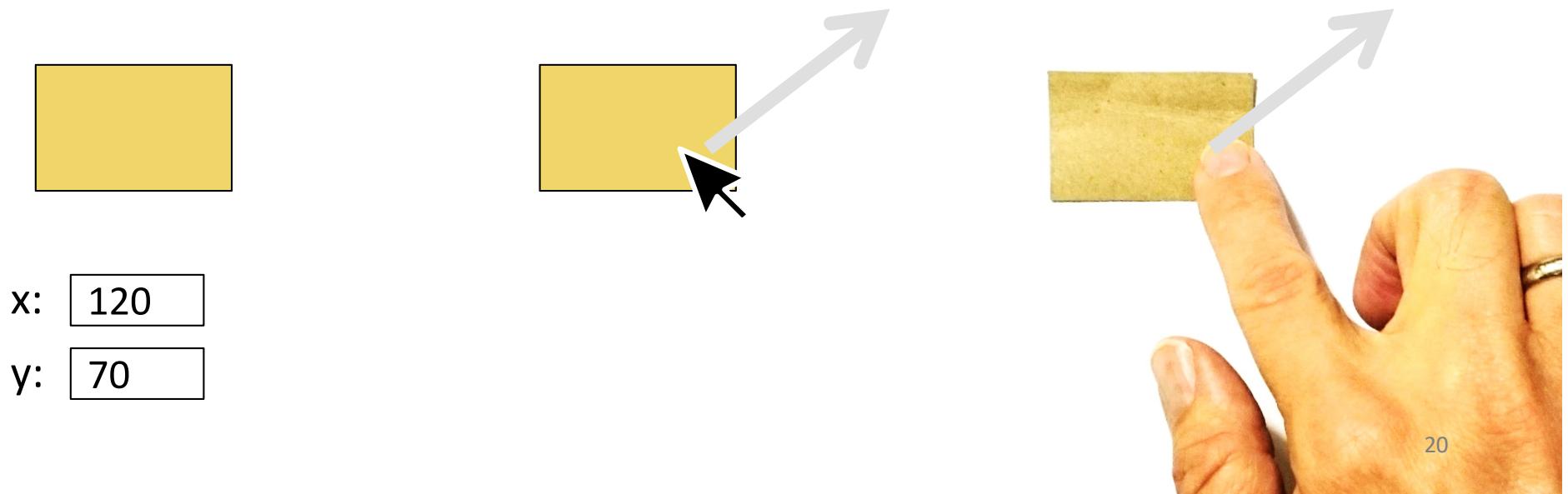
All related to manipulating the “objects of interest” or task objects.

1. Continuous representation of task objects and actions.
2. Task objects are manipulated by physical actions, not complex syntax
3. Fast, incremental, and reversible actions with effects on task objects immediately apparent.
4. Layered, self-revealing approach to learning

(from Schneiderman, 1983)

Benefit of Direct Manipulation

While interacting with DM interfaces, users feel as if they are interacting with the *domain rather than with the interface*, so they *focus on the task rather than on the technology*. There is a feeling of direct involvement with a world of task objects rather than communication with an intermediary.



Direct Manipulation on Desktop

- How does this apply to WIMP interfaces?
- Where do we see Direct Manipulation?
 - Desktop (container)
 - Click to activate (e.g. buttons, toolbars, other objects)
 - Click and drag to move icons or reposition elements (e.g. icon, file)
- Desktop metaphor is prevalent
 - Doesn't really use DM (e.g. we don't drag to open folders)

