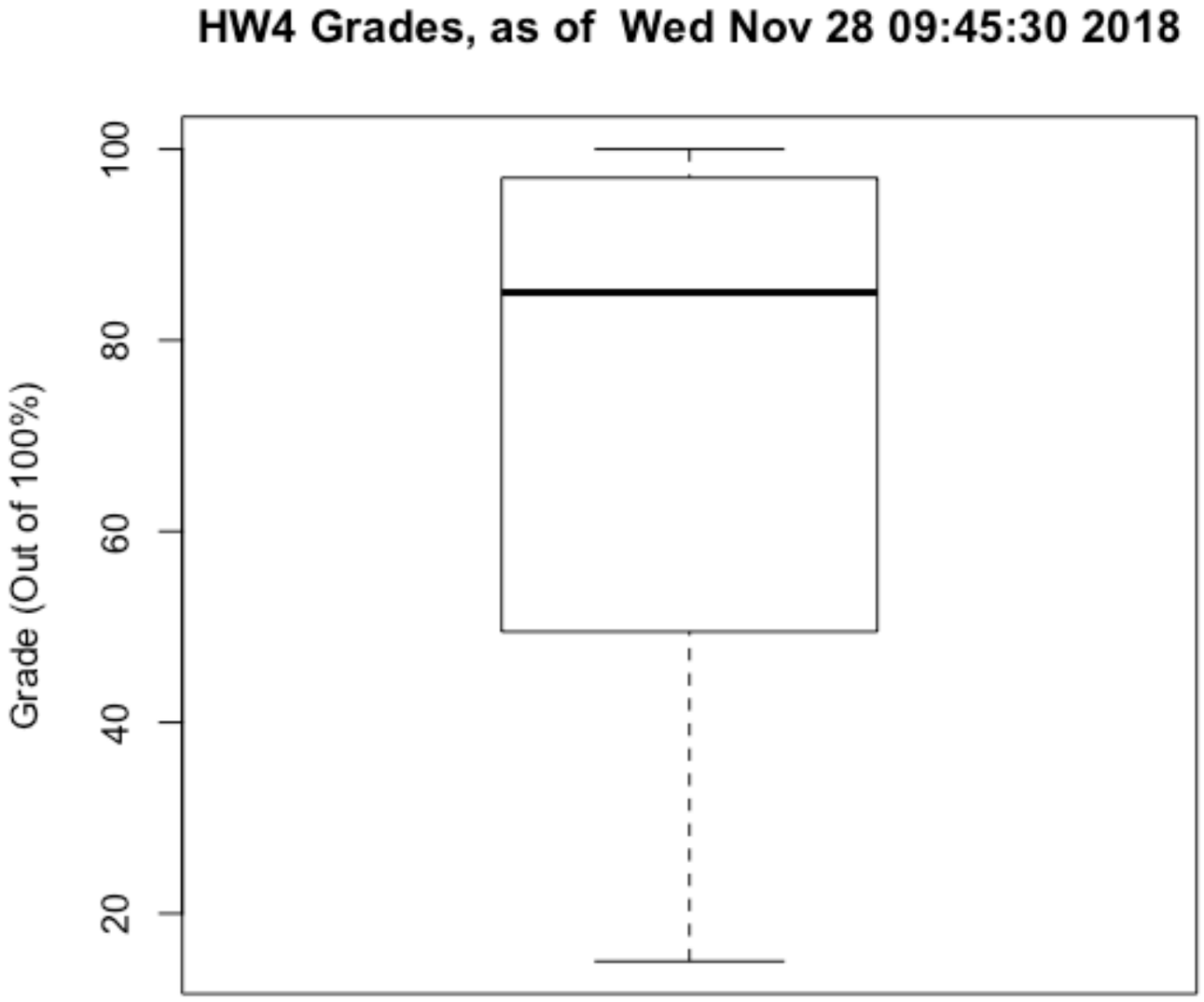
Interaction Techniques

SWE 432, Fall 2018

Web Application Development

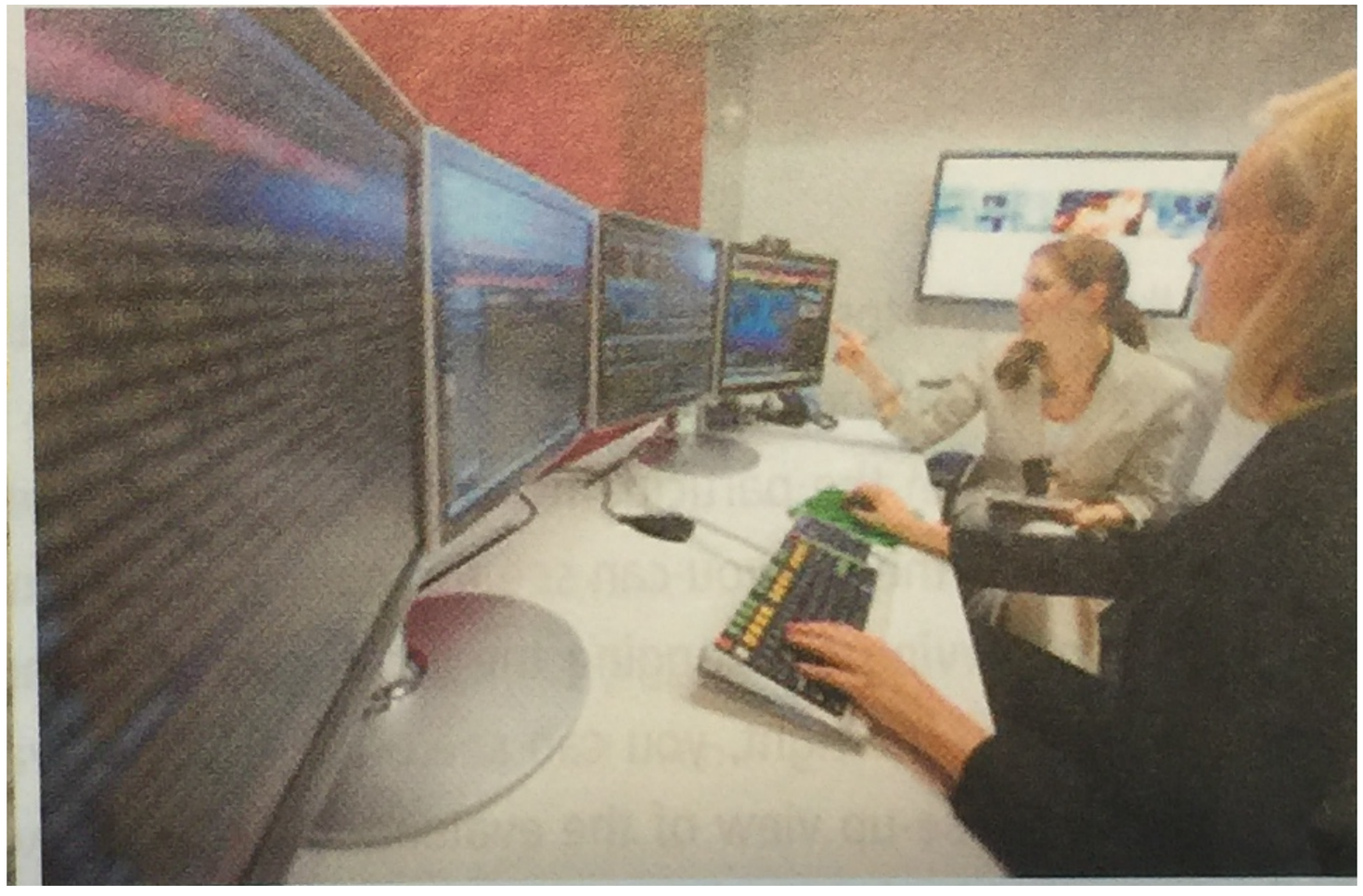
GEORGE MASON UNIVERSITY

HW4 Graded


HW4 Grades, as of Wed Nov 28 09:45:30 2018
Grade (Out of 100%)
100
80
60
40
20

Review: Why conduct usability studies

- Evaluate interaction design with **real** empirical data, gathering ground truth of user performance
- Identify **usability issues**



Review: Data collection

- Think aloud
- Screencast
- Questionnaires or interview questions to gather participant feedback

Review: Piloting study design

- Dress rehearsal for conducting actual study
- Goals
  - Ensure software / prototype won't "blow up"
  - Test tasks - ensure right length & difficulty
  - Test that materials are comprehensive and comprehensible
  - Test data collection protocol and methods
- As-needed piloting
  - Use first study session as pilot only if issues arise and must be addressed

Today

- What principles guide the design of usable interaction techniques?
  - How can interaction designs help support making plans, taking action, and interpreting feedback?
  - How does a direct manipulation interface make complex tasks easier?

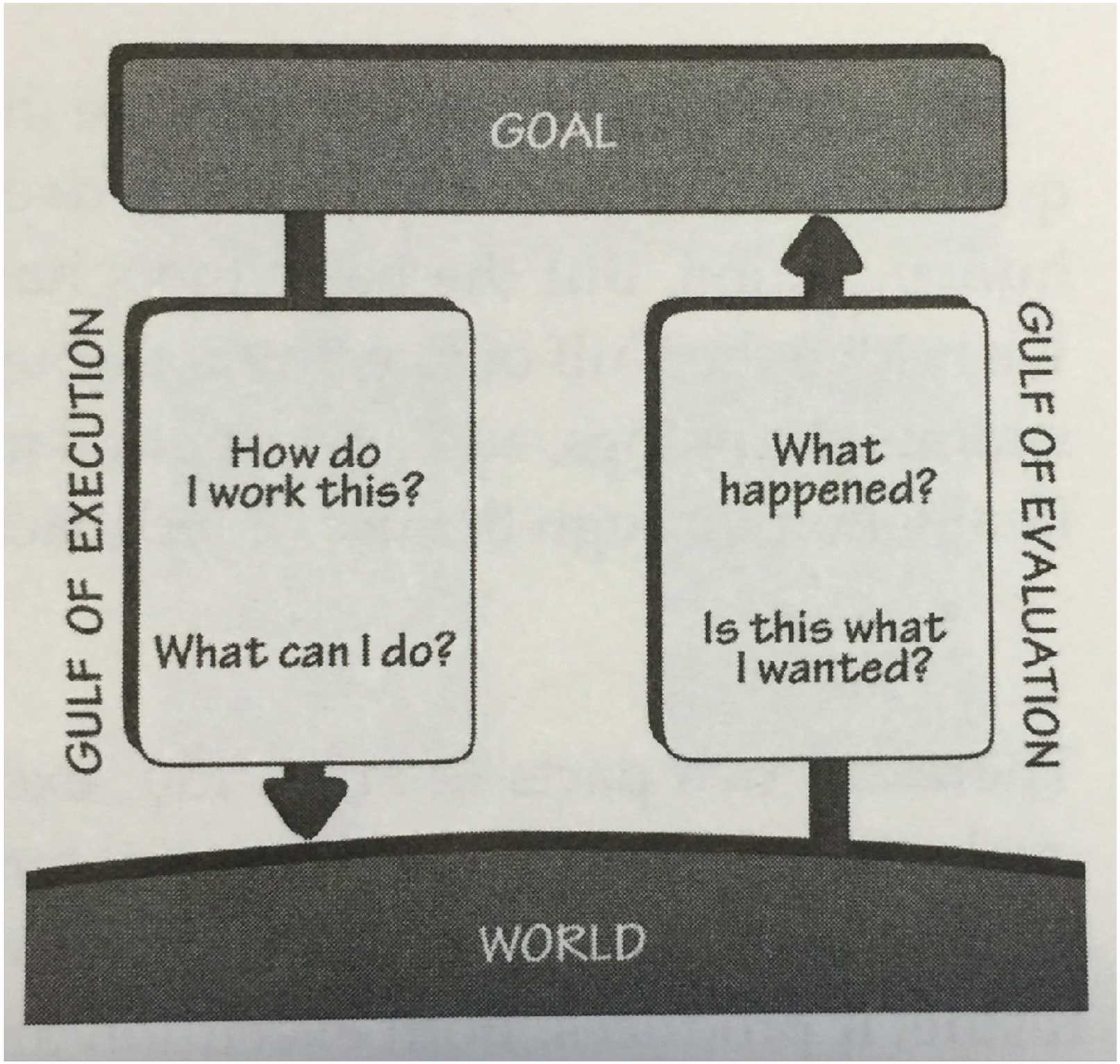
Interaction technique

- A method by which a user can perform an action or sequence of actions with a computer.
- Might encompass **software** (e.g., accelerators on a menu) and/or specialized **hardware** (momentum scrolling on iOS)
- What makes a good interaction technique?
  - Usability: task performance, discoverability, learnability, …

