Direct Manipulation

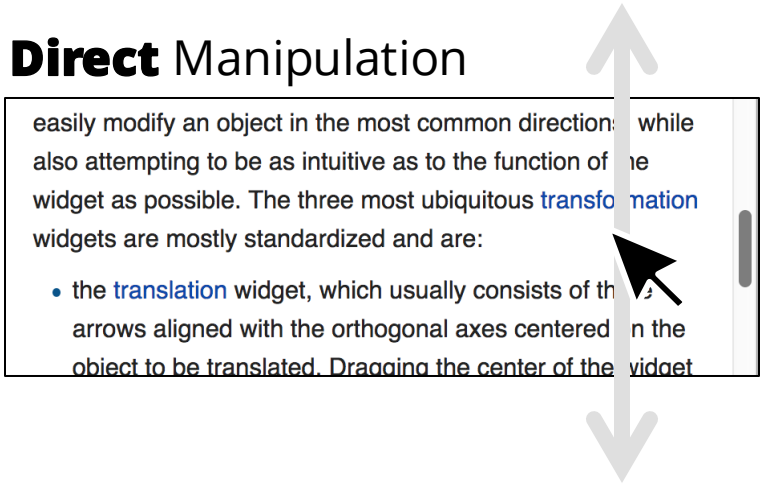
- Instrumental Interaction
- Dragging
- Drag-and-drop
- Transformable Shape

Direct Manipulation

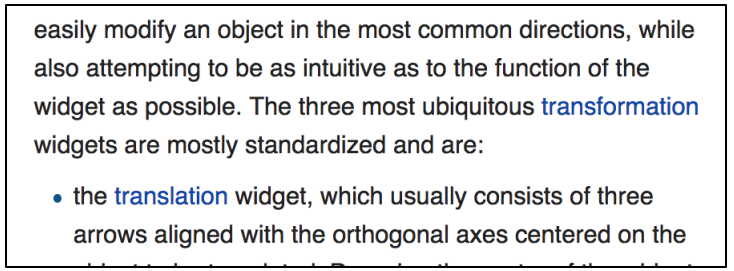
When a virtual representation of an object is manipulated in a similar way to a real–world object

(proposed in 1983 by Ben Schneiderman)

Direct Manipulation



Indirect Manipulation



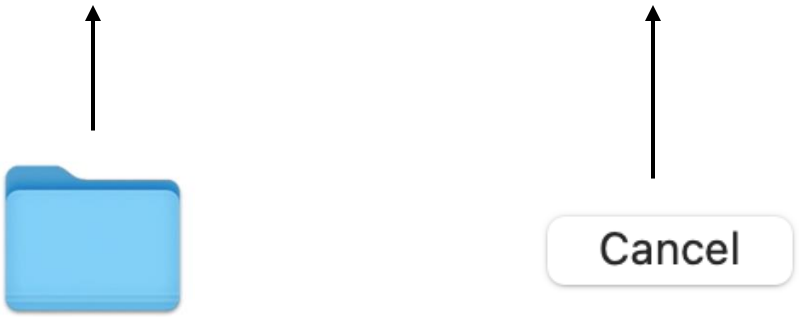

```
>scroll –down 500
```

Direct Manipulation

Make the interaction feel like manipulating a “real object” instead of working through an intermediary

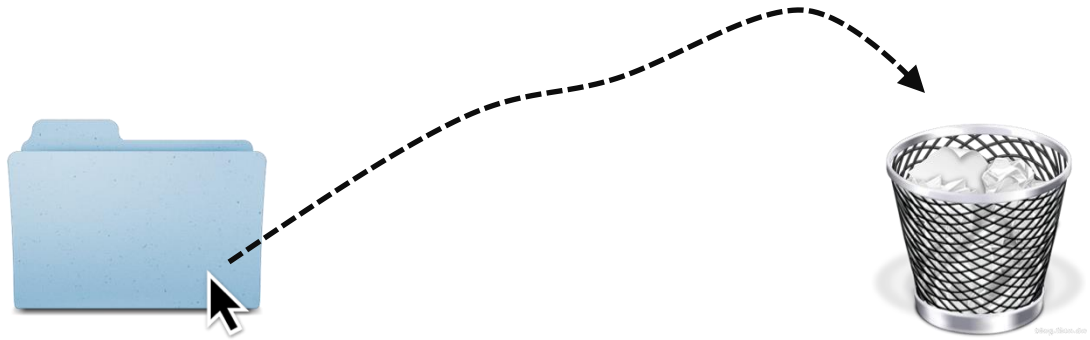
Requires a representation of "task objects"

- A *task object* is something the user can manipulate
- Can be the **object of interest** or an **interface widget**

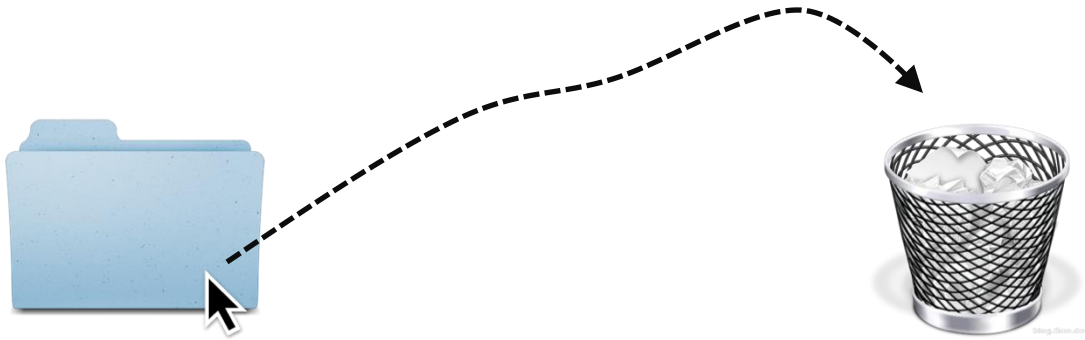


Analogous Behaviours

Real World	Direct Manipulation 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



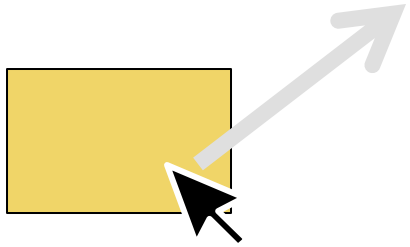
Schneiderman's Characteristics of Direct Manipulation



Benefit of Direct Manipulation

Feel as if interacting with the *task object rather than with the interface*, so you *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



x: 120

y: 70

BumpTop - A Multi Touch 3D Physics Desktop

- https://youtu.be/M0ODskdEPnQ

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**." (Beaudouin-Lafon, 2000)

We want an interaction model that works with typical desktop interaction, and is useful to describe **Direct Manipulation**