A GUI Doesn't Always Use Direct Manipulation

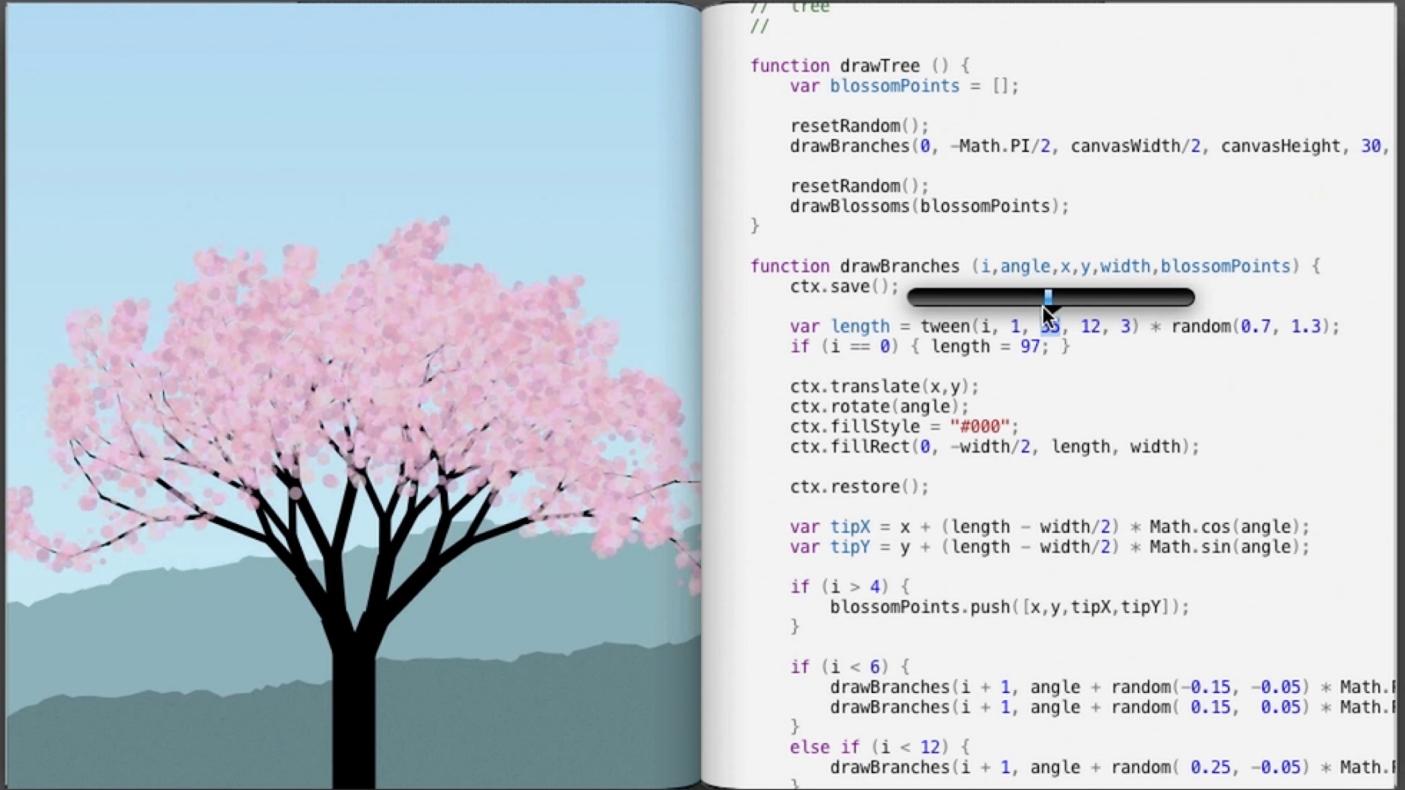
Desktop interfaces do a poor job of modeling Direct Manipulation (DM)

- Many commands are invoked *indirectly*
 - Menus, dialog boxes, toolbars are not really direct manipulation
... they're “tools” that pull users away from objects of interest
- Many objects of interest are *hidden*
 - Text styles, page layout,
- Many objects in the interface *are not objects of interest*
 - Toolbar palettes
- Desktop interfaces don't really feel like physical interactions.

-- Michel Beaudoin-Lafon, 2000

Direct Manipulation Issues

- Visually impaired users can't see the graphics; no linear flow for screen readers; physically impaired may have difficulty with required movements
- Consumes valuable screen space, forcing valuable information off-screen.
- Switching between keyboard and pointer is time consuming
- Analogies may not be clear
 - Users need to learn meaning of visual representations
 - Visual representations may be misleading



Bret Victor, Inventing on Principle (talk from CUSEC 2012)

- <http://vimeo.com/36579366>
- [https://www.student.cs.uwaterloo.ca/~cs349/videos/Bret Victor - Inventing on Principle-HD.mp4](https://www.student.cs.uwaterloo.ca/~cs349/videos/Bret%20Victor%20-%20Inventing%20on%20Principle-HD.mp4)

Interaction Model

“An interaction model is a set of principles, rules, and properties that guide the design of an interface. It describes **how to combine interaction techniques in a meaningful and consistent way** and defines the look and feel of the interaction from the user's perspective. Properties of the interaction model **can be used to evaluate specific interaction designs.**” (Lafon, 2000)

We want to define an interaction model that describes typical desktop interaction, and is compatible with Direct Manipulation.

Beaudouin-Lafon. 2000.

Instrumental interaction: an interaction model for designing post-WIMP user interfaces. Proceedings of CHI '00, 446-453.

<http://doi.acm.org/10.1145/332040.332473>

Instrumental Interaction

“A model of interaction based on how we naturally use tools (or instruments) to manipulate objects of in the physical world.”

- Interfaces have *interaction instruments* and *domain objects*
 - **Interaction instrument:** a necessary mediator between the user and domain objects
 - **Domain objects:** the thing of interest, data and associated attributes, which is manipulated using an interaction instrument

