

# Interaction Techniques

SWE 432, Fall 2016

Design and Implementation of Software for the Web

# 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?

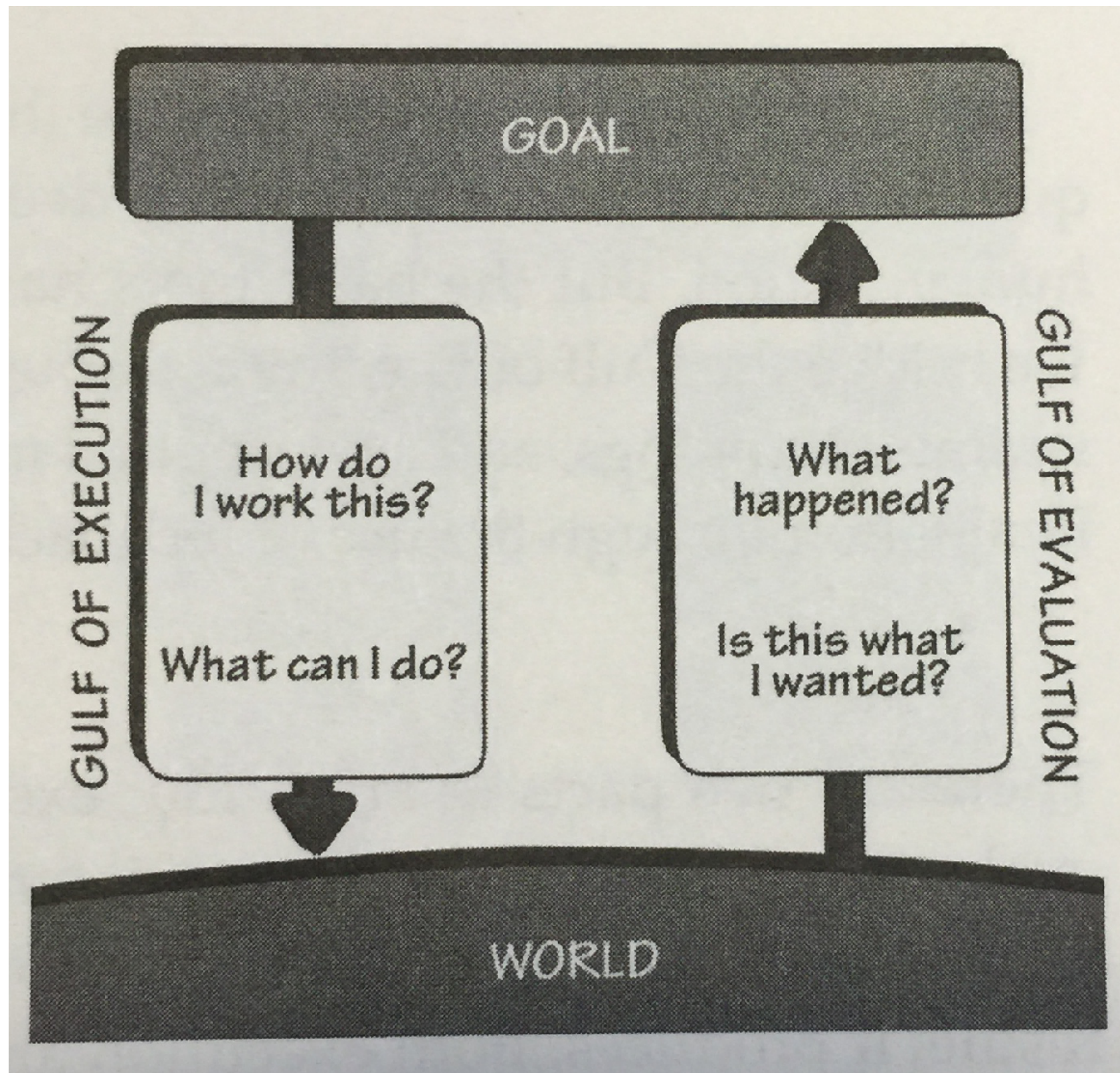
# Interaction technique

- 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>

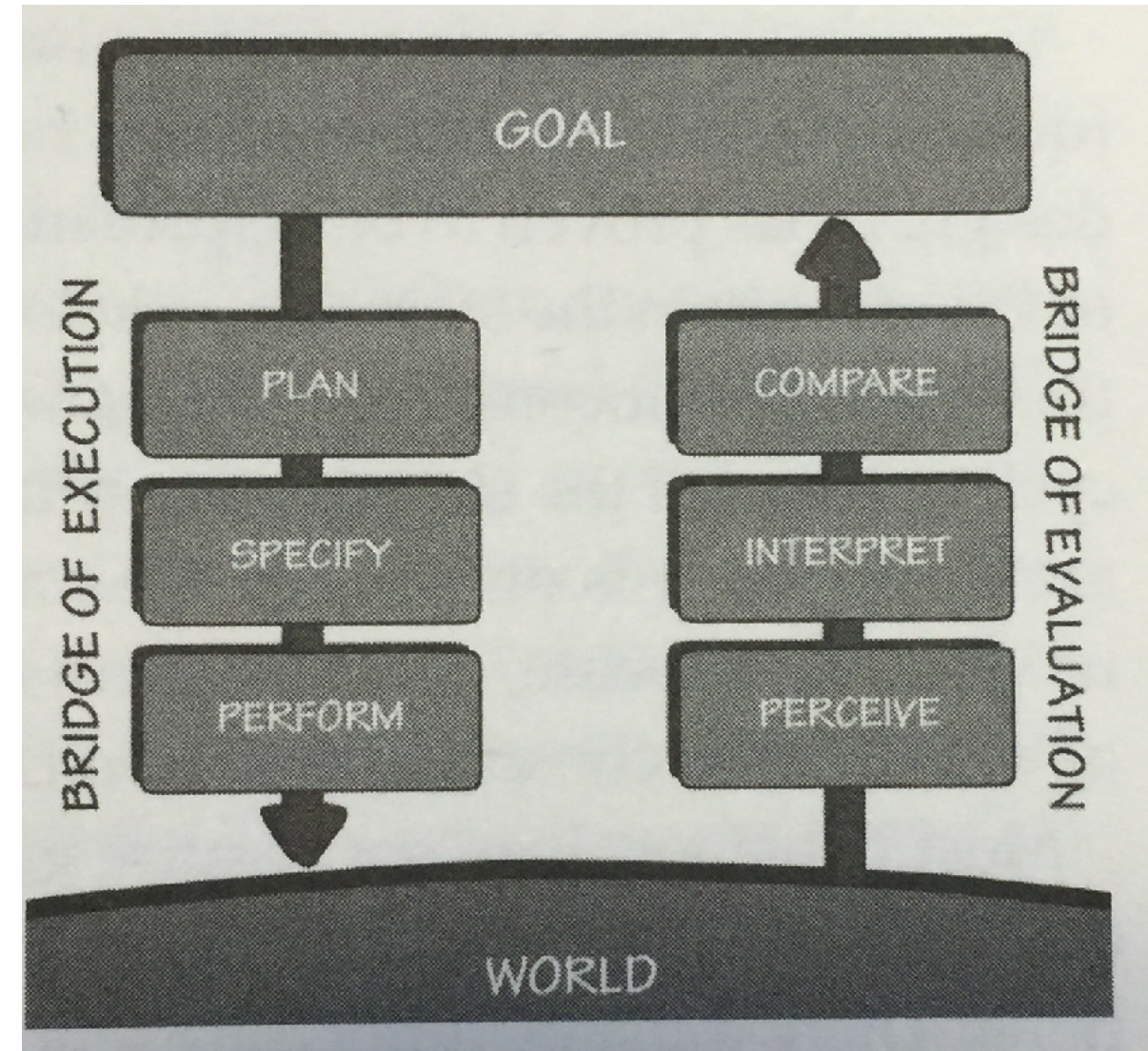
# Gulfs of execution and evaluation





# 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)



# Translation

goals  $\longrightarrow$  action sequence

# 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

The screenshot shows a test runner interface for a function named `calculate`. The test status is "failed". The description is "it should throw an exception if the parameters are invalid". The execution time is 6ms. The message is "expected 4 to equal 3". The diff shows "3 - 4". The code snippet is `expect(calculate('+',[1,2])).to.equal(3);`.