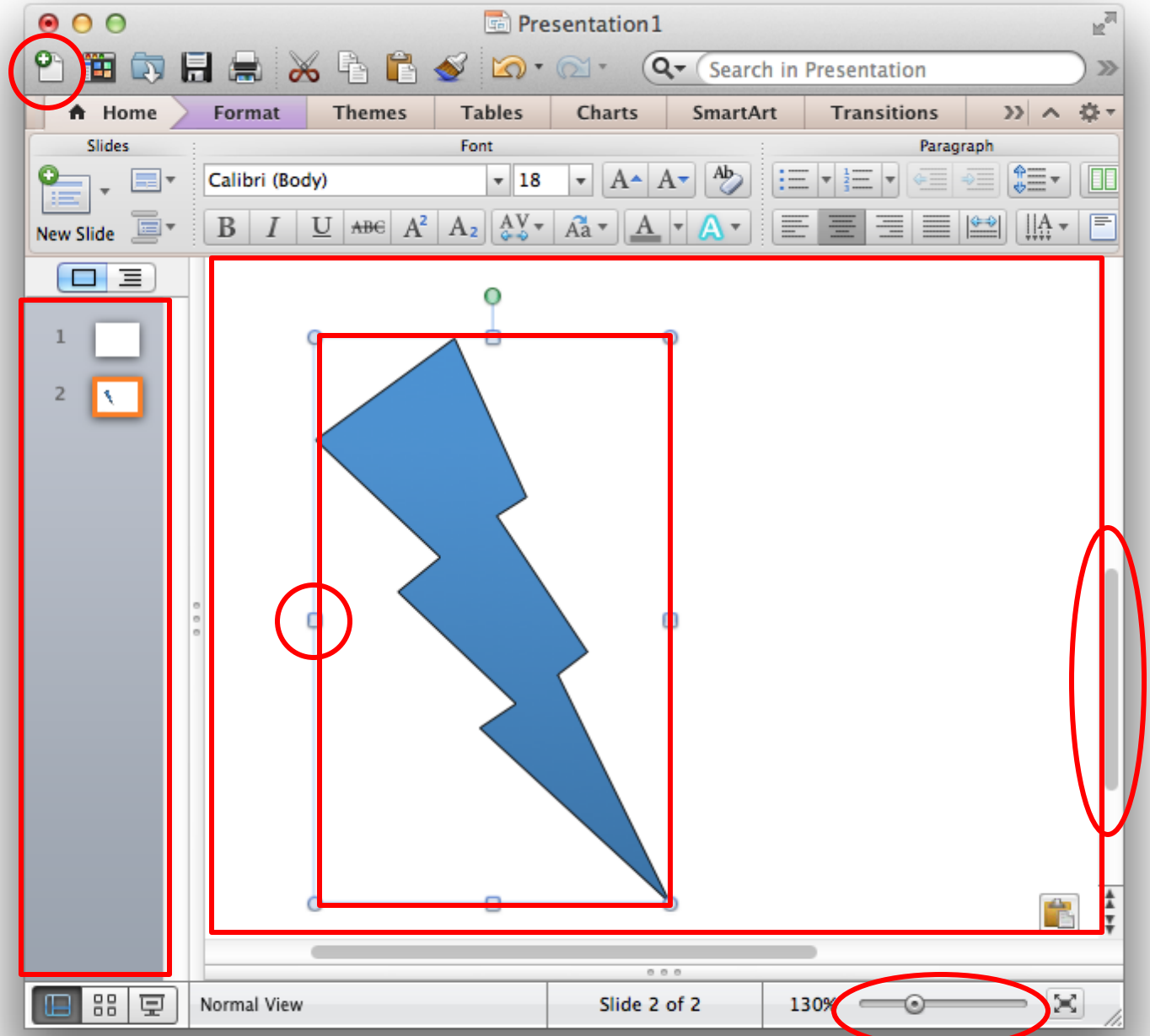
Beaudouin-Lafon. Instrumental interaction: an interaction model for designing post-WIMP user interfaces. Proceedings of CHI 2000, 446-453.

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

Instrumental Interaction

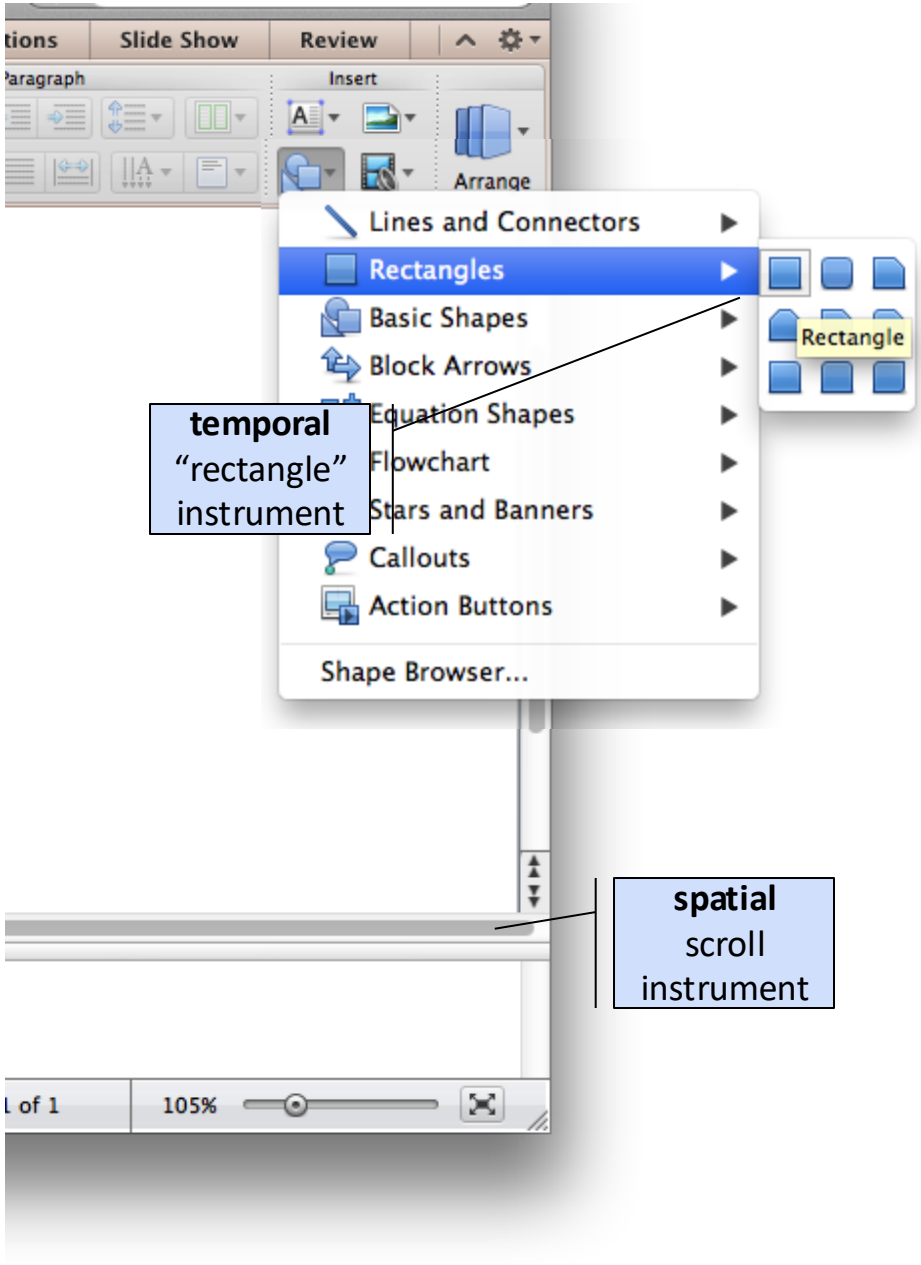
"A model of interaction based on how we naturally use tools (or instruments) to manipulate objects in the physical world."