Example: Filtering

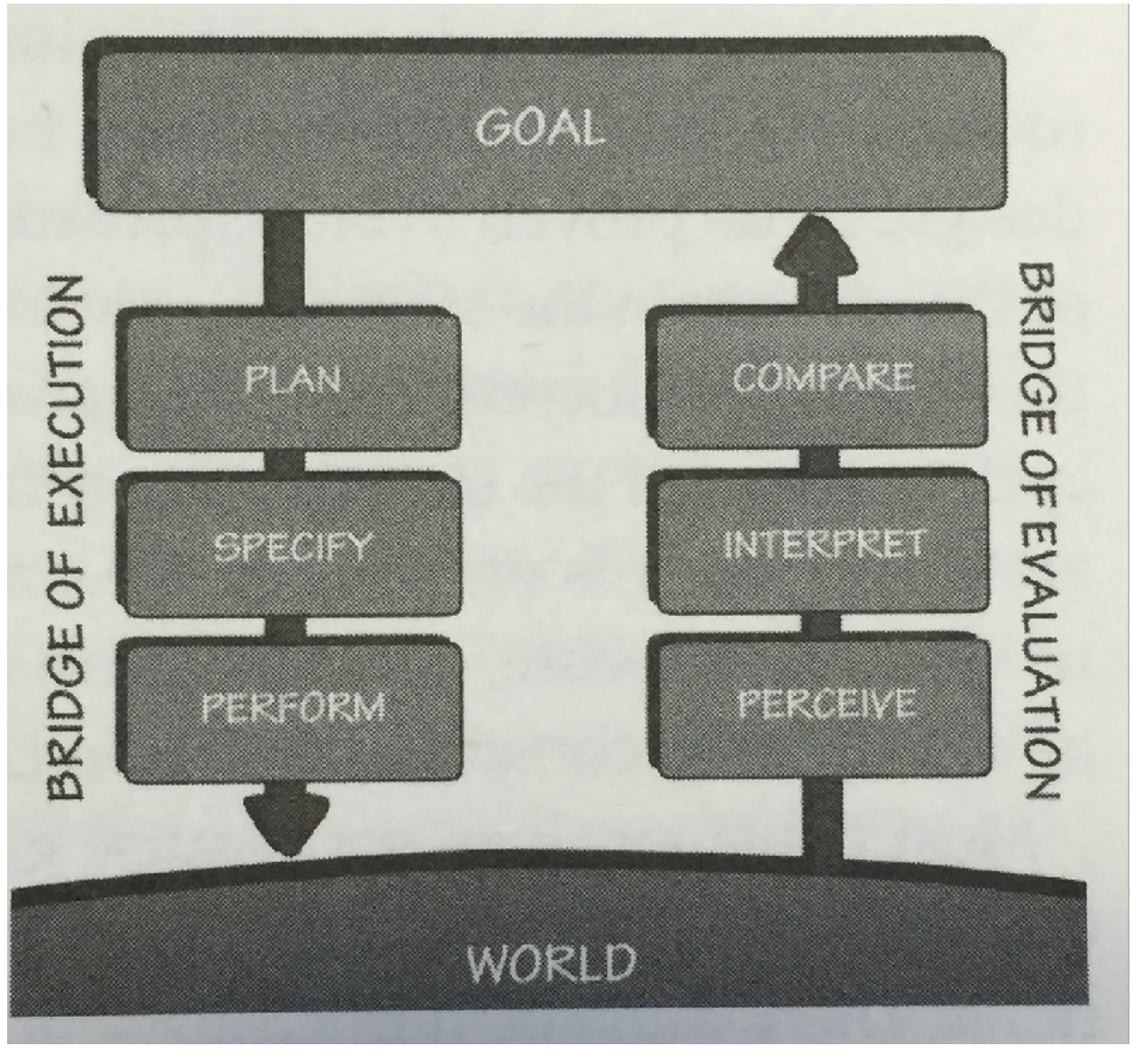
- http://www.kayak.com

Gulfs of execution and evaluation



Norman's 7 stages of action

1. Goal (form the goal)
2. Plan (the action)
3. Specify (action sequence)
4. Perform (action sequence)
5. Perceive (the state of the world)
6. Interpret (the perception)
7. Compare (outcome w/ goal)



Signifiers

- a.k.a “cognitive affordances” [Hartson & Pyla]
- Goals
  - Show which UI elements can be manipulated
  - Show how they can be manipulated
  - Help users get started
  - Guide data entry
  - Suggest default choices
  - Support error recovery

Hinting

- Indicate which UI elements can be interacted with
- Possible visual indicators
  - Static hinting - distinctive look & feel
  - Dynamic hinting - rollover highlights
  - Response hinting - change visual design with click
  - Cursor hinting - change cursor display

Help users predict outcome of actions

- What does this do?
- Should I click it?

Clarity of wording (Example)

- Design for clarity & precision

Implement Function Behavior (10 Pts)

Implement a behavior for this function

← Back | Dispute this test | Inspect code | ▶ Run Tests

STATUS	failed
DESCRIPTION	it should throw an exception if the parameters are invalid
EXECUTION TIME	6ms
MESSAGE	expected 4 to equal 3
DIFF	3 - 4
CODE	1 expect(calculate('+',[1,2])).to.equal(3);

Function Editor

```
11  * @return {Number}
12  */
13 function calculate(command,numbers){
14     if( ['*','/','+','-'].indexOf(command) == -1 )
15         throw 'command not recognized'
16
17     if( !(numbers instanceof Array) || numbers.length === 0)
18         throw 'numbers not valid';
19
20     switch( command ){
21         case '+':
22             var res = sum(numbers[0],numbers[1]);
23             return res;
24         case '*':
25             var res = proc
26             return res;
27         default:
28     }
```

sum(numbers[0],numbers[1]) X
4
stub this function call

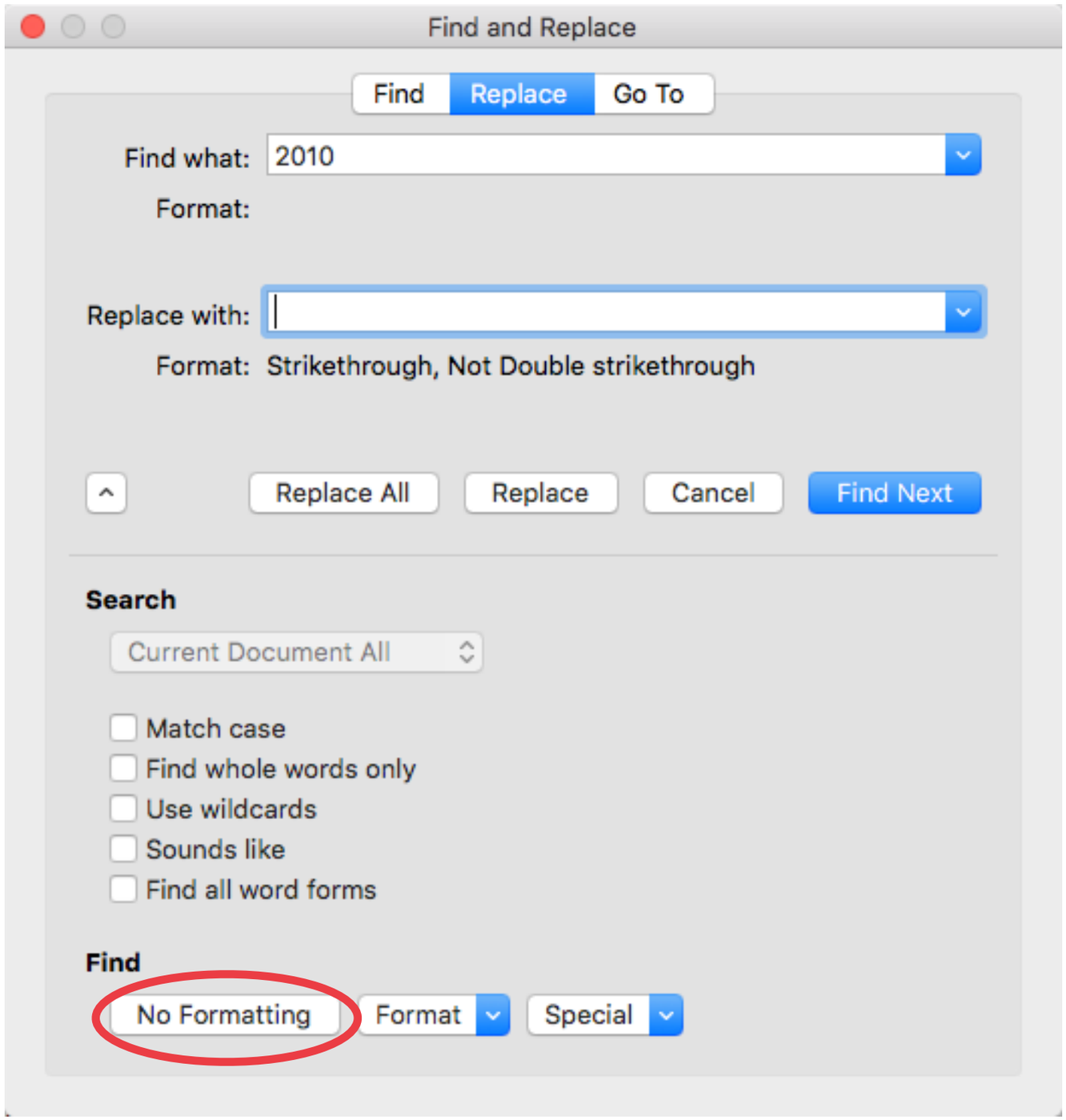
1. Line 15: Missing semicolon.