The function editor shows the following code:

```
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 = prod(numbers[0],numbers[1]);
26             return res;
27         default:
28             // ...
29     }
30 }
```

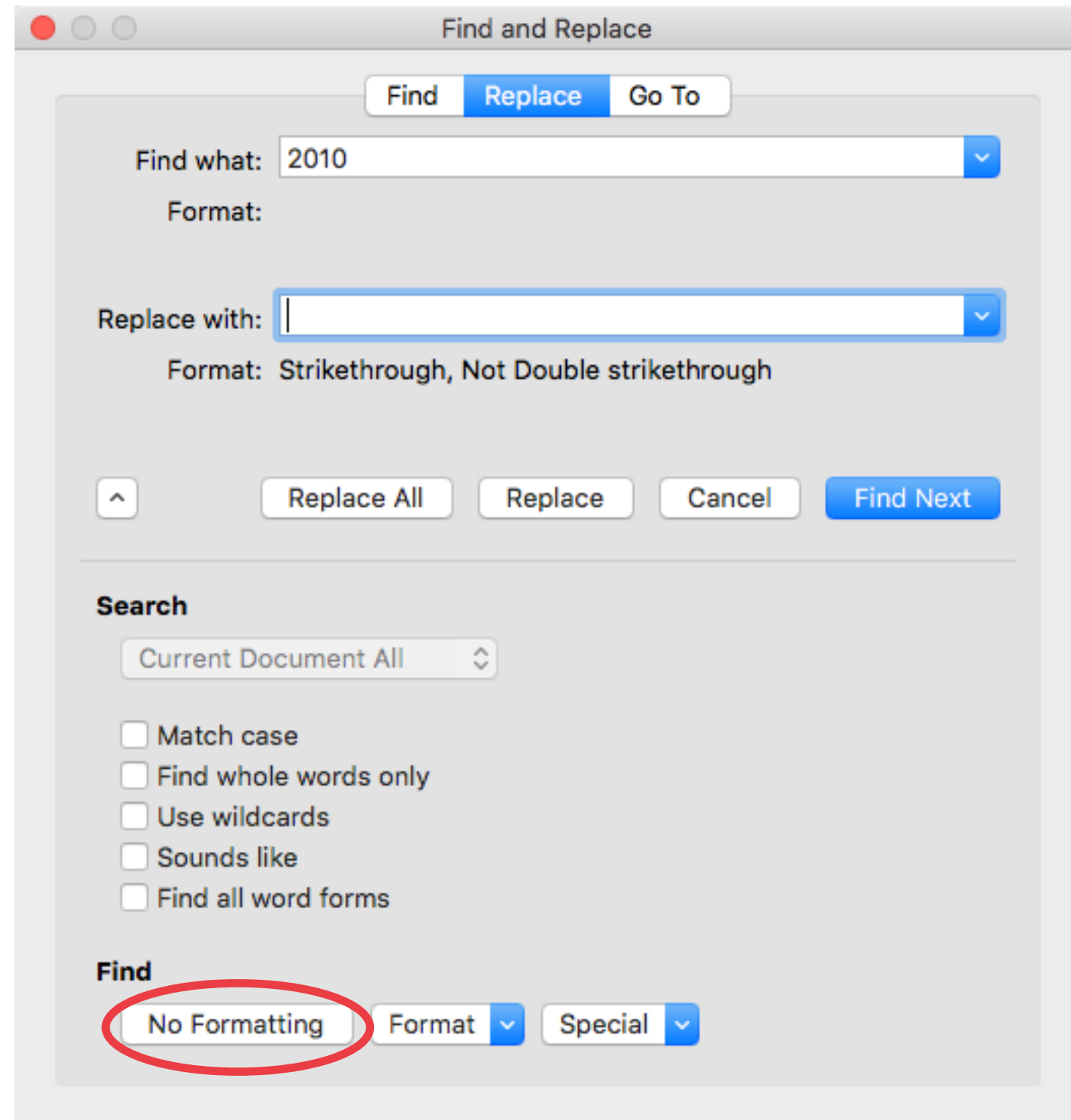
A tooltip is visible over the `sum` function call on line 22, showing the result of the function call: `sum(numbers[0],numbers[1]) X 4`. Below the code editor, a list of errors is shown: "1. Line 15: Missing semicolon."