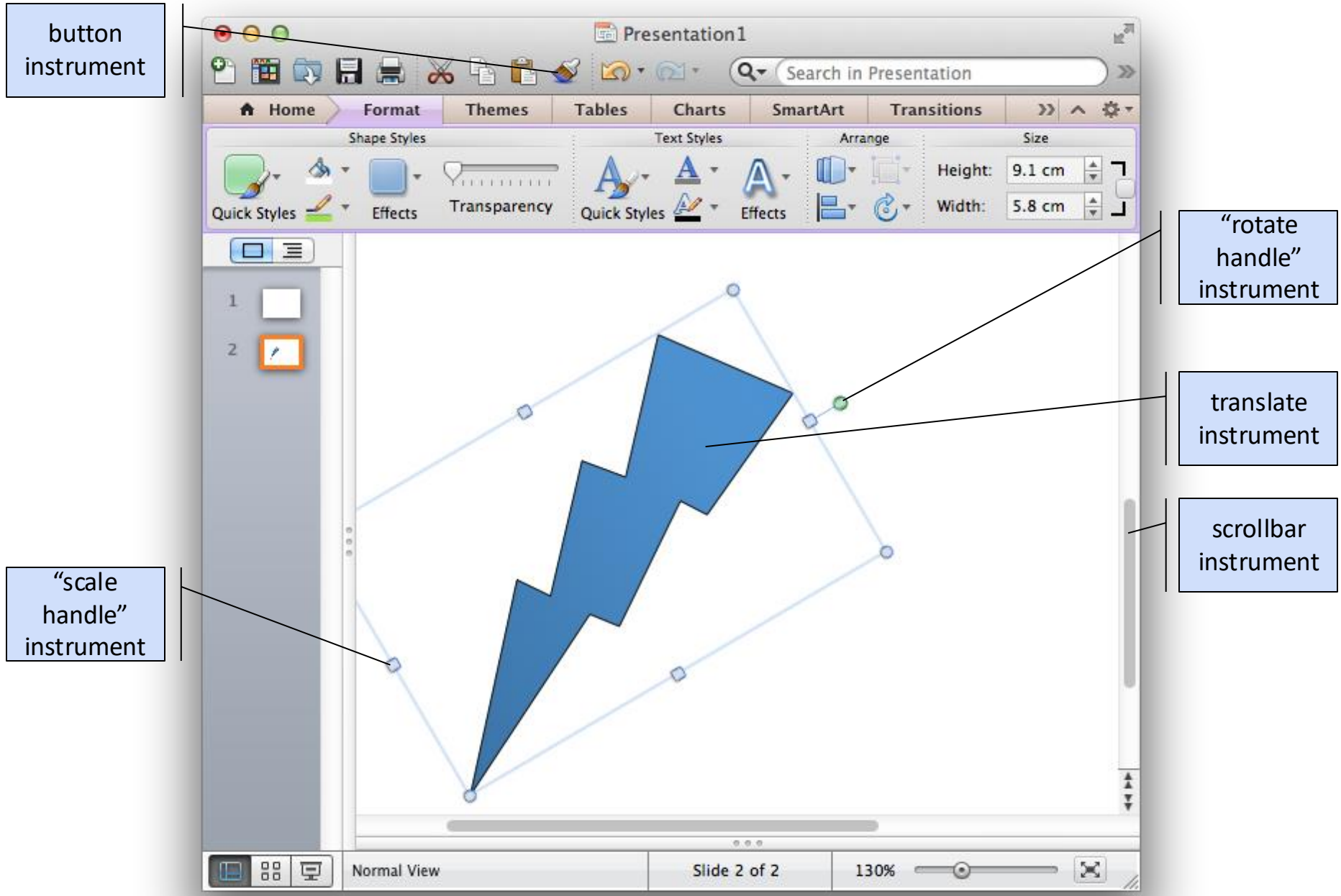
- Interfaces have *interaction instruments* and *domain objects*
  -
  -

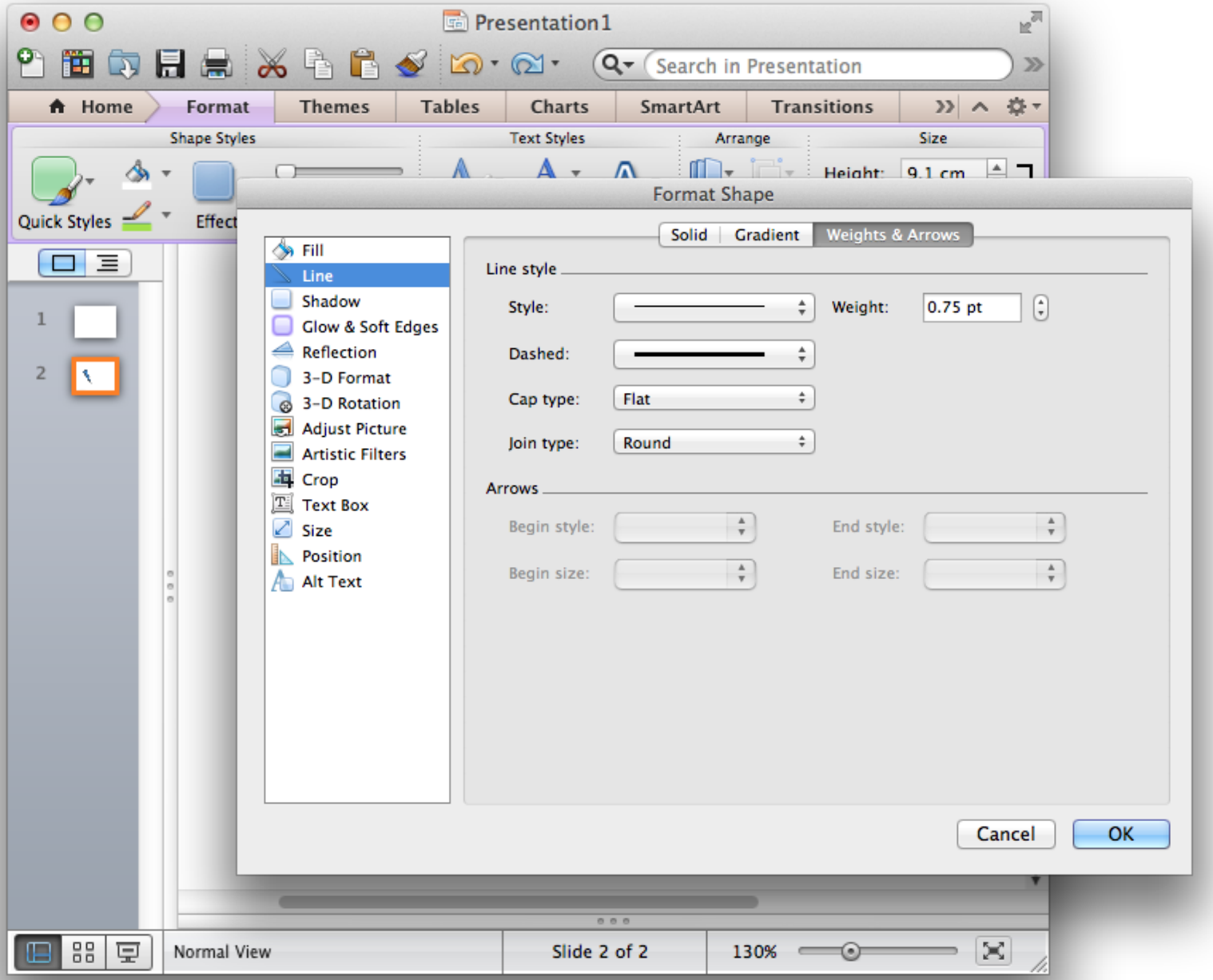

Presentation1
Search in Presentation
Home
Format
Themes
Tables
Charts
SmartArt
Transitions
Slides
Font
Paragraph
New Slide
Calibri (Body)
18
Normal View
Slide 2 of 2
130%

Instrument Activation

-




button instrument
Presentation1
Search in Presentation
Home
Format
Themes
Tables
Charts
SmartArt
Transitions
Shape Styles
Quick Styles
Effects
Transparency
Text Styles
Quick Styles
Effects
Arrange
Size
Height:
9.1 cm
Width:
5.8 cm
"rotate handle" instrument
translate instrument
scrollbar instrument
"scale handle" instrument
Normal View
Slide 2 of 2
130%