Clarity of wording

- Choose words carefully
- Speak the user's language
- Avoid vague, ambiguous terms
- Be as specific as possible
- Clearly represent domain concepts

Consistency

- In use of **terms**
  - e.g., do not use “revise” and “edit” interchangeably
- In how commands **map** to UI interactions



Likely & useful defaults

- Default text, if relevant (e.g., date)
- Default cursor position
- Avoid requirements to retype & re-enter data

Avoid using modes

- Modes create inconsistent mapping
  - E.g., control S sometimes saves, sometimes sends email
  - Especially dangerous for frequent interactions that become highly automatic System 1 actions
  - Avoid when possible
  - Clearly distinguish if necessary

Physical actions

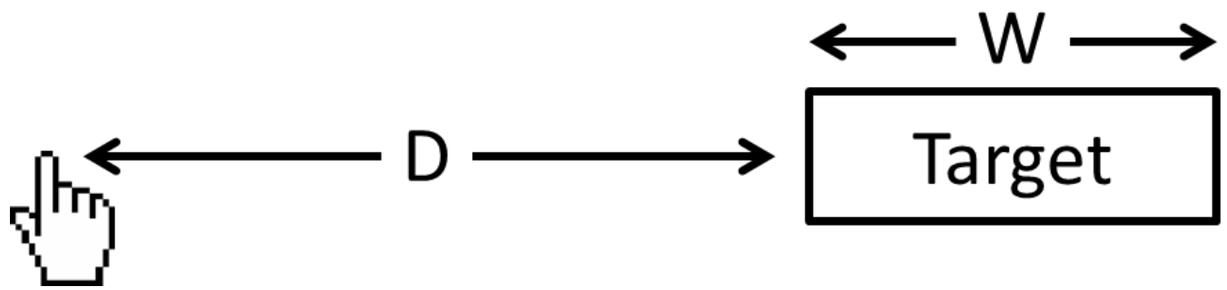
Provide intermediate feedback during interactions

- As user is interacting with objects, provide feedback on interactions
- Examples
  - While dragging object, show new position
  - As selecting text, show selection
  - While clicking on button, show button changing

Avoid physical awkwardness

- Switching between input devices takes time
- Avoid forcing user to constantly switch between input devices (e.g., keyboard & mouse)
  - e.g., Effective tab order between fields
- Avoid awkward keyboard combinations

Fitt's law



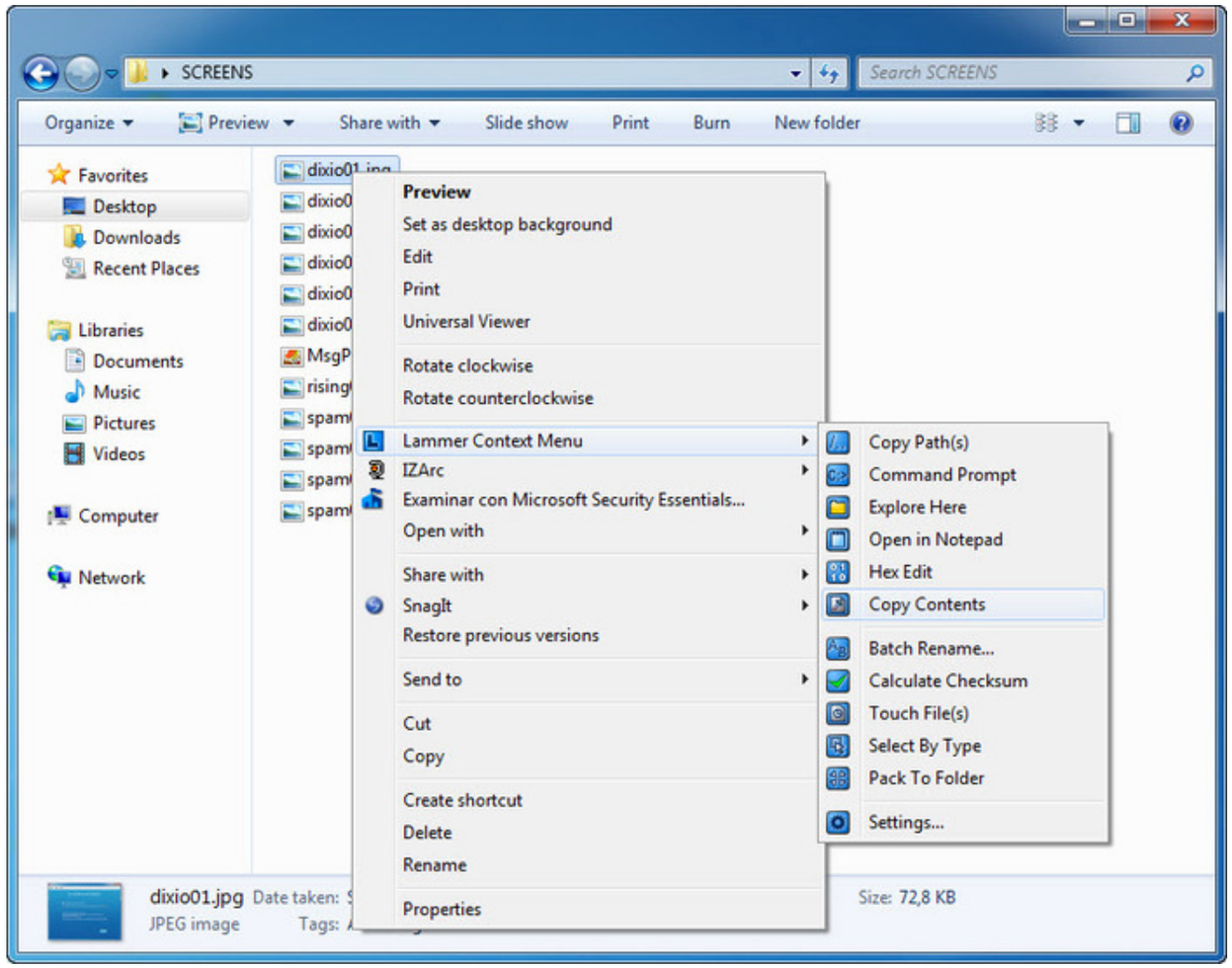
- Time required to move to a target **decreases** with target **size** & **increases** with **distance** to the target
- Movements typical consist of
  - one large quick movement to target (**ballistic** movement)
  - fine-adjustment movement (**homing** movements)
- Homing movements generally responsible for most of movement time & errors
- Applies to rapid pointing movements, not slow continuous movements

Design implications of Fitt's law

- **Constraining** movement to one dimension dramatically increases speed of actions
  - e.g., scroll bars are 1D

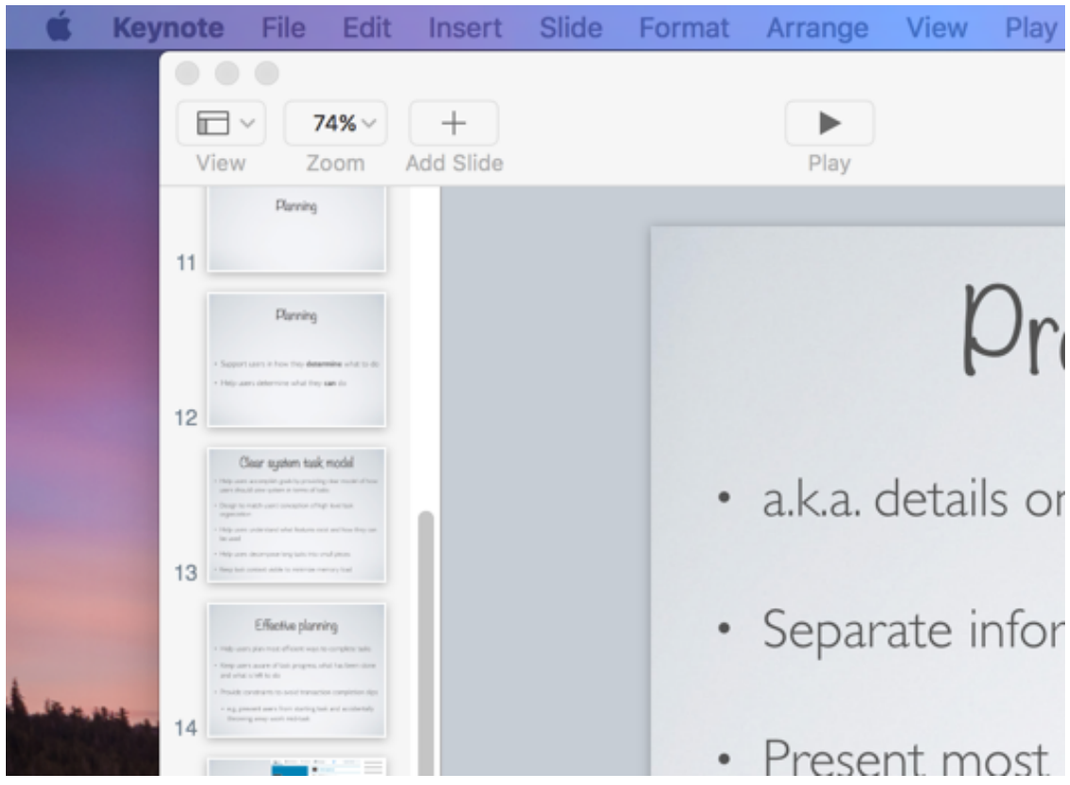
Design implications of Fitt's law

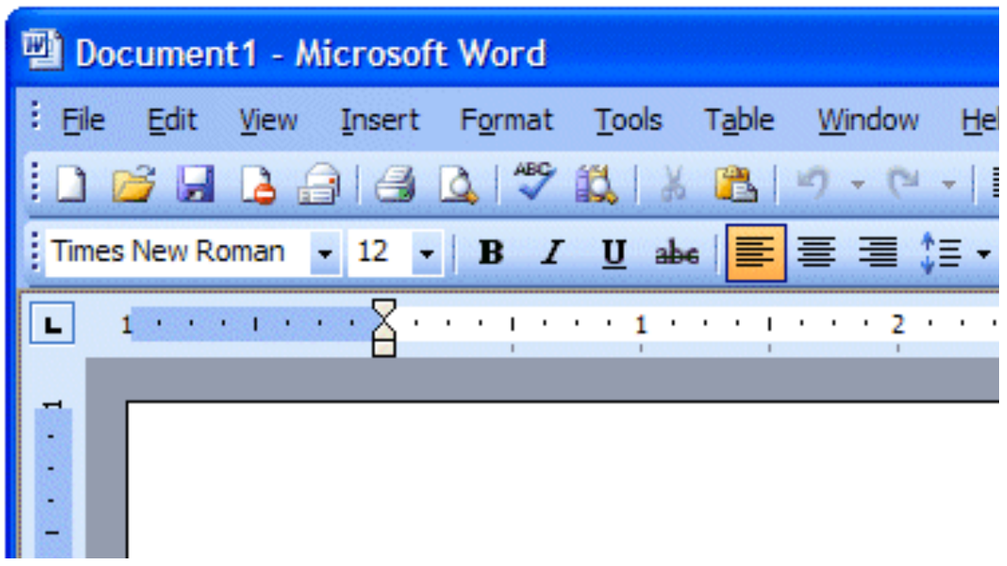
- Making controls **larger** reduces time to invoke actions
- Locating controls closer to user **cursor** reduces time
  - e.g., context menus



Design implications of Fitt's law

- Positioning button or control along **edge** of screen acts as barrier to movement, substantially reducing homing time & errors