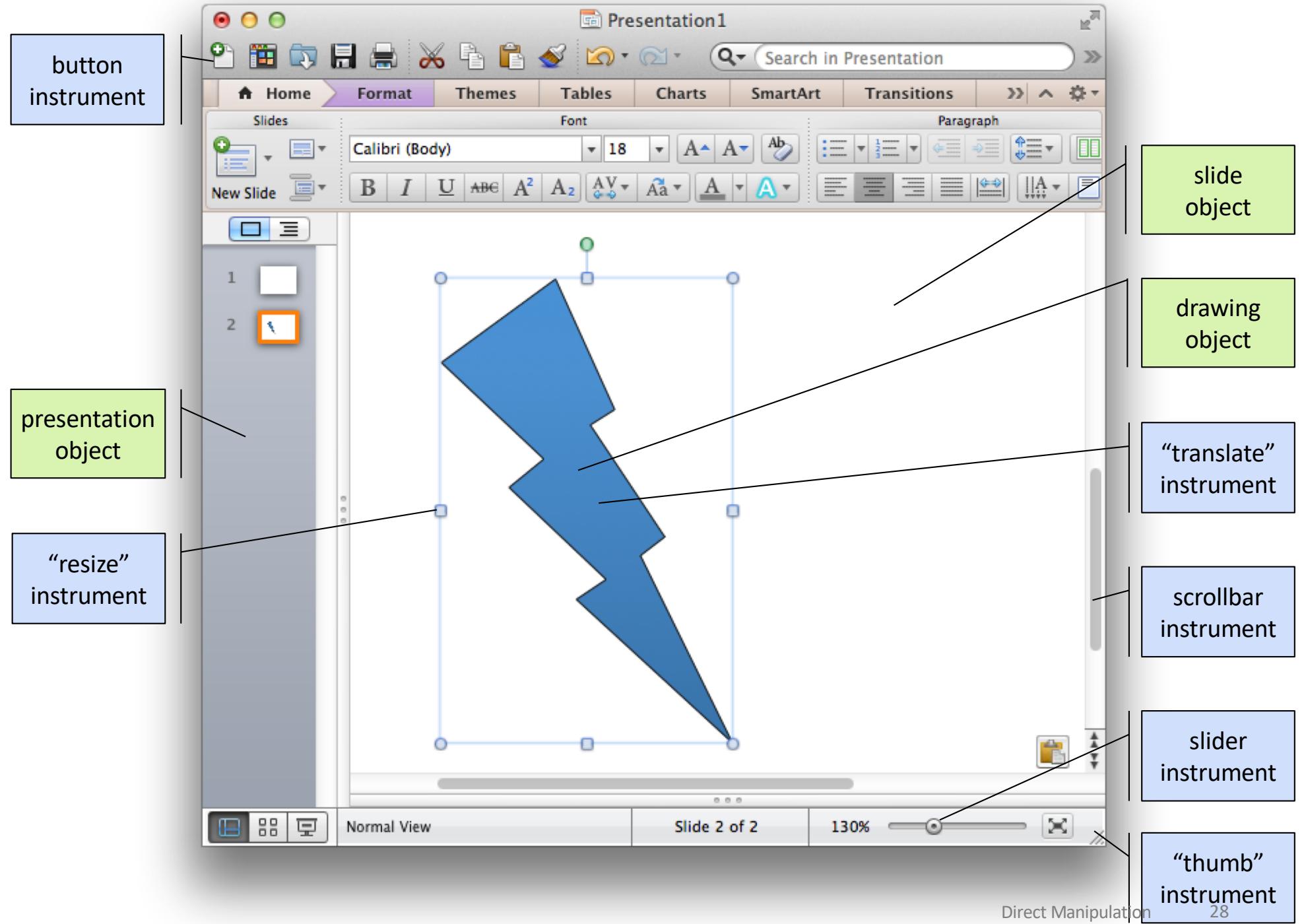


(Beaudoin-Lafon, 2000)