# 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

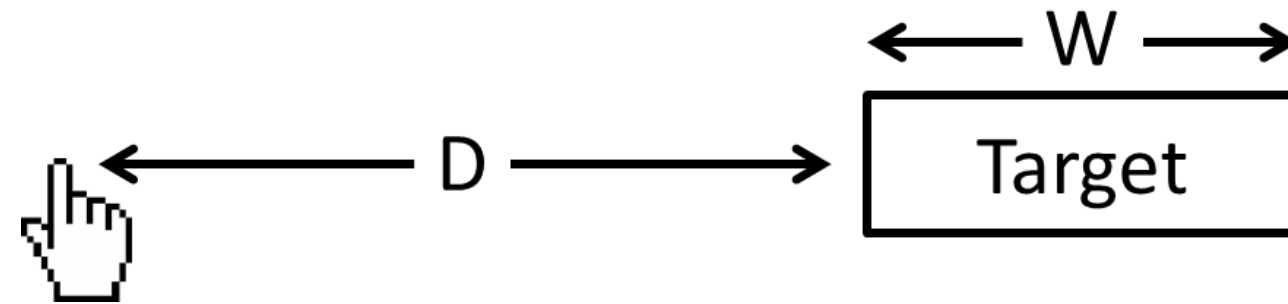
- As user is interacting with objects, provide feedback on interactions
- Examples
  - While dragging object, show new position
  - As selecting text, show selection
  - While clicking on button, show button changing