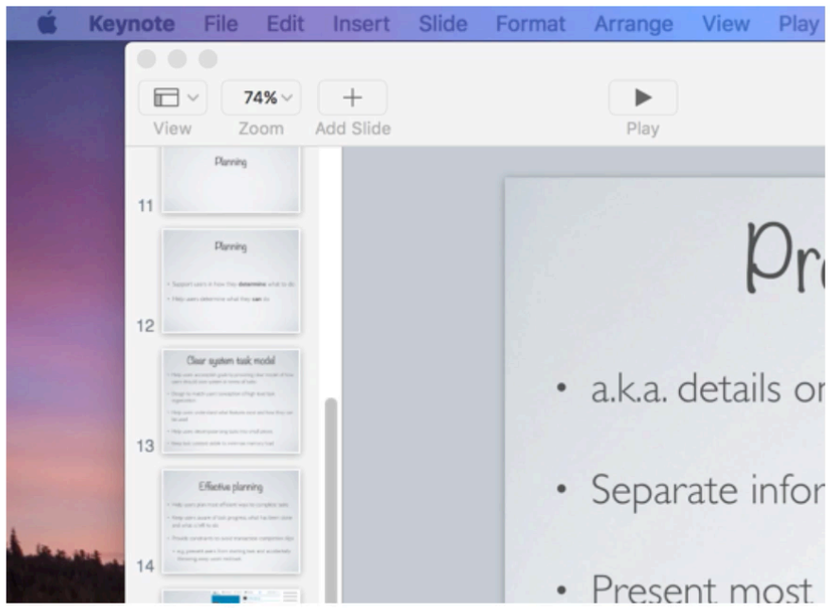


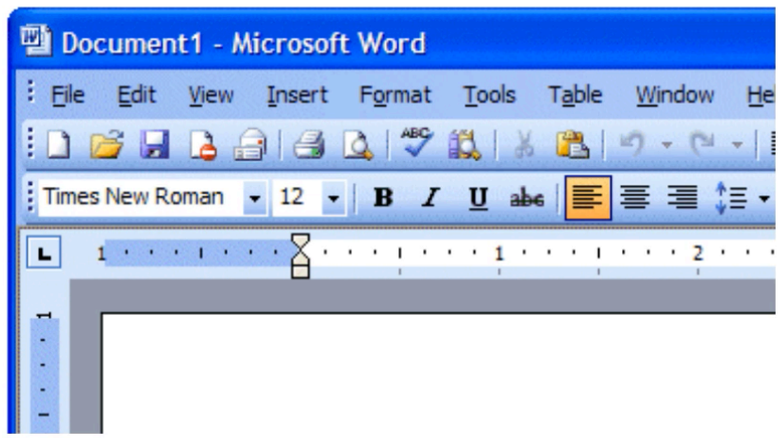


Design implications of Fitt's Law

- Problem: How to benefit from both having menu near the mouse at all times AND have the barriers?

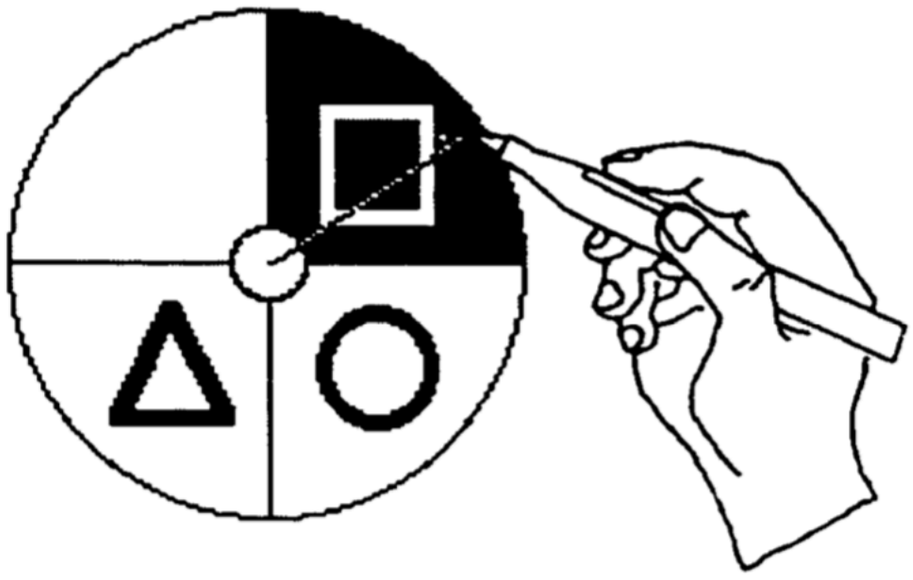
  - Positioning button or control along **edge** of screen acts as barrier to movement, substantially reducing homing time & errors





Alternatives: Pie menu

- Only angle matters, not distance
- Advantages:
  - Faster to select angle than distance
  - Fixed distance to travel
  - Scale independent
- Disadvantages:
  - Angle varies with number of items (can't always have shared items in same place)
  - Large?

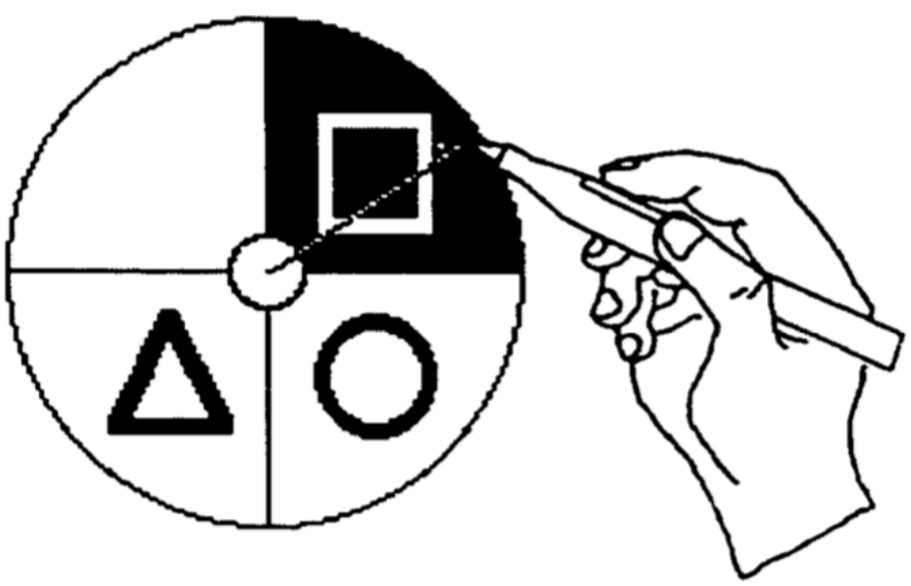


Kurtenbach '93

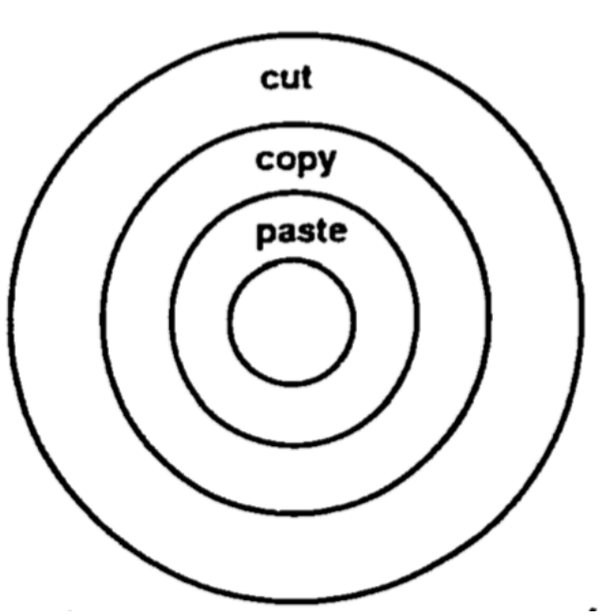
More: https://medium.com/@donhopkins/the-design-and-implementation-of-pie-menus-80db1e1b5293