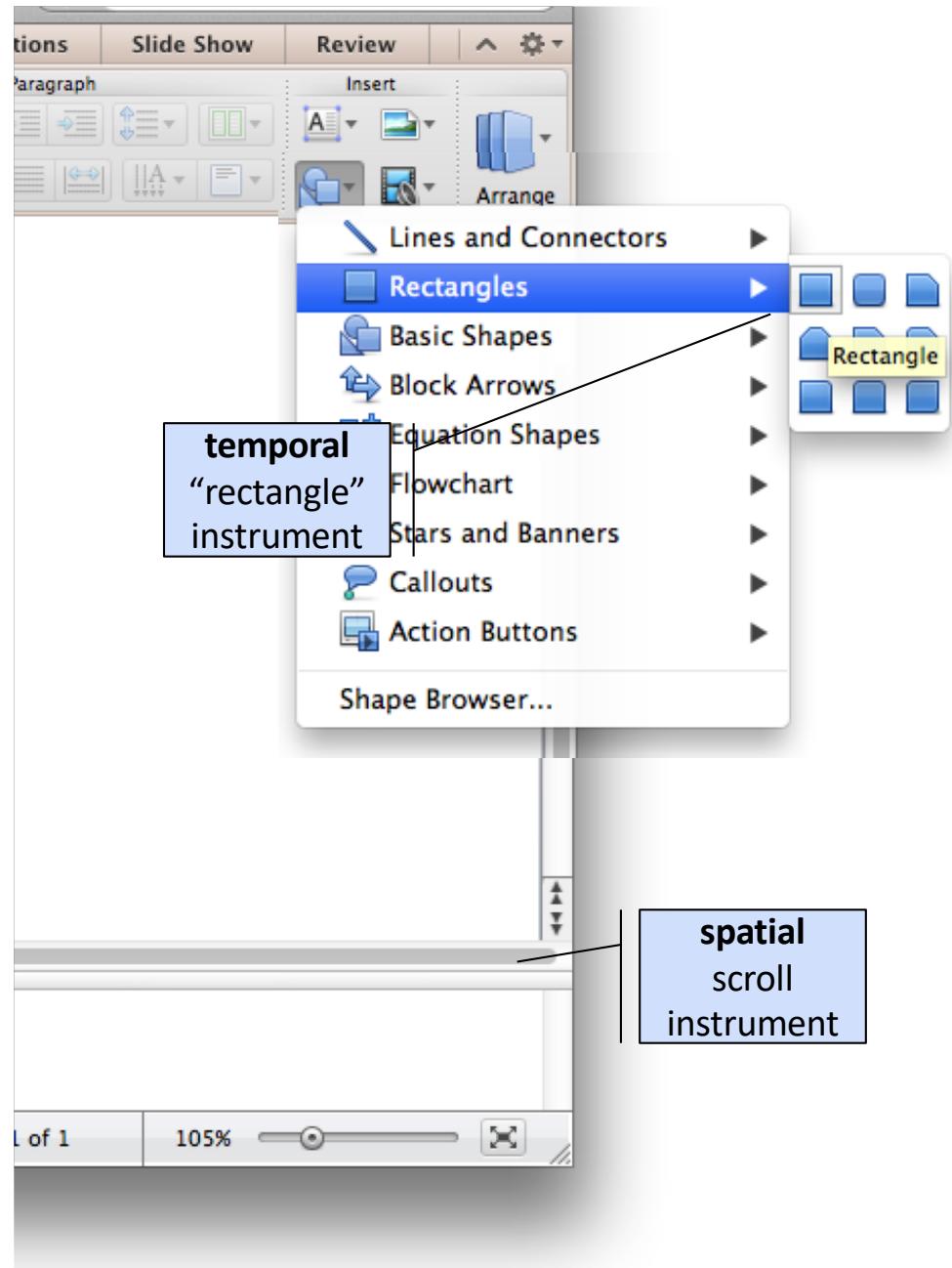


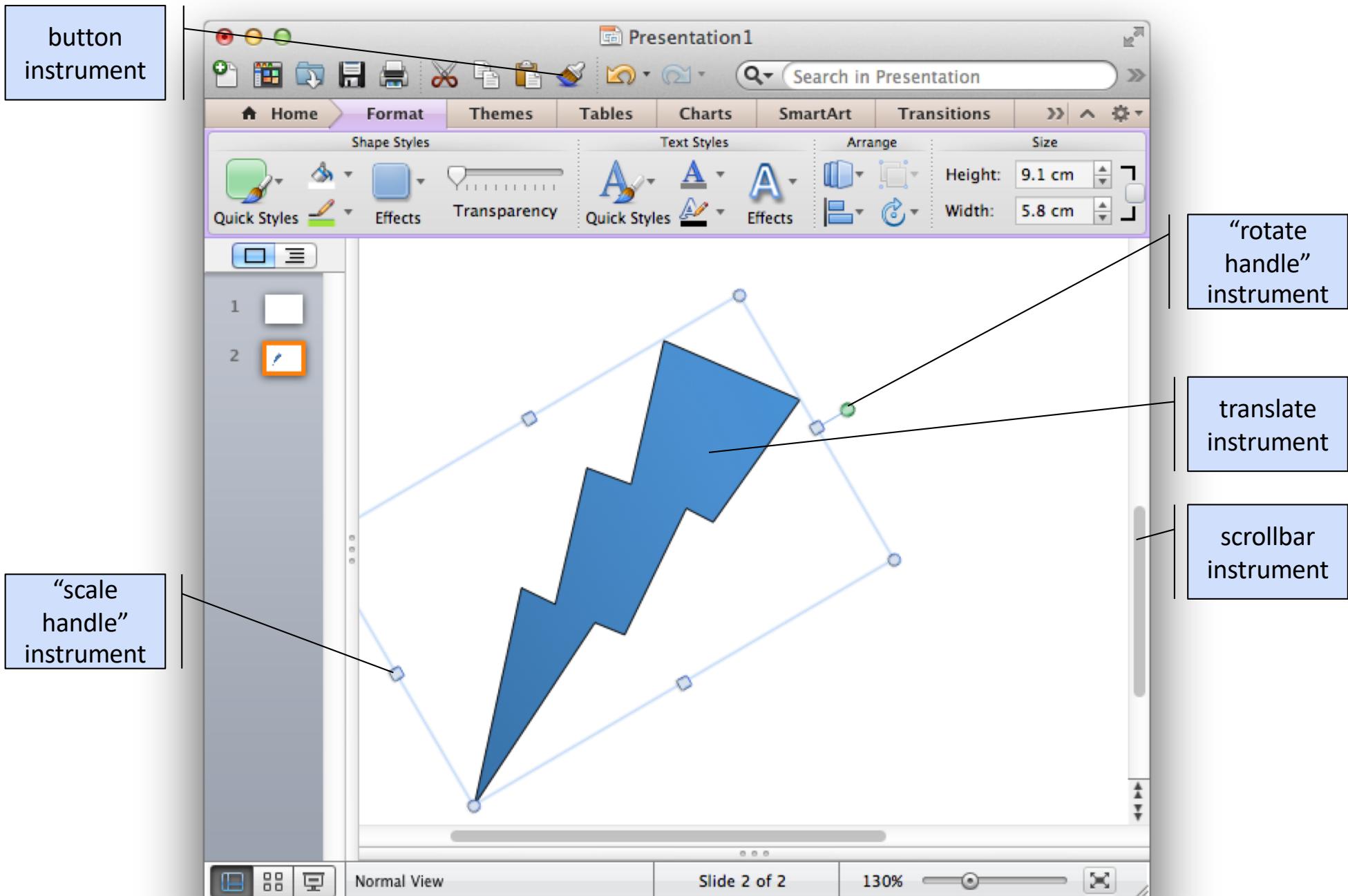
Indirect manipulation of a graphic (domain object).