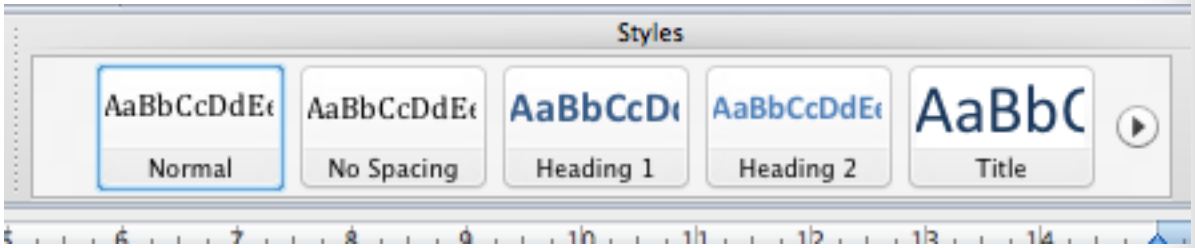

Presentation1
Search in Presentation
Home
Format
Themes
Tables
Charts
SmartArt
Transitions
Shape Styles
Text Styles
Arrange
Size
Quick Styles
Effect
Height:
9.1 cm
Format Shape
Fill
Line
Shadow
Glow & Soft Edges
Reflection
3-D Format
3-D Rotation
Adjust Picture
Artistic Filters
Crop
Text Box
Size
Position
Alt Text
Solid
Gradient
Weights & Arrows
Line style
Style:
Weight:
0.75 pt
Dashed:
Cap type:
Flat
Join type:
Round
Arrows
Begin style:
End style:
Begin size:
End size:
Cancel
OK
Normal View
Slide 2 of 2
130%

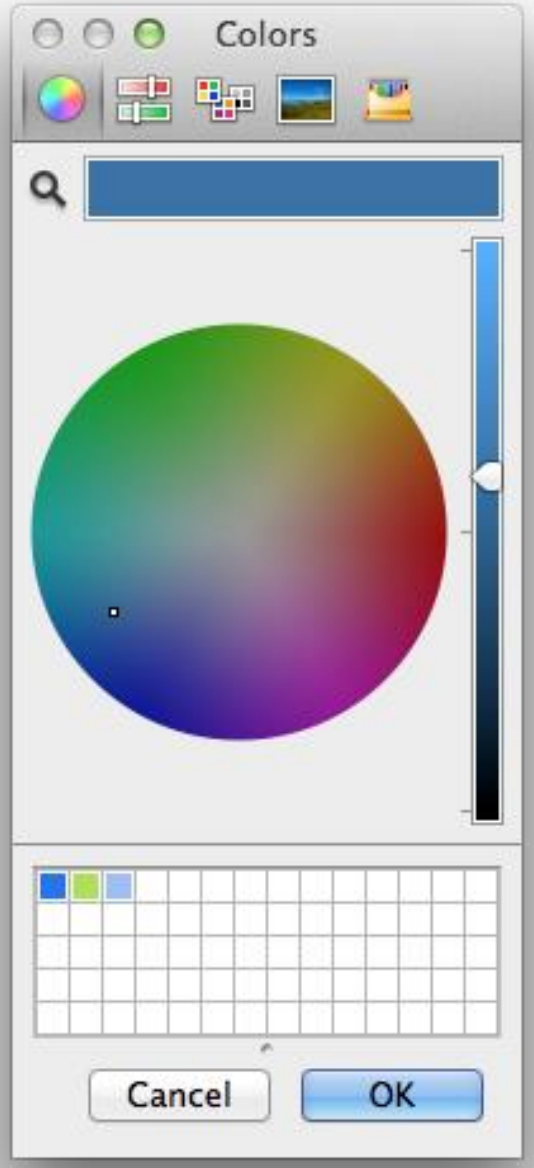
Reification and Meta-Instruments

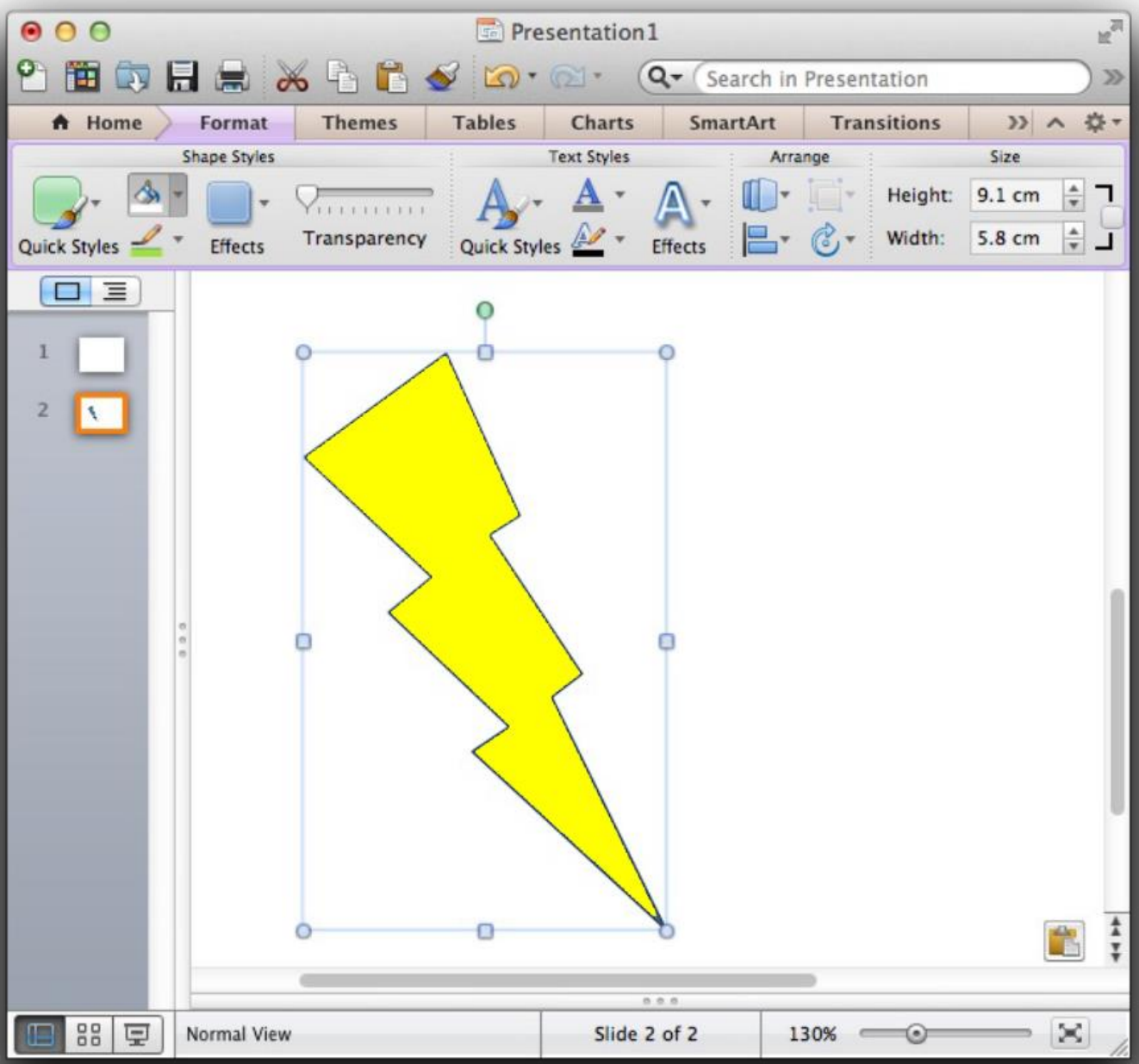
- **Reification:** turning concepts into something concrete
  -
- **Meta-instrument:** an instrument that acts on another instrument
  - the *other instrument* becomes an object of interest
  - analogy:

Object Reification

- Turning attributes of a primary object into other objects of interest






Presentation1
Search in Presentation
Home
Format
Themes
Tables
Charts
SmartArt
Transitions
Shape Styles
Quick Styles
Effects
Transparency
Text Styles
Quick Styles
Effects
Arrange
Size
Height:
9.1 cm
Width:
5.8 cm
1
2
Normal View
Slide 2 of 2
130%

Describing Instruments

Three properties:

- Degree of **indirection**
  -
- Degree of **integration**
  -
- Degree of **compatibility**
  -