Alternatives: Bullseye

- Only distance matters, not angle
- Scale dependent

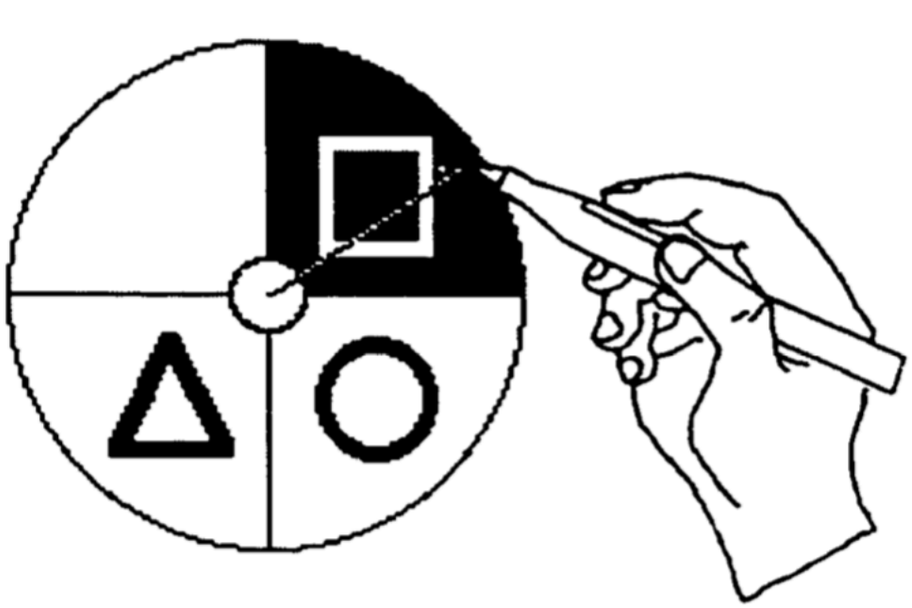


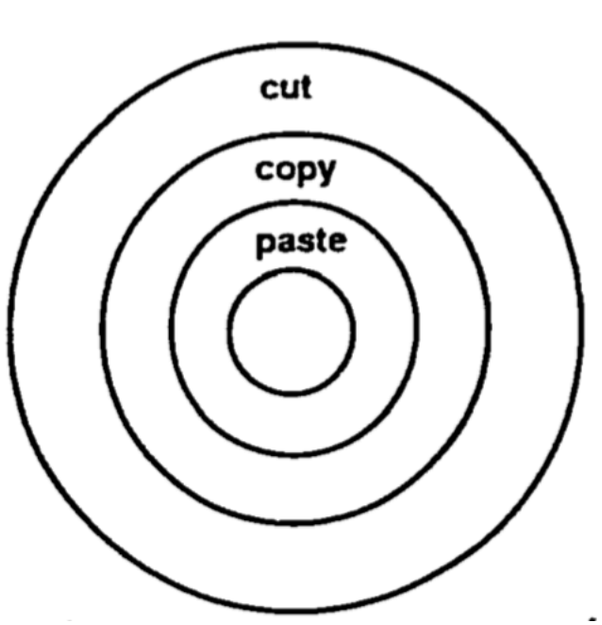
Kurtenbach ‘93

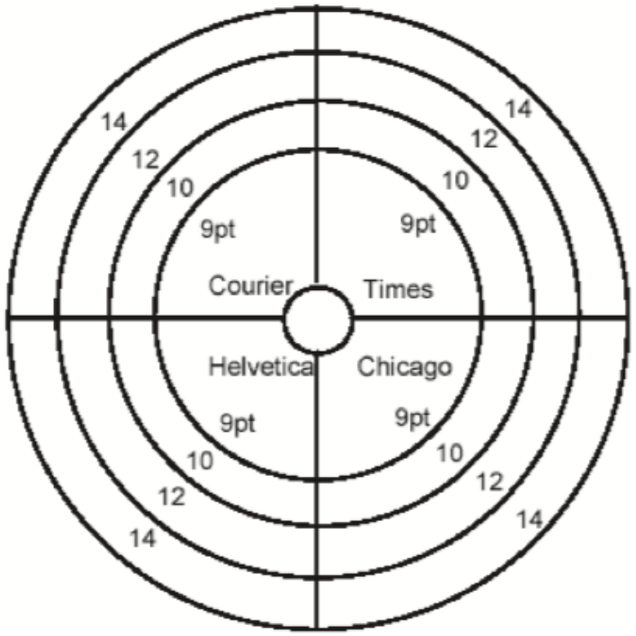


Alternatives: Dart Board

- Angle + Distance



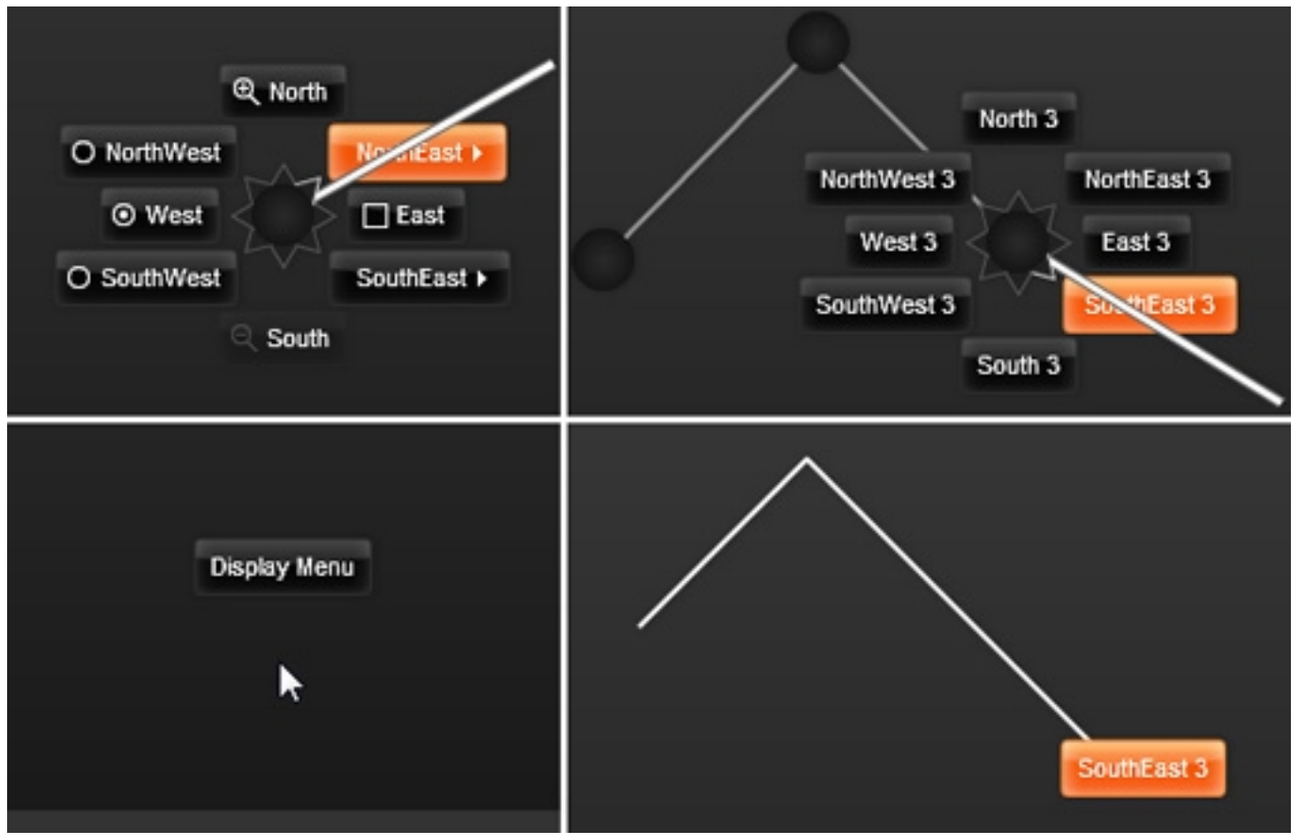




Kurtenbach ‘93

Accelerating Menus - Marking Menus

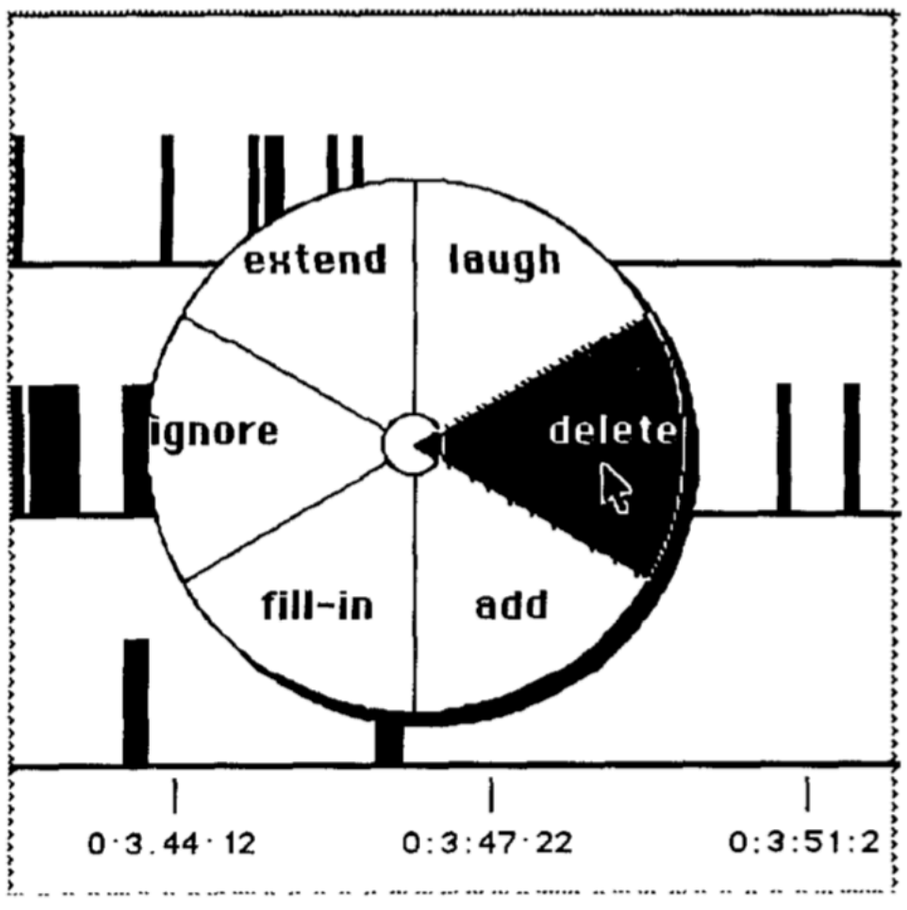
- Idea: focus on the **movement** and not just the visual properties
- Just like the pie, but if you hold mouse and draw, the system interprets as a gesture
- https://www.youtube.com/watch?v=dtH9GdFSQaw



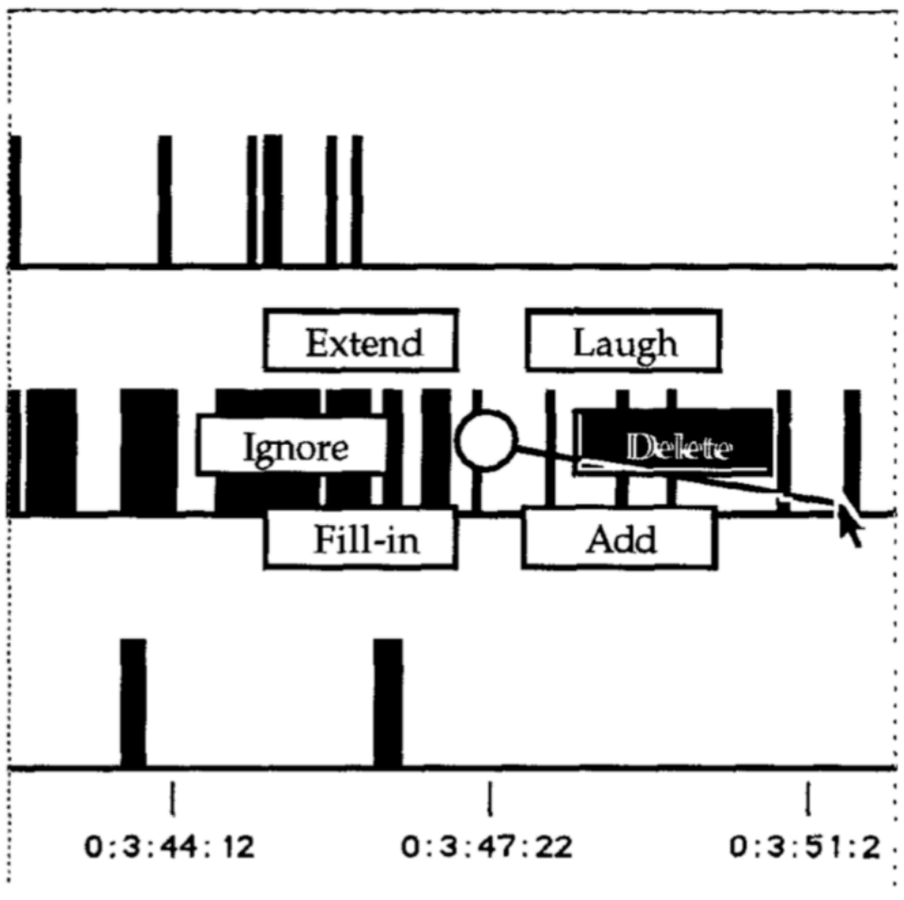
Kurtenbach and Buxton '93

Design Details - Marking Menus

- Principle: Should help user maintain visual context
- Problem: Pies take up space
- Refinement: show only labels



Traditional pies

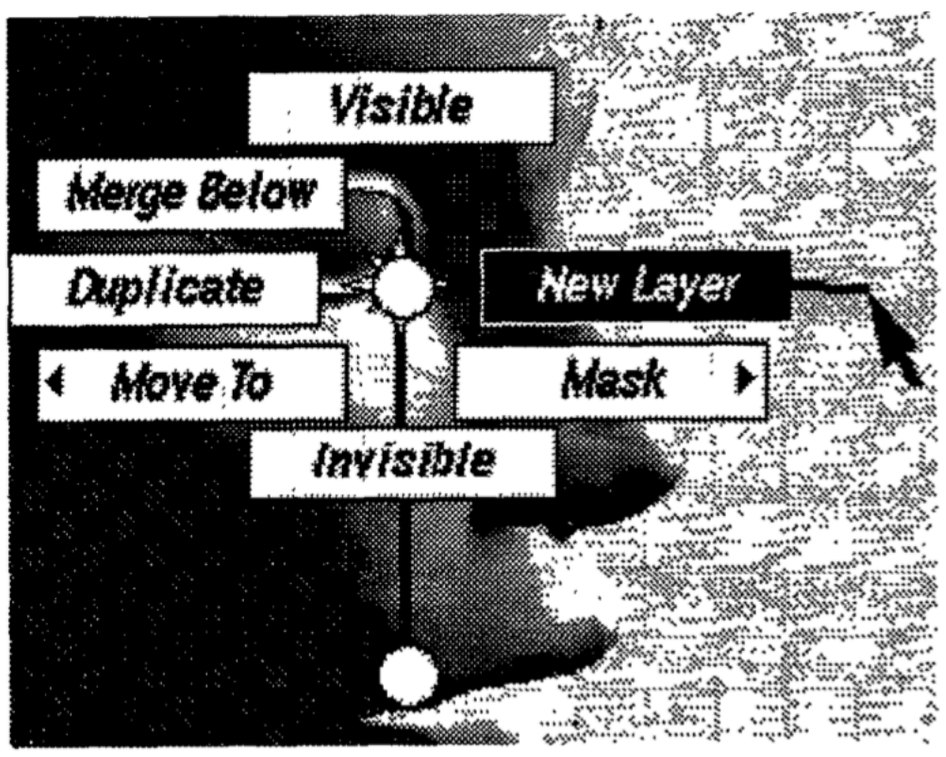


Labels only

Tapia & Kurtenbach ‘95

Design Details - Marking Menus

Principle	Refinment
Maintain visual context	Display only labels Ignore pie wedges Make labels symmetric
Hide unnecessary information	Hide parent menus
Support skill development using graphical feedback	Use eight item menus Use compass star around menu center Show idealized marks