Instrument Activation

- When instrument is under user's control
- WIMP instruments activated **spatially and temporally**
 - spatially has a *movement* cost
 - temporally has a *time* cost
- UI layout and design is concerned with the tradeoff of these costs





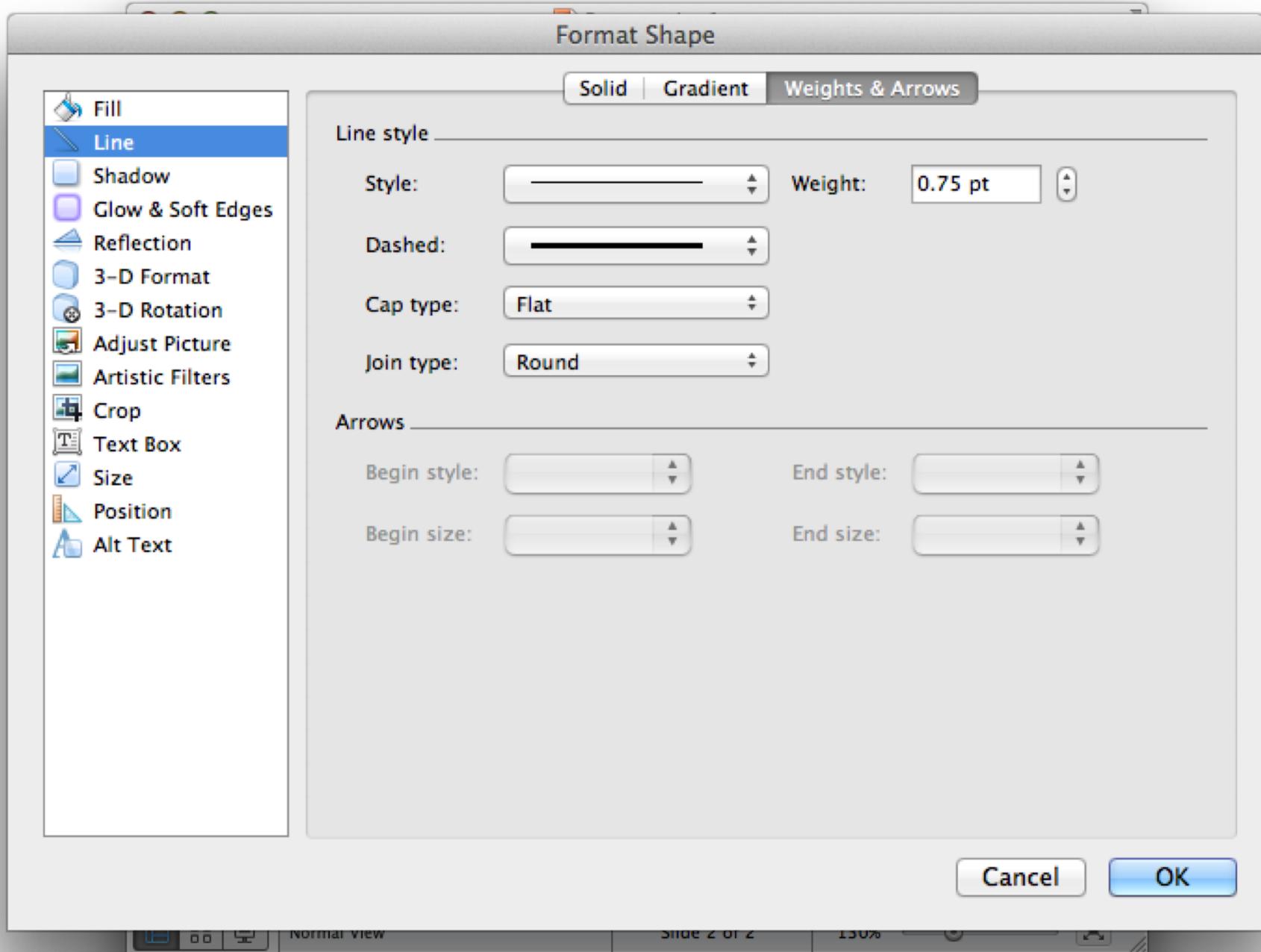
button
instrument

"rotate
handle"
instrument

translate
instrument

scrollbar
instrument

"scale
handle"
instrument



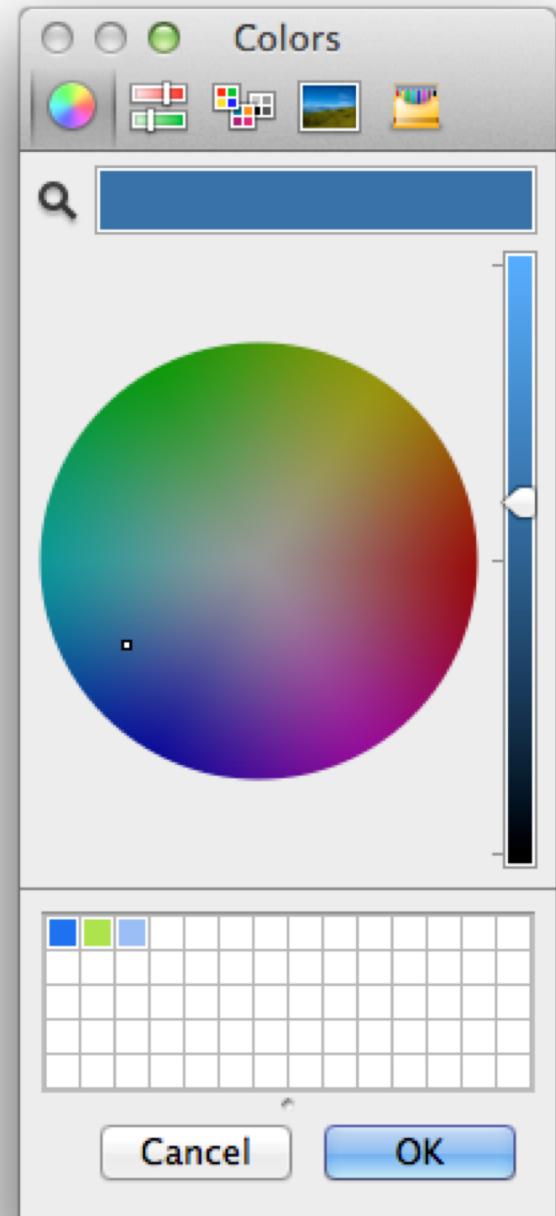
Reification and Meta-Instruments