Degree of Indirection

Two-dimensional measure of "distance" from instrument to object

- Spatial Dimension
  - e.g. *near*: drag to translate, handle to resize
  - e.g. *medium*: scrollbar near page
  - e.g. *far*: dialog box
- Temporal Dimension
  - e.g. *short*: direct drag response (e.g. drag shape)
  - e.g. med: activate tool in toolbar, start direct manipulation
  - e.g. *long*: using dialog, full drag-and-drop operation

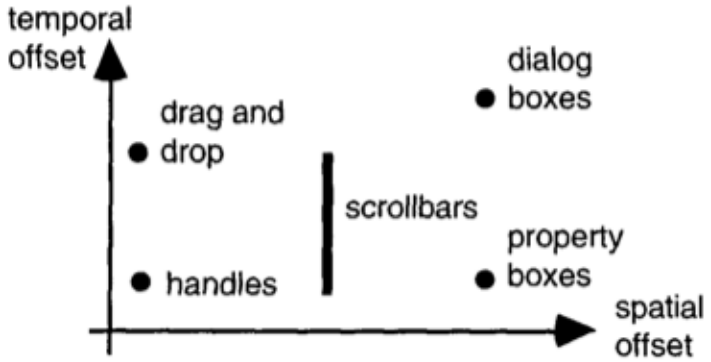


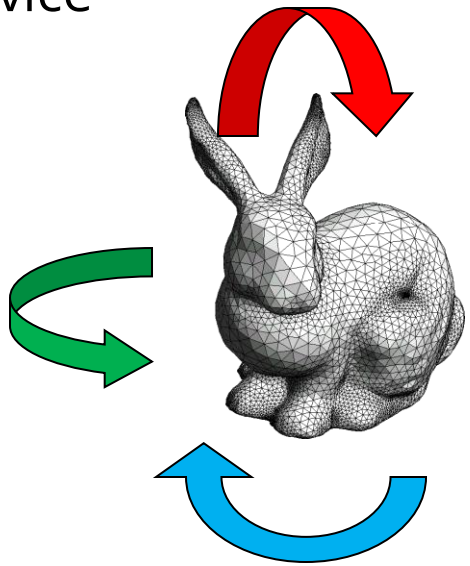
Figure 2: Degree of indirection

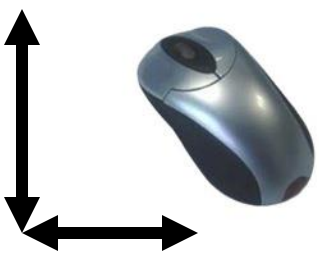
(image: lafon 2000)

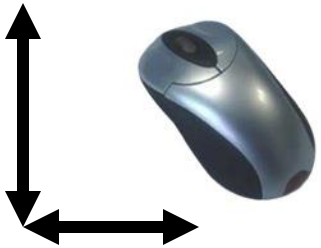
Degree of Integration

The ratio of the degrees of freedom (DOF) of the instrument over the DOF captured by input device.

- Captures the suitability of the device





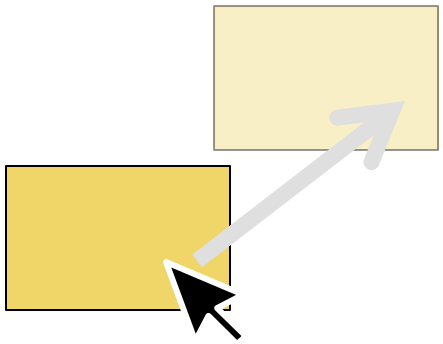


Degree of Compatibility

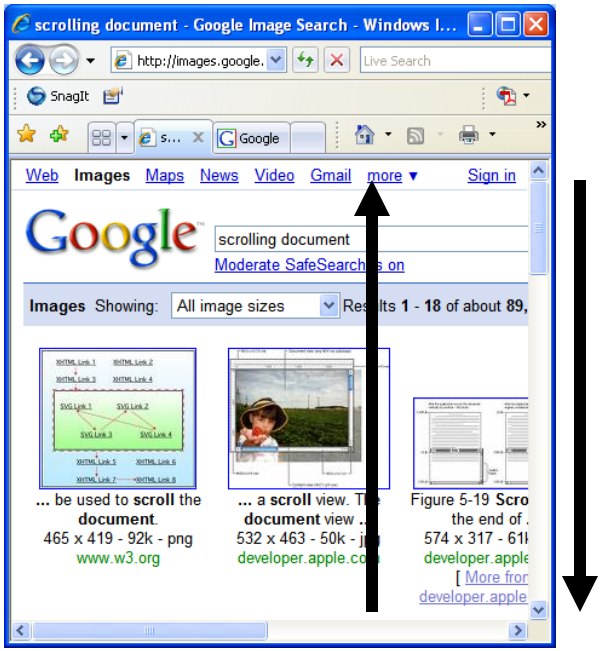
Similarity of *physical actions on instrument* and *response of the object*