Tapia & Kurtenbach ‘95

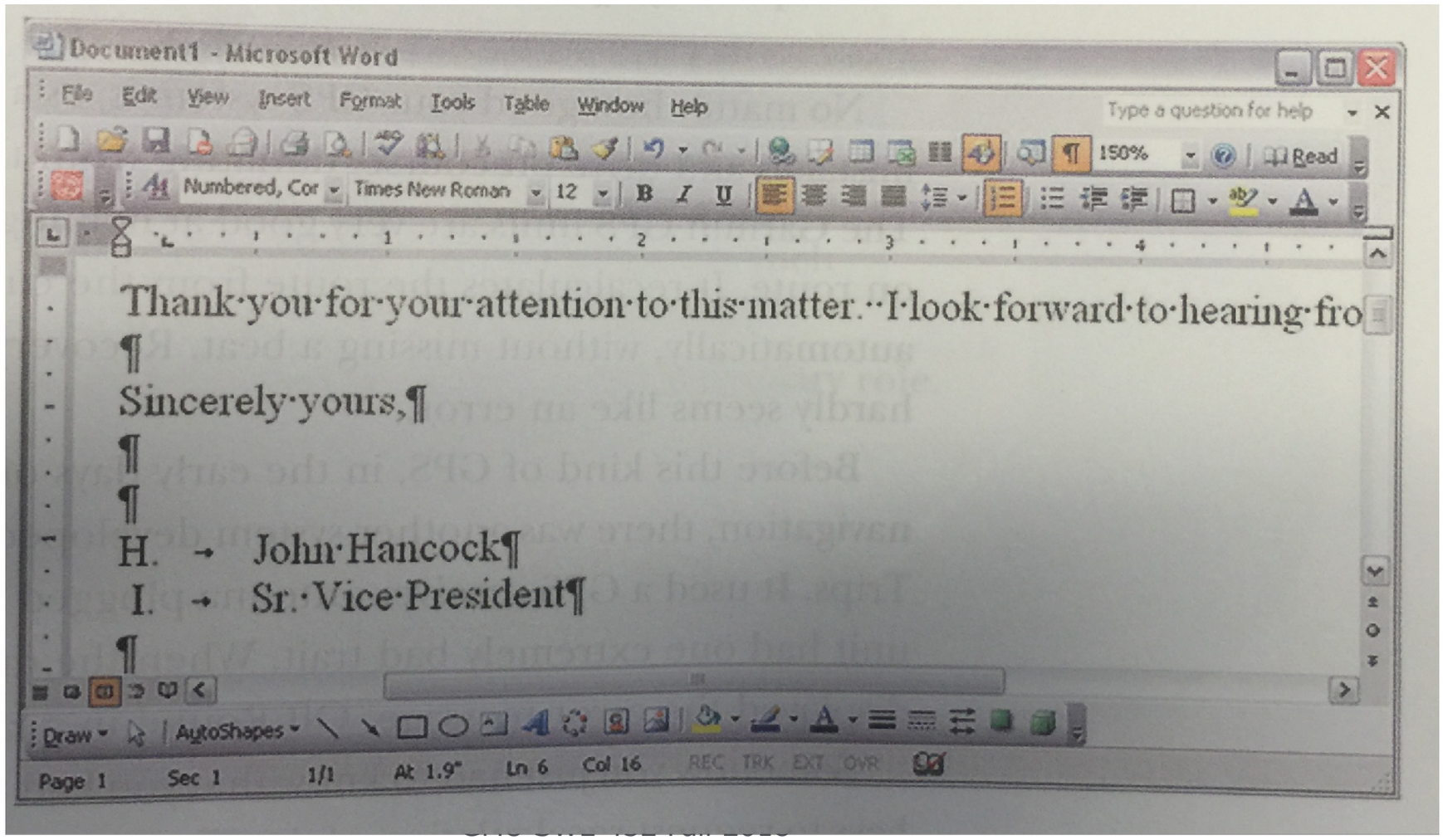
System feedback

System response times

- 0.1second - reacting **instantaneously**
  - requiring no special feedback except displaying result
  - limit for direct manipulation of objects in UI
- 1.0 second - **freely** navigating commands
  - noticeable delay, limit for keeping user's flow of thought uninterrupted
- 10 seconds - keeping users **attention**
  - limit for keeping user's attention focus in UI
  - longer delays create task breaks
- [Nielsen, Usability Engineering, 1993]

Automation

- Keep user in control at highest task levels
- Take control from user when need is obvious & user is busy
- Provide visibility of automation & opportunities to correct when necessary



Provide feedback for all user actions

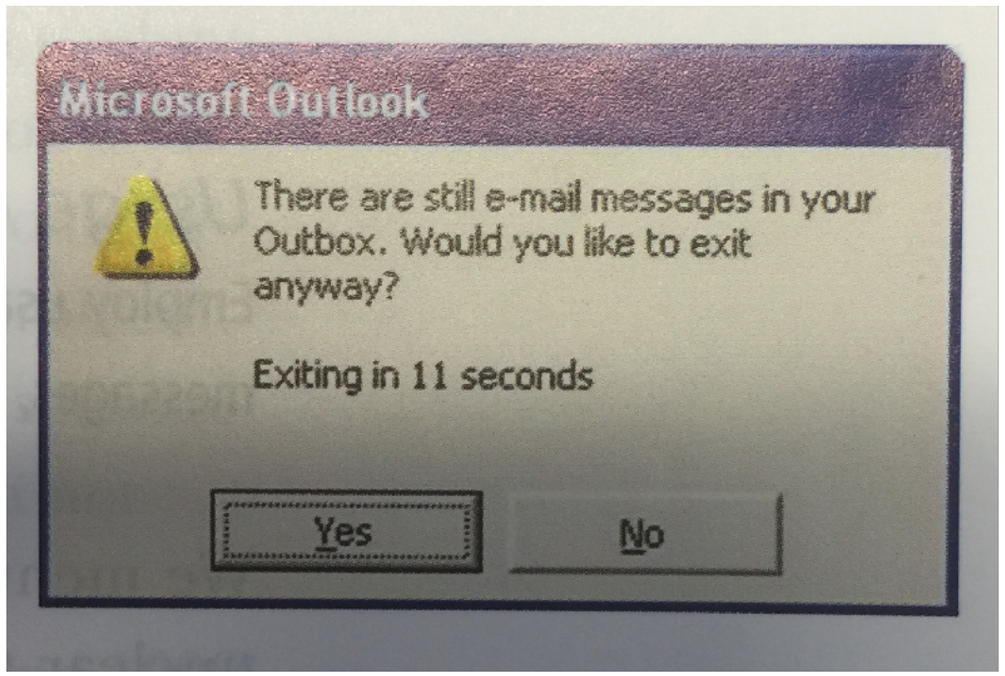
- Feedback helps keep users on track in accomplishing goals
- Request confirmation to prevent costly errors (but use sparingly)
- Make feedback visible, noticeable, legible, located w/ in users focus of attention
- Provide feedback early
- Provide feedback consistently

Tone of feedback

- Establishes relationship with user
- Important not to take user feel “stupid”
- Make the system take blame for errors
- Be positive, to encourage
- Provide helpful messages, not cute messages
- Avoid violent, negative, demeaning, threatening terms (e.g., illegal, invalid)

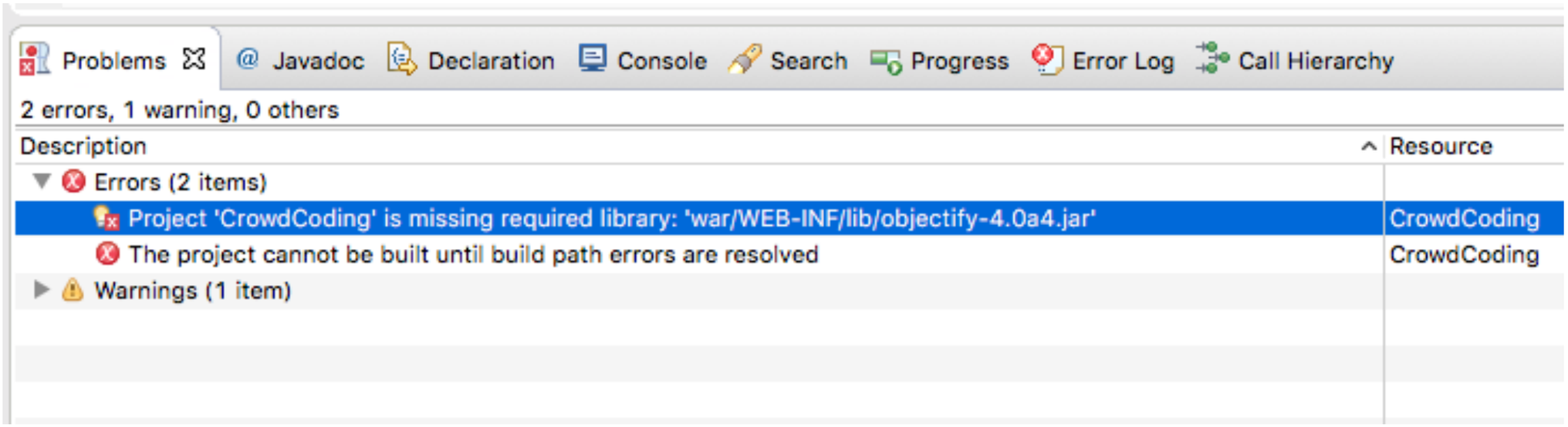
Crafting feedback text

- Clarity - support clear understanding of outcome
- Precise wording
- Completeness - include enough information to fully understand outcomes



Show users how to fix errors

- Good: detecting user errors
- Better: directly showing how errors can be fixed
- (Best: using constraints to prevent errors from ever occurring)