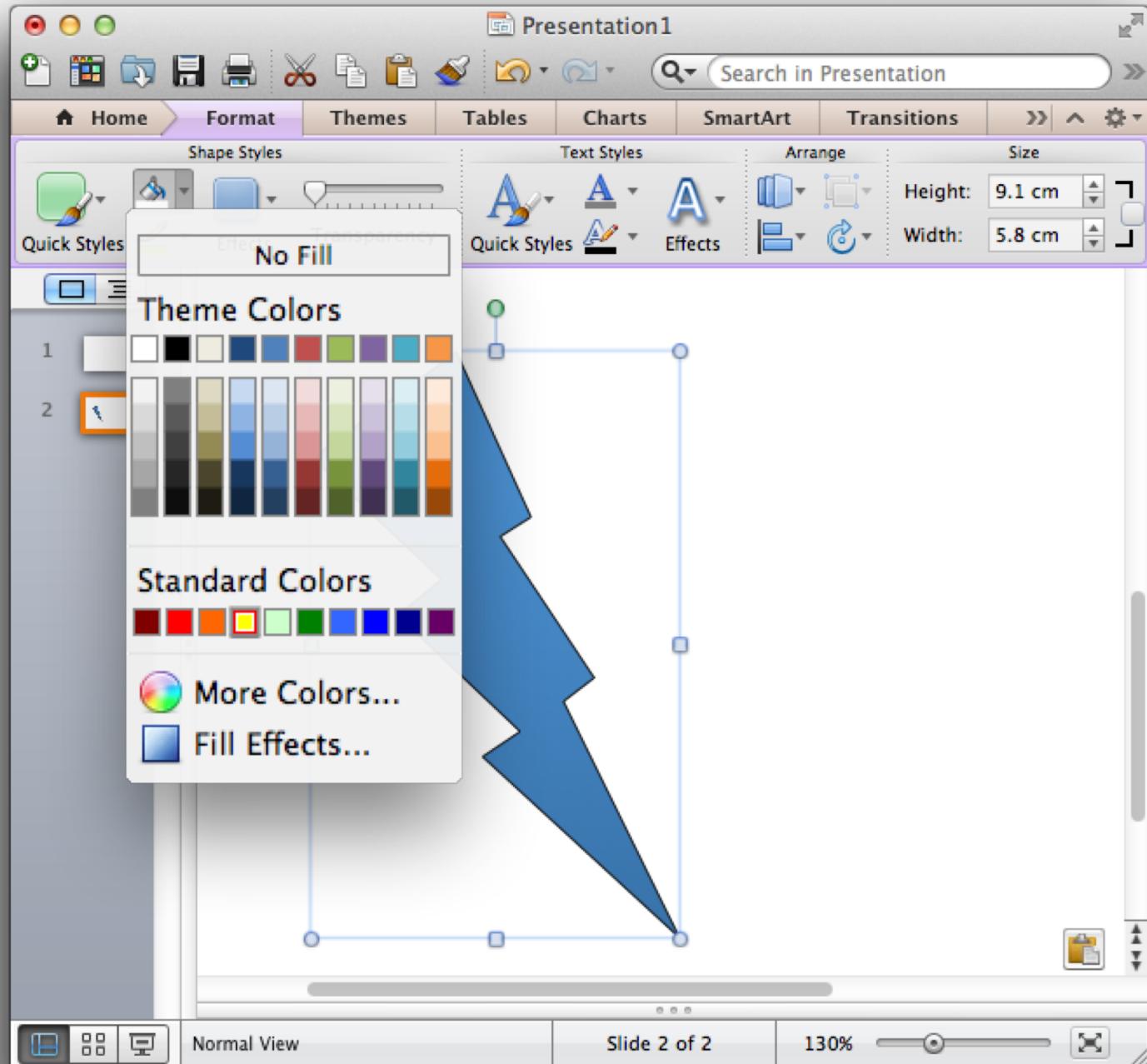
- **Reification:** turning concepts into something concrete
 - i.e. an instrument is the *reification* of a command
 - e.g. a scrollbar *reifies* scroll commands
- **Meta-instrument:** an instrument that acts on another instrument
 - the other instrument become an object of interest
 - e.g. a pencil is an instrument to manipulate the object “paper”, but when the pencil tip breaks, the pencil becomes an object of interest manipulated by a sharpener *meta-instrument*
 - GUI examples?