- more similar is better since it's more *natural* or *intuitive*

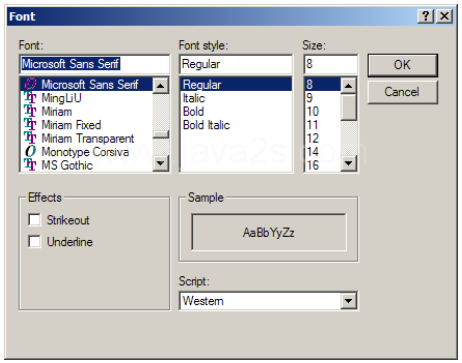
Dragging =



Scrolling =

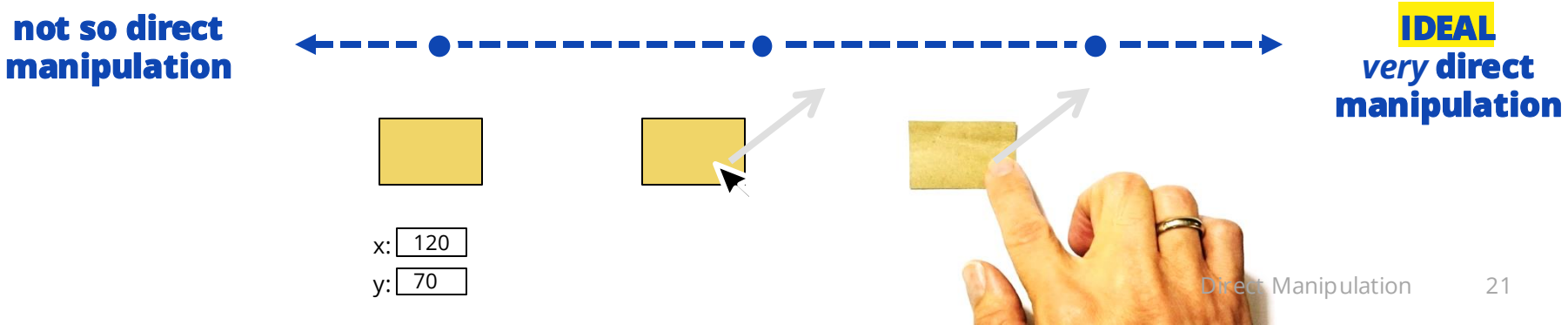


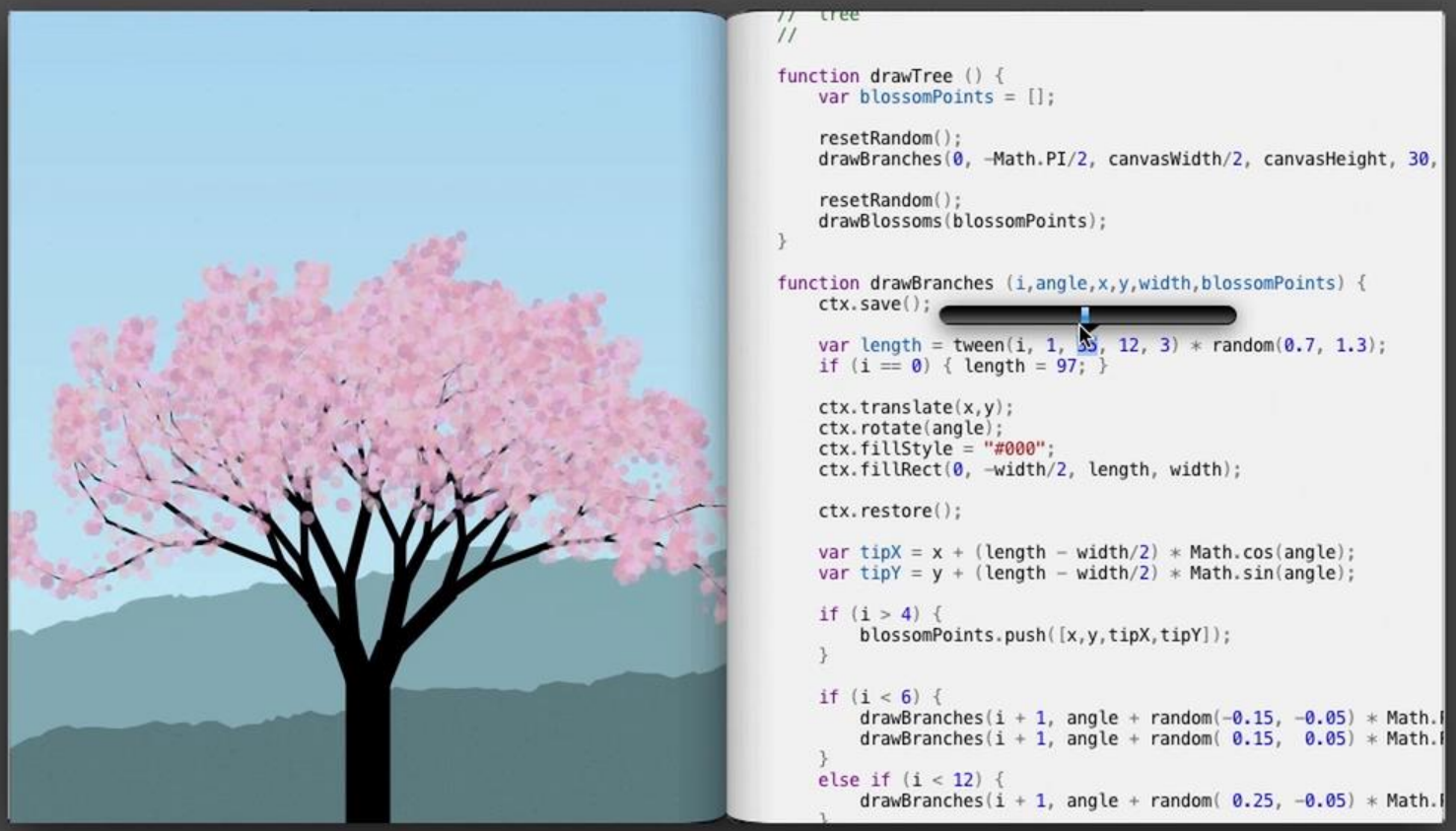
Dialog =



Direct Manipulation as 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)
  - Integration of 1 or less than 1 (higher than 1 means modes)
  - High compatibility (similarity of action and effect)
- Ideal level of direct manipulation is when *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.





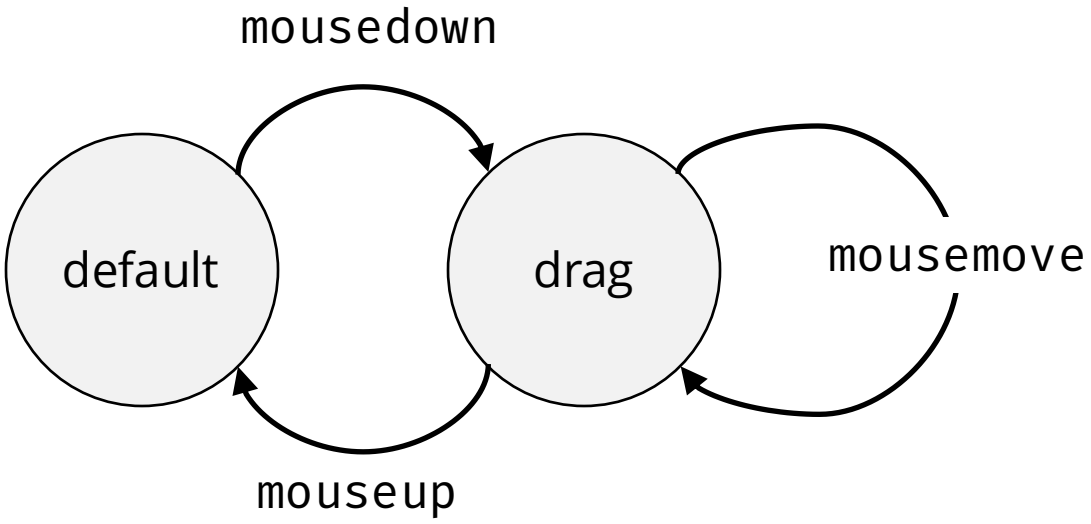
```
//
//

function drawTree () {
    var blossomPoints = [];

    resetRandom();
    drawBranches(0, -Math.PI/2, canvasWidth/2, canvasHeight, 30,

    resetRandom();
    drawBlossoms(blossomPoints);
}

function drawBranches (i,angle,x,y,width,blossomPoints) {
    ctx.save();

    var length = tween(i, 1, 35, 12, 3) * random(0.7, 1.3);
    if (i == 0) { length = 97; }

    ctx.translate(x,y);
    ctx.rotate(angle);
    ctx.fillStyle = "#000";
    ctx.fillRect(0, -width/2, length, width);

    ctx.restore();

    var tipX = x + (length - width/2) * Math.cos(angle);
    var tipY = y + (length - width/2) * Math.sin(angle);

    if (i > 4) {
        blossomPoints.push([x,y,tipX,tipY]);
    }

    if (i < 6) {
        drawBranches(i + 1, angle + random(-0.15, -0.05) * Math.
        drawBranches(i + 1, angle + random( 0.15,  0.05) * Math.
    }
    else if (i < 12) {
        drawBranches(i + 1, angle + random( 0.25, -0.05) * Math.
    }
```

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

- https://youtu.be/PUv66718DII
- https://vault.cs.uwaterloo.ca/s/yNWWBnqRQftzMdt

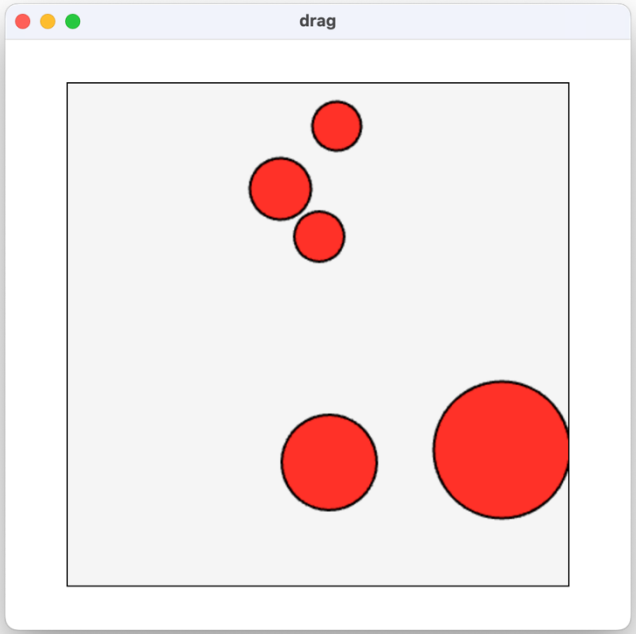
Dragging

- mousedown on shape starts drag
  - calculate *offset* from position to the shape frame-of-reference
- mousemove to drag
- mouseup to end drag