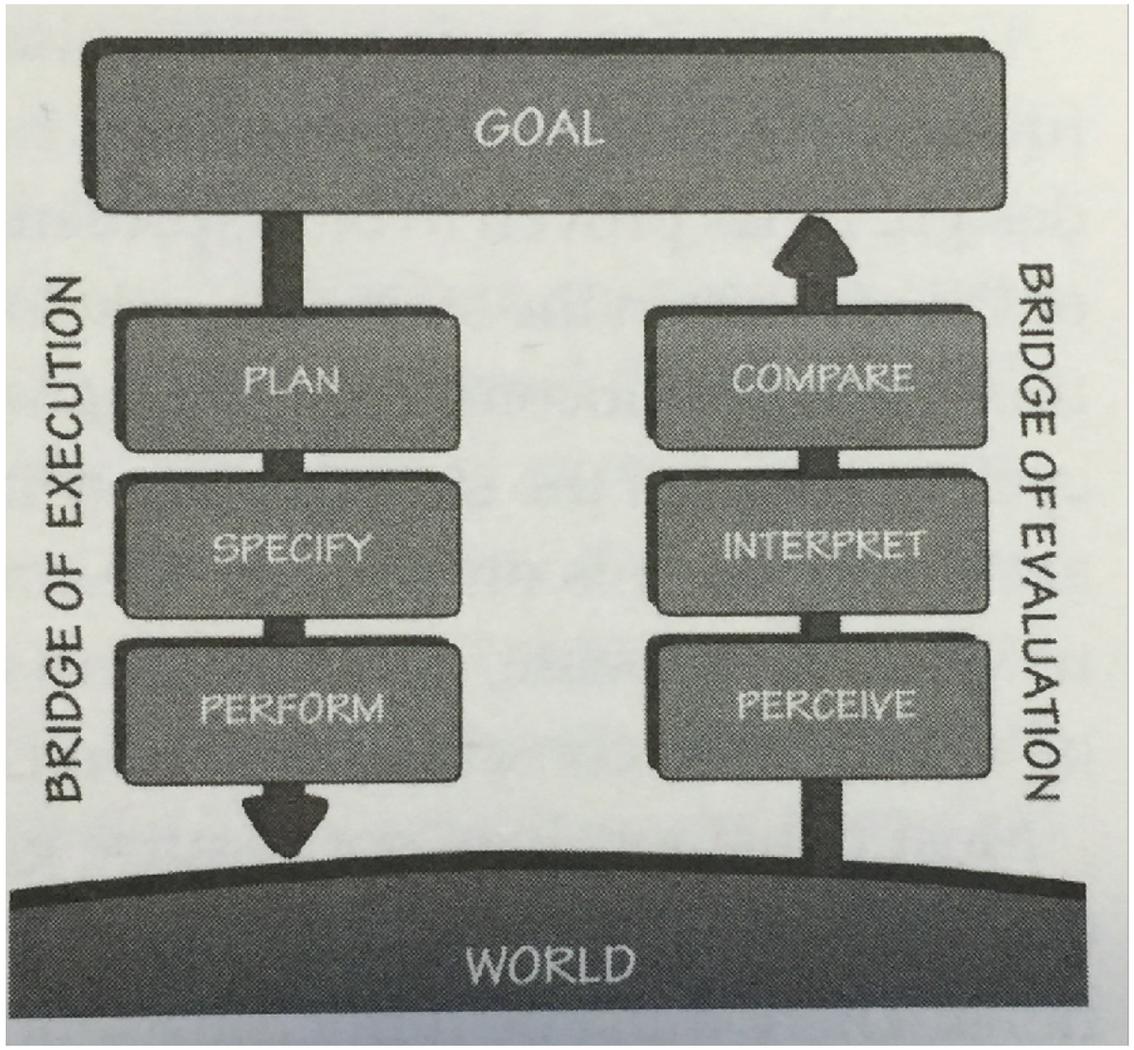
Avoid anthropomorphism (in most contexts)

- Anthropomorphism - the attribution of human characteristics to non-human objects
  - e.g., “Sorry, but I cannot find the file you need”
- Provides a false mental model
  - leads to user thinking they can interact with system as person
  - can be over promising & condescending
- May work in spoken interaction settings, where system does match user’s mental model

Direct manipulation

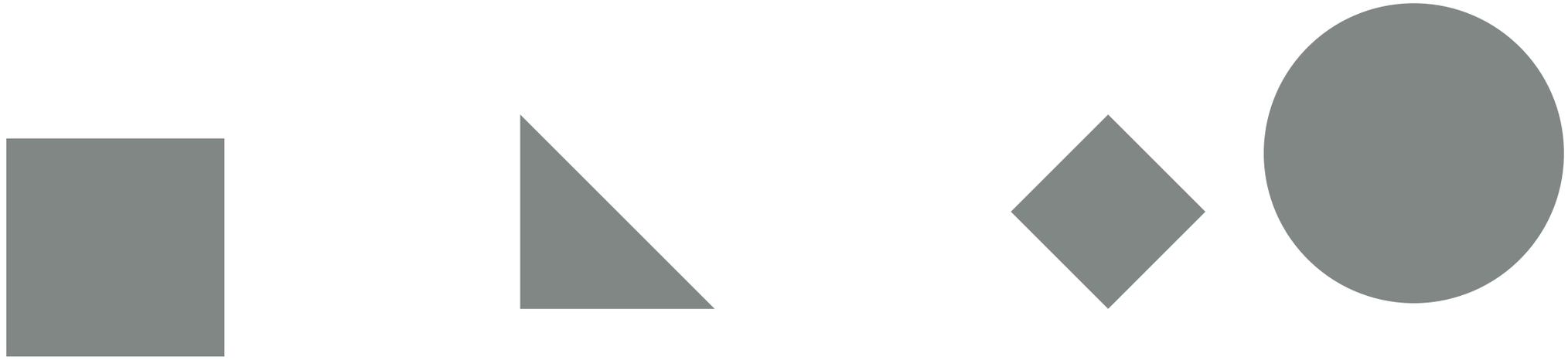
Motivation

- User is trying to do a task, manipulating some [model] of world
- Hard to plan out long sequence of actions in advance
- Gulf of execution: hard to know if took correct action
- Gulf of evaluation: hard to understand if successfully manipulated world
- Hard to compare hidden world to desired world



Direct manipulation

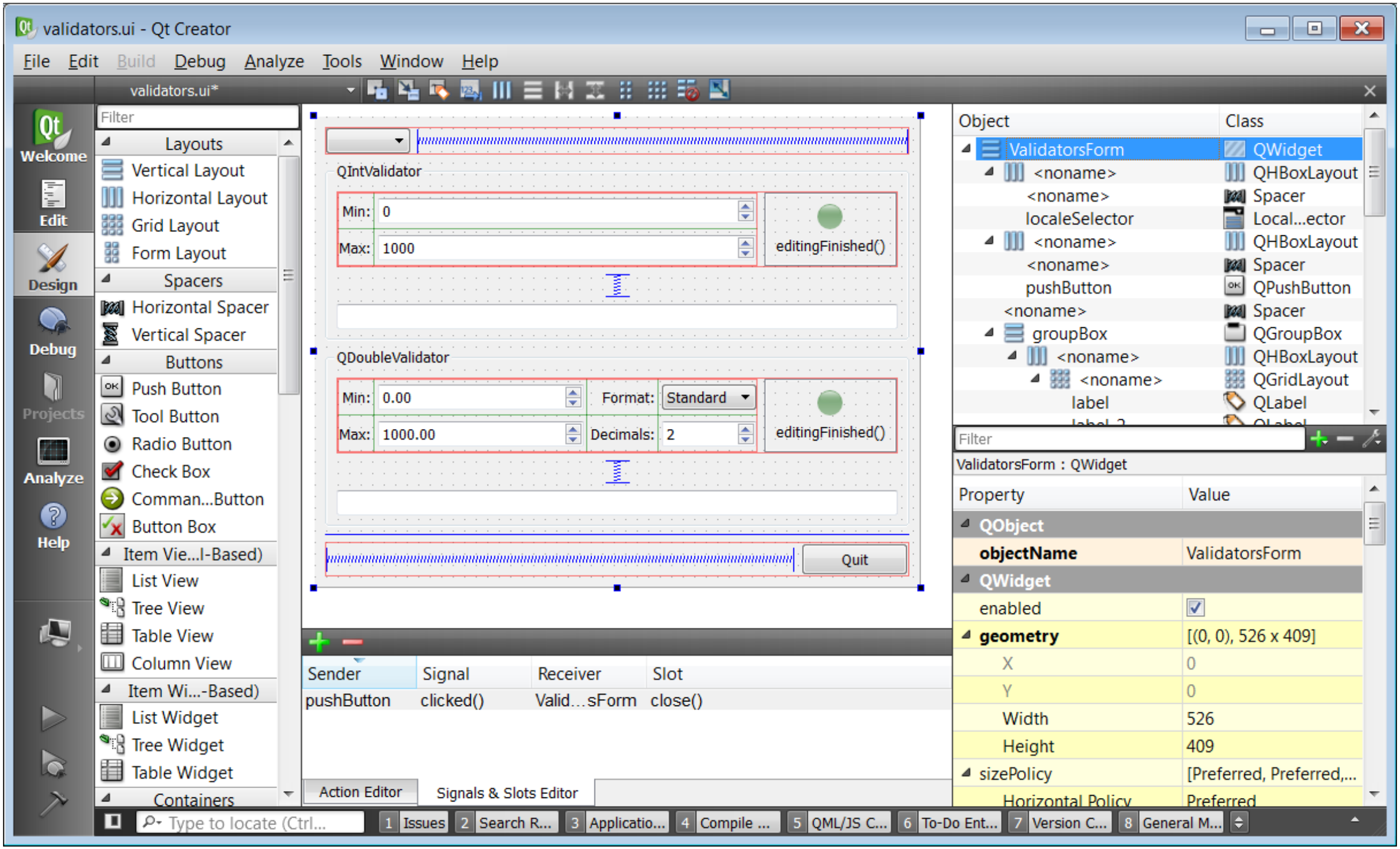
- “Rapid incremental reversible operations whose impact on the objects of interest is immediately visible” (Shneiderman, 1982)