Object Reification

- Turning attributes of a primary object into other objects of interest
 - e.g. colour swatch, font styles, shader materials





Evaluating Instruments

How do we describe instruments?

How do we evaluate their effectiveness?

- Degree of **indirection**
 - Spatial/temporal offset between instrument and action on object
- Degree of **integration**
 - Ratio of degrees of freedom of instrument to degrees of freedom of input device
- Degree of **compatibility**
 - Similarity of action on control device/instrument to action on object

Degree of Indirection

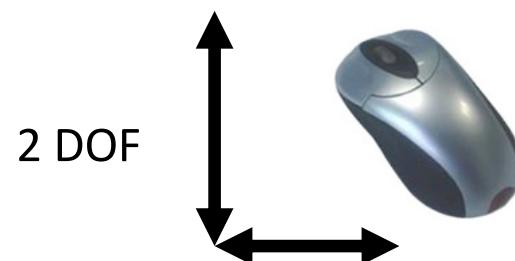
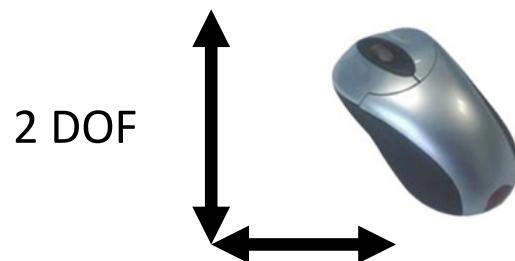
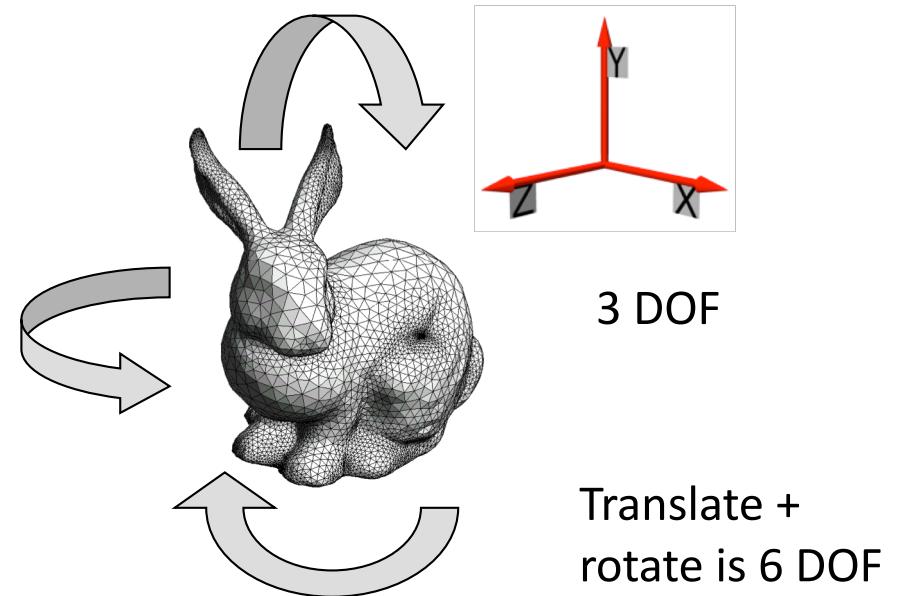
- 2D measure of spatial and temporal offsets of instrument (*lower==better*)
- Spatial Dimension
 - e.g. *near*: drag to translate, handles on to resize
 - e.g. *med*: scrollbars
 - e.g. *far*: dialog boxes
- Temporal Dimension
 - e.g. *short*: direct dragging response
 - e.g. *med*: activate tool in toolbar, start direct manipulation
 - e.g. *long*: using dialog, finishing drag-and-drop



Figure 2: Degree of indirection

Degree of Integration

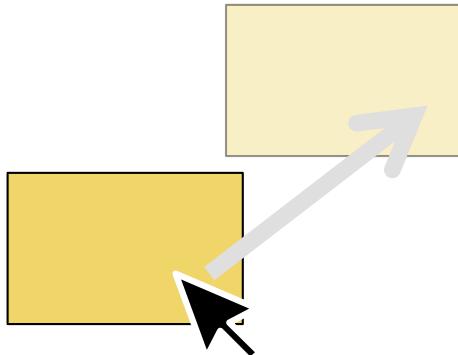
- the ratio between the number of degrees of freedom (DOF) of the instrument and the DOF captured by input device (reflects *suitability*)