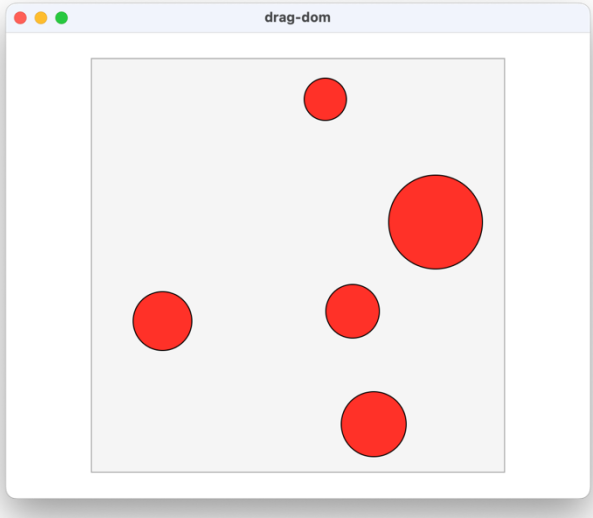
drag

- Similar to A1 SimpleKit architecture, but using HTML DOM
  - Circle Drawable
  - draw loop
  - makeDraggable function
- Implementation is incomplete, need to fix 3 things:
  1. prevent selecting multiple circles at once
  2. drag “front” circle not “back” one
  3. prevent ”jump” to centering on mouse cursor



drag-dom

- Dragging HTML element
  - `div` as circle (using border-radius: 50% style)
  - style to move element independently
    `position: absolute`
  - left and top are x and y coordinates of top-left "corner" of `div`
- Demo:
  - try listening for `mousemove` and `mouseup` on **circle** `div`
    instead of **parent** div

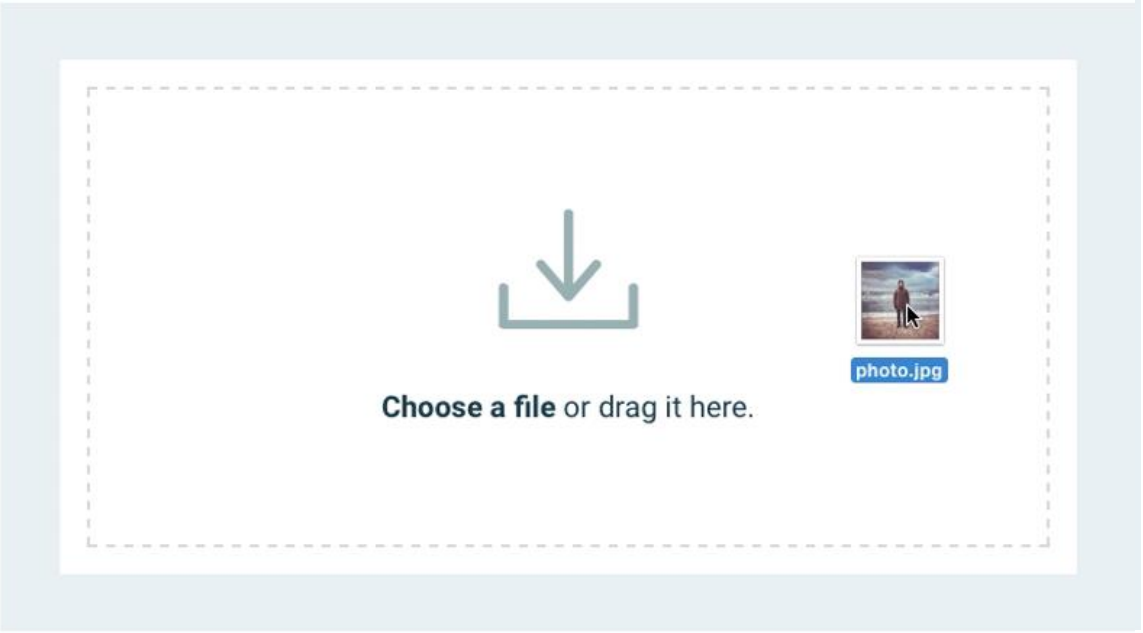


Drag-and-Drop

Most UI Toolkits have built-in support for drag-and-drop

A drag-and-drop interaction is:

1. Press button (`mousedown`) with cursor on source object/data
2. Drag source object onto target object
3. Release button (`mouseup`) to "drop" object/data on target

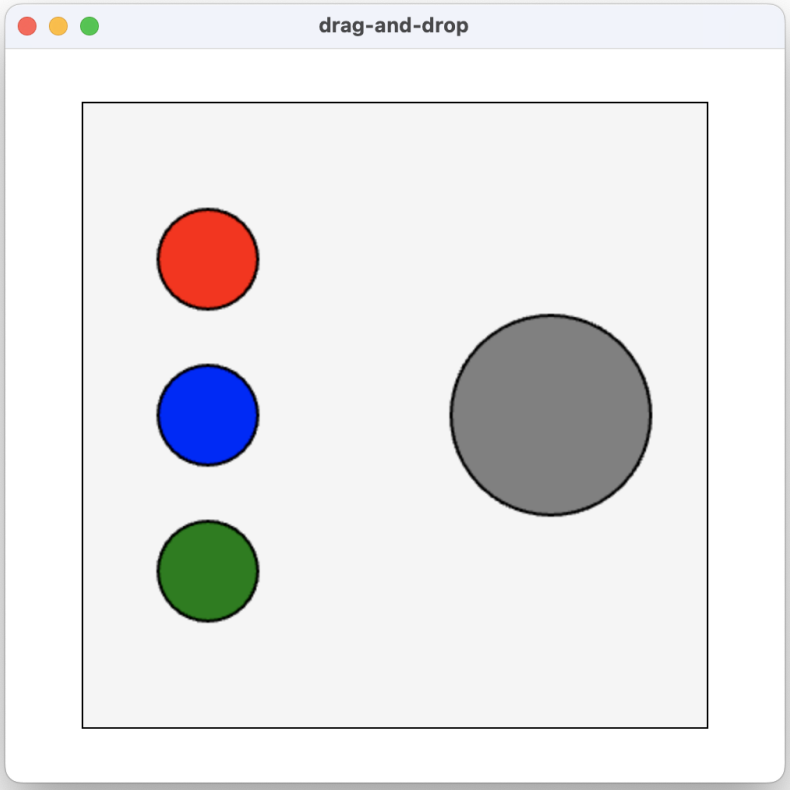


dragging a file to upload is a common drag-and-drop interaction

GIF from https://css-tricks.com/drag-and-drop-file-uploading/

Drag-and-drop

- Extension of makeDraggable to makeDragAndDroppable



HTML Drag-and-Drop API

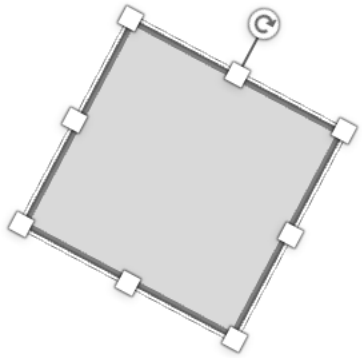
- Enables drag-and-drop interactions in browser applications
  - make element draggable with attribute
    `<div draggable="true" ...`
  - manage a "drag operation" by handling drag-and-drop events:
    `dragstart, drag, dragenter, dragleave, dragover, drop, dragend`
  - define what information is manipulated
- Direct manipulation feedback is important
  - visual indication of elements that can be dragged
  - feedback showing possible drop targets