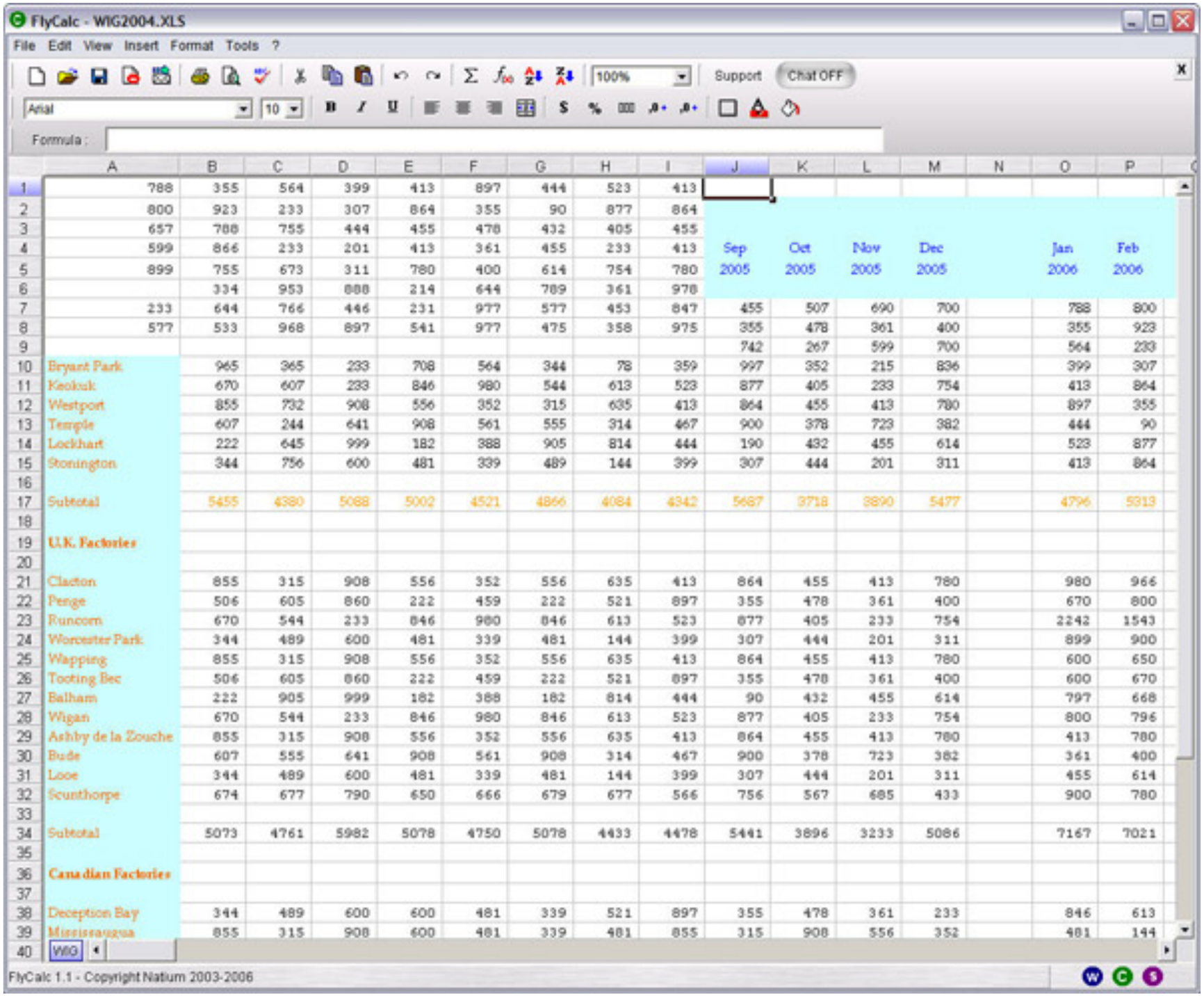
Benefits

- Supports exploration
  - Don't plan long sequence of actions: pick an action, try it, can change mind if want to do something else instead
- Provides immediate feedback
  - Can quickly see what outcome of actions are in manipulating the world
  - Easy to compare desired state of the world to actual state of the world

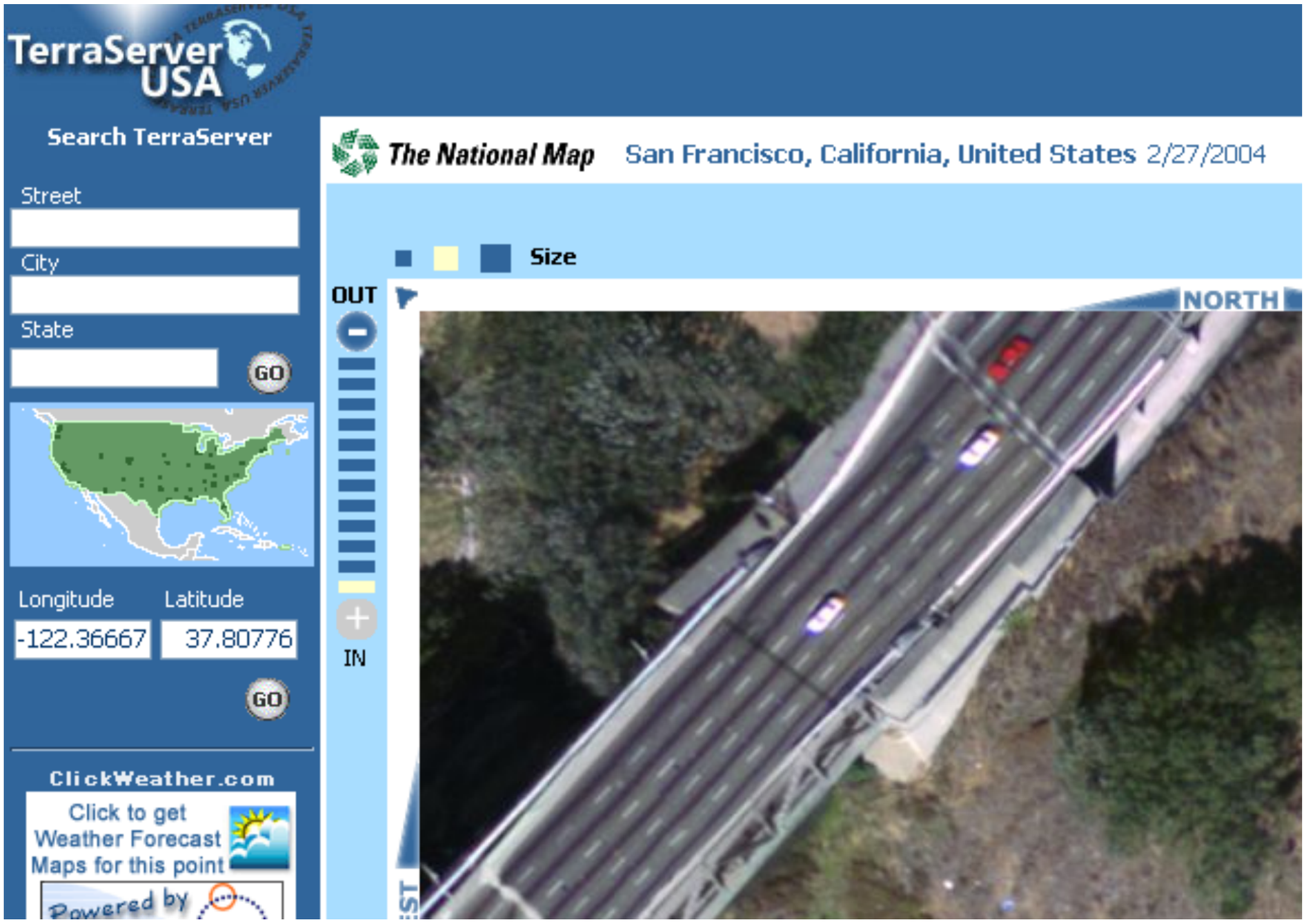
Example - GUI builder



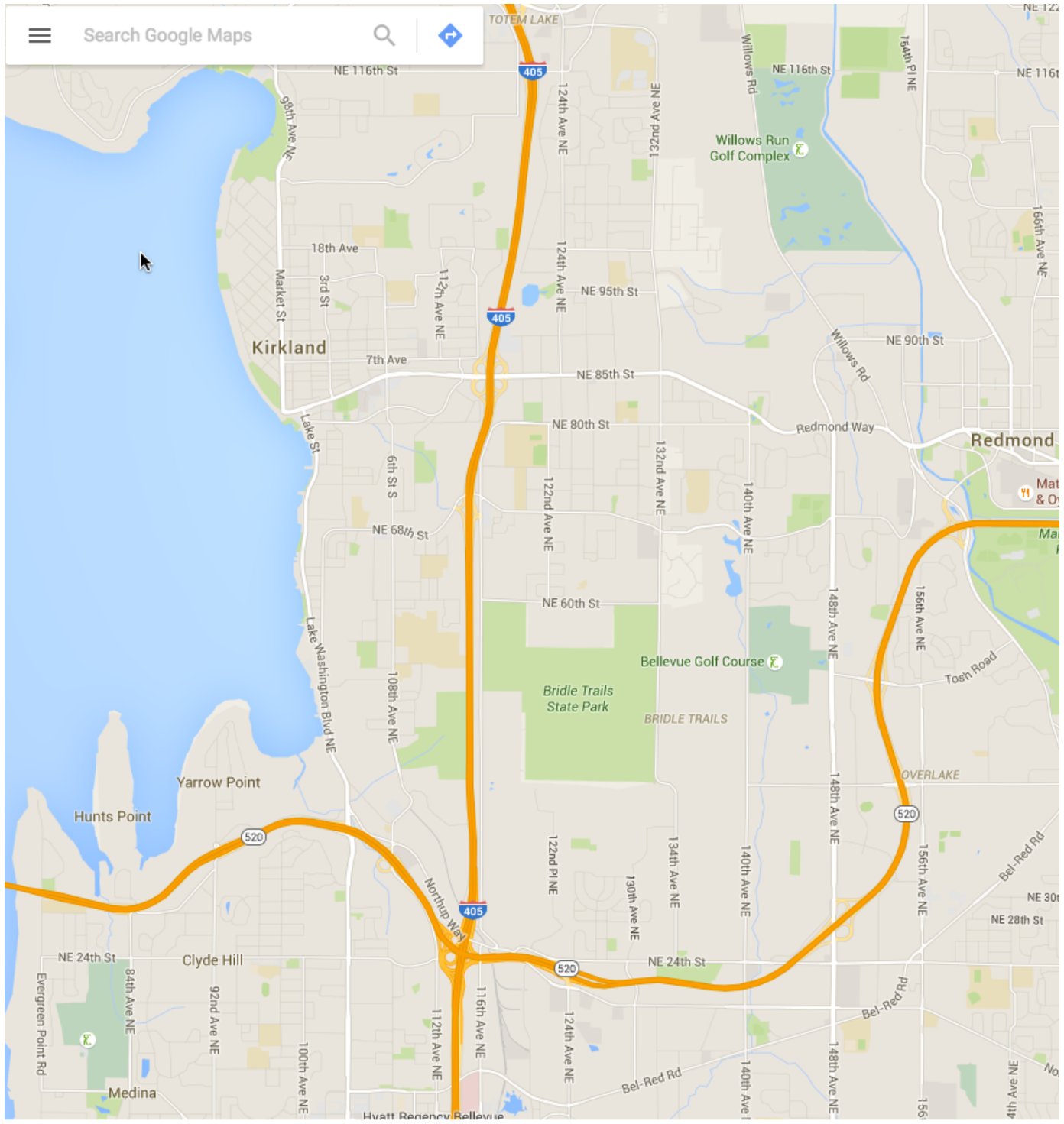
Example - Spreadsheets



Example - Microsoft TerraServer



Example - Google Maps



Example - Kayak

Advice: BUY Learn more

Create a price alert

Stops Show all

- ✓ nonstop $127
- 1 stop $145
- 2+ stops $303

Times Show all

Take-off Washington (DCA)
Fri 2:41p – 10:30p

Take-off Chicago (CHI)
Mon 5:30a – 10:00p

Show landing times

Airports Show all

- Depart/Return same

Washington

- ✓ DCA: Reagan-Nati… $127
- BWI: Baltimore/Wa… $207

DCA ↔ CHI
108 of 1115 flights

Dec 16 ↔ Dec 19
Friday — Monday

Economy cabin | 1 traveler

Change

Sort by: **Price** Recommended Duration More

Round-trip | Flight-by-flight

$207

View Deal

JustFly, Experience world-class service
Click "View Deal" to find our cheapest flights

justfly.com

$207 nonstop View Deal
www.justfly.com

Ad

$227
American Airlines

View Deal

American Airlines
8:12p DCA → **9:26p** ORD 2h 14m nonstop
3:25p ORD → **6:12p** DCA 1h 47m nonstop

Show details Economy

$227
American Airlines

View Deal

American Airlines
8:12p DCA → **9:26p** ORD 2h 14m nonstop
11:55a ORD → **2:42p** DCA 1h 47m nonstop

Show details Economy

In Class Activity

- In groups of 2 or 3:
  - Identify at least 3 separate usability issues of a web application that violates one of the interaction design principles in this lecture
  - For each issue, brainstorm ways that this usability issue might be addressed.

In Class Activity: Direct Manipulation Programming Interactions

- In groups of 2
  - Design a system for writing React code through direct manipulation
    - Create sketches showing key screens
    - Should support
      - Standard programming language features (variables, conditionals, loops, functions)
      - Should make it faster and easier to make code changes
      - Should make it easier to get feedback on if program works