# 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

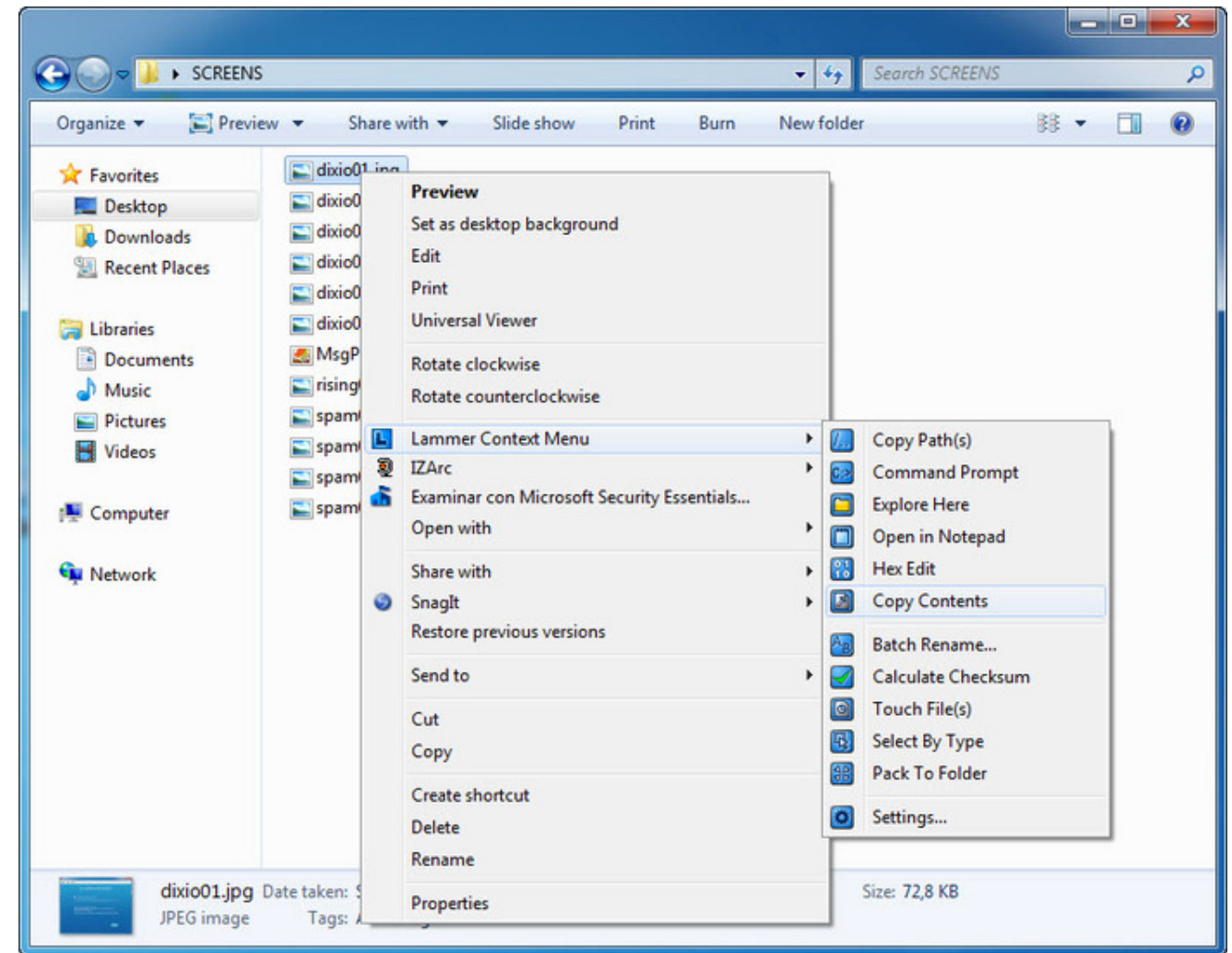
# Design implications of Fitt's law

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



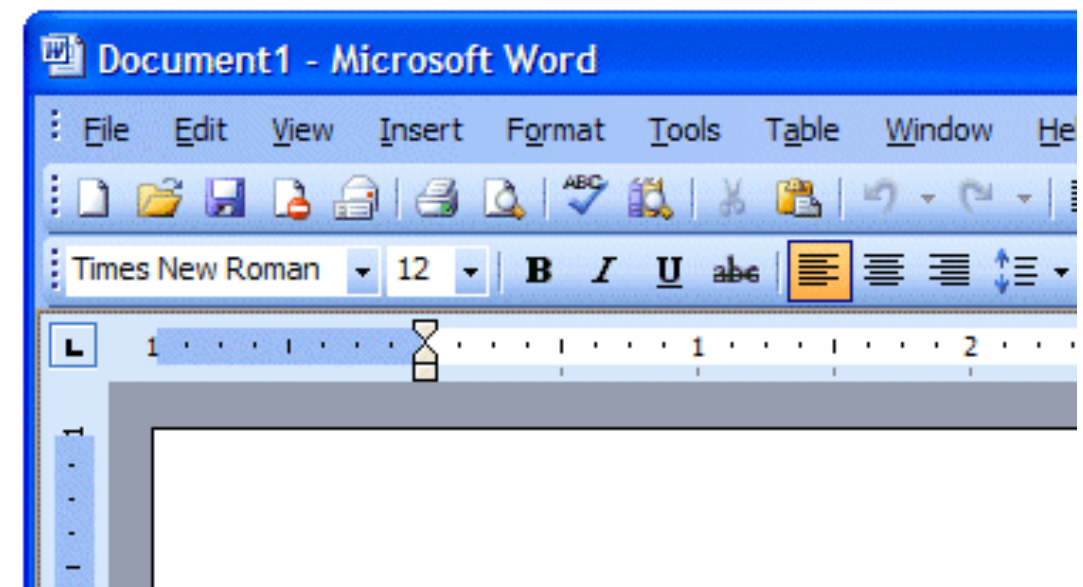