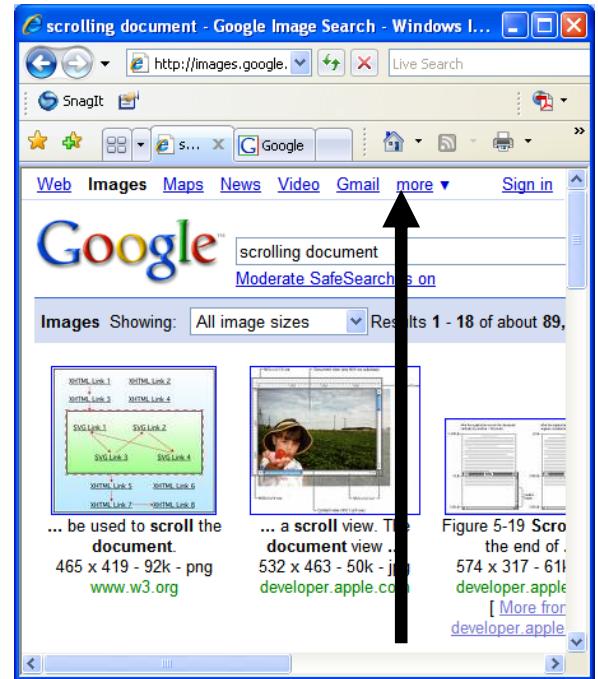
Degree of Compatibility

- the similarity between the physical actions on the instrument and the response of the object (similarity makes actions feel *natural* or *intuitive*).

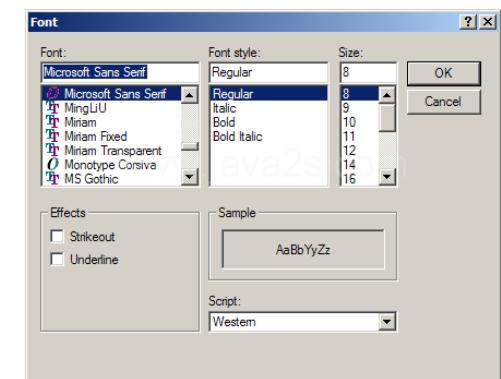
Dragging = high



Scrolling = medium



Dialog = low

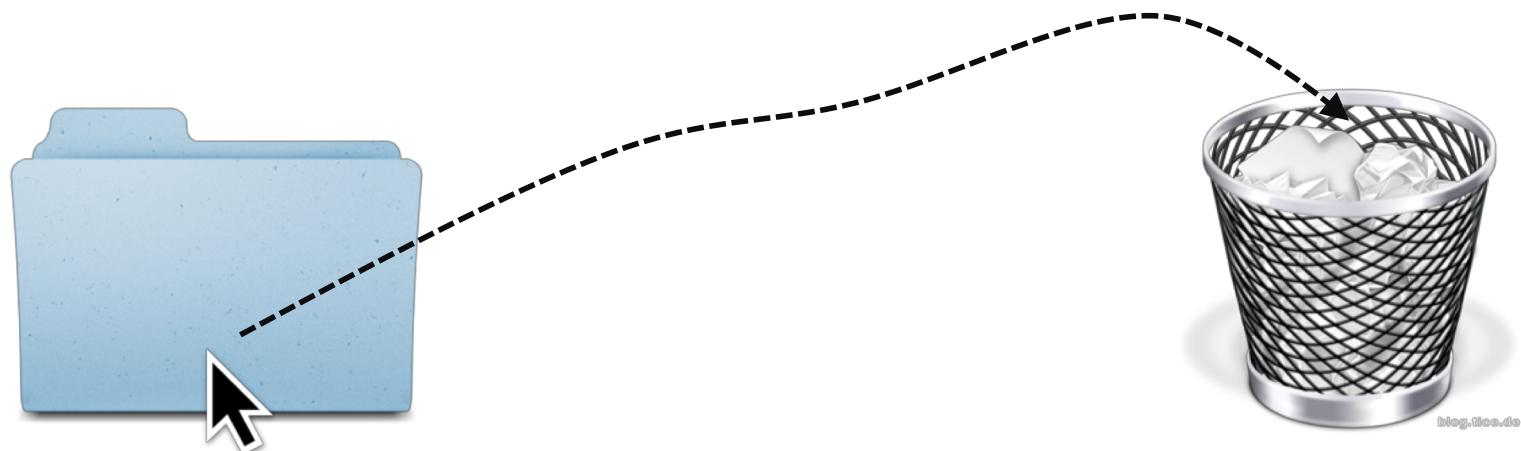


Direct Manipulation & Instrumental Interaction?

- A direct manipulation interface allows a user to *directly* act on a set of objects in the interface.
 - Low indirection (low spatial and temporal offsets)
 - High integration (1:1 correspondence)
 - High compatibility (similarity of action and effect)
- Direct means *instruments* are visually indistinguishable from *objects* they control
 - The actions on instrument/object entities are analogous to actions on similar objects in the real world.
 - The actions on instrument/object entities preserve the conceptual linkage between instrument and object.

Using Analogy in Direct Manipulation

- Perceived affordance: what the user believes that they can do with an object, based on its appearance. Actions that are suggested by the object.
 - e.g. handles should be grasped, knobs should be turned.
- “Affordances in the interface are like affordances for analogous actions in the real world”
 - Should build on existing experiences and intuitions to aid learning



Analyzing an Analogy

Real World	DM Interface
Object to be discarded	Icon of object to be discarded
Move hand to object	Move pointer to object
Pick up object with hand	Click to acquire object
Waste basket	Waste basket icon
Move to waste basket	Drag to waste basket icon
Release object from hand	Release button to discard object