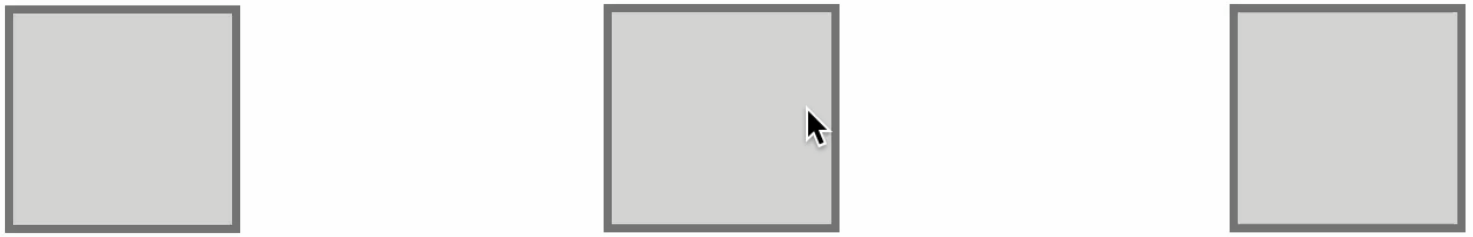
https://developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API#specifications

Transformable Shape

- Direct manipulation widgets to transform an object:
  - Scale by dragging *corner and side handles*
  - Rotate by dragging *rotation handle*
  - Translate by dragging shape

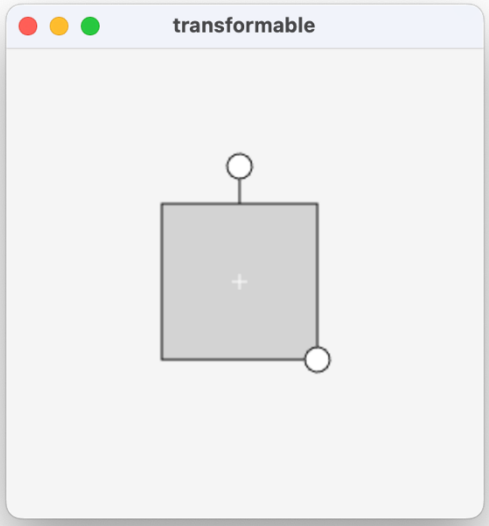
"handle" is a draggable widget





transformable (main structure)

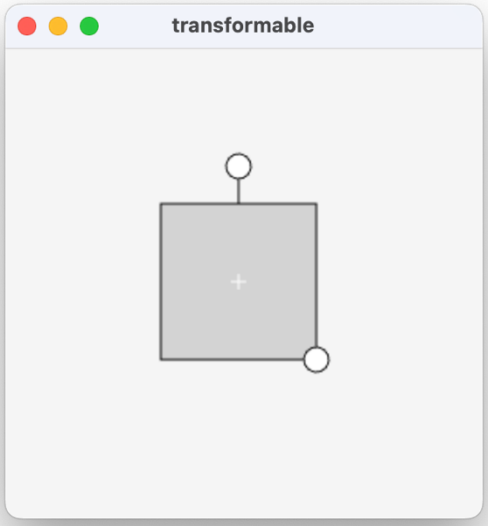
- Main code
  - Fullscreen canvas with draw loop (like SimpleKit)
  - Create Transformable shape and draw it
- Transformable shape class
  - Draw method (draws at current position, size, angle)
  - Three listeners: mousedown, mousemove, mouseup
  - Property to track manipulation mode (idle, translate, scale, rotate)



transformable (shape local coordinate frame)

- The shape has a *local coordinate frame*
  - Centred at (0,0) with rotation 0
- To draw shape, translate it by (x,y) and rotate it by angle

```
gc.translate(x, y);
gc.rotate(angle);
this.drawShape(gc, width, height)
```

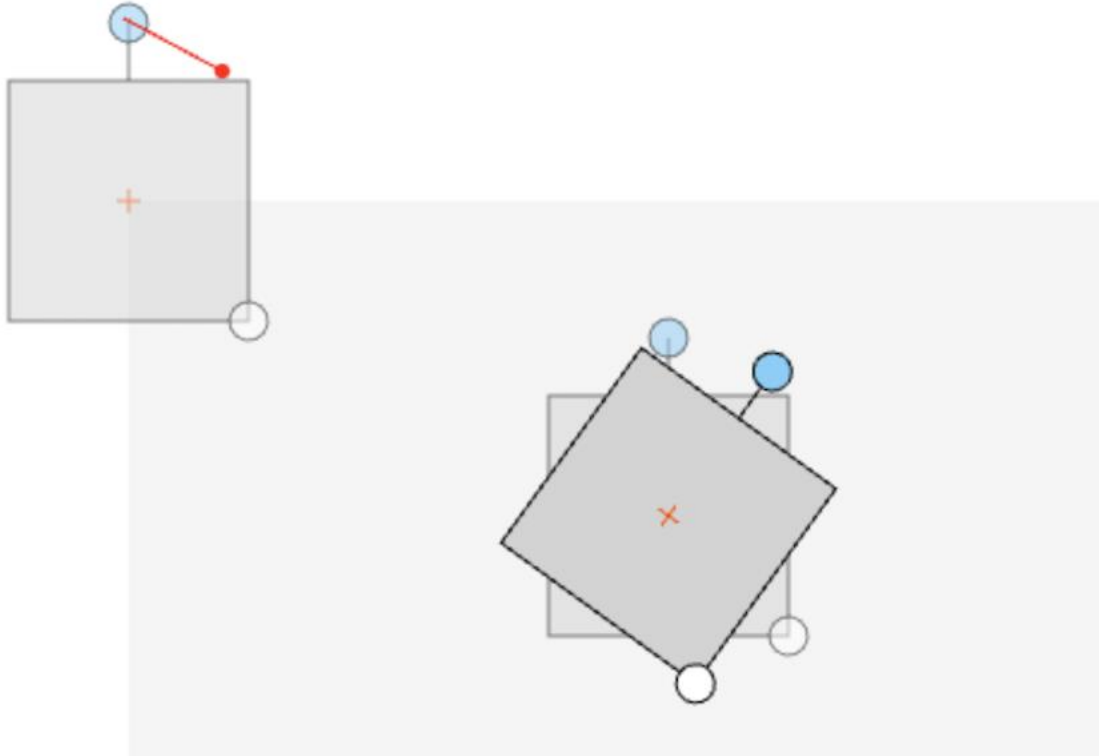


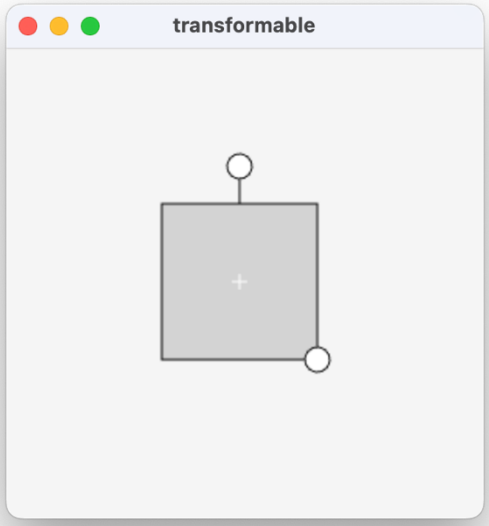
transformable (mouse transformations)

- *Hit-testing* and *dragging* are performed in shape coordinates
  - Must transform mouse position to shape coordinates

```
[mx, my] = this.transformMouseToShapeCoord(_mx, _my);
```

- DEMO: set demo.on to true
  - see shape and mouse in local shape coordinates
  - Visualize drag delta as red line





Exercise 1

- Re-implement the drag-canvas demo in Preact
  - Start from the canvas demo from last lecture

Exercise 2

EXERCISE

- Extend the `Transformable` shape to **have resize handles on each corner and each side**.
- *Hint:* create an object/class with parameters for each handle, e.g.
  - Relative position
  - If currently being dragged
  - How to adjust width and height to "pin" opposite corner/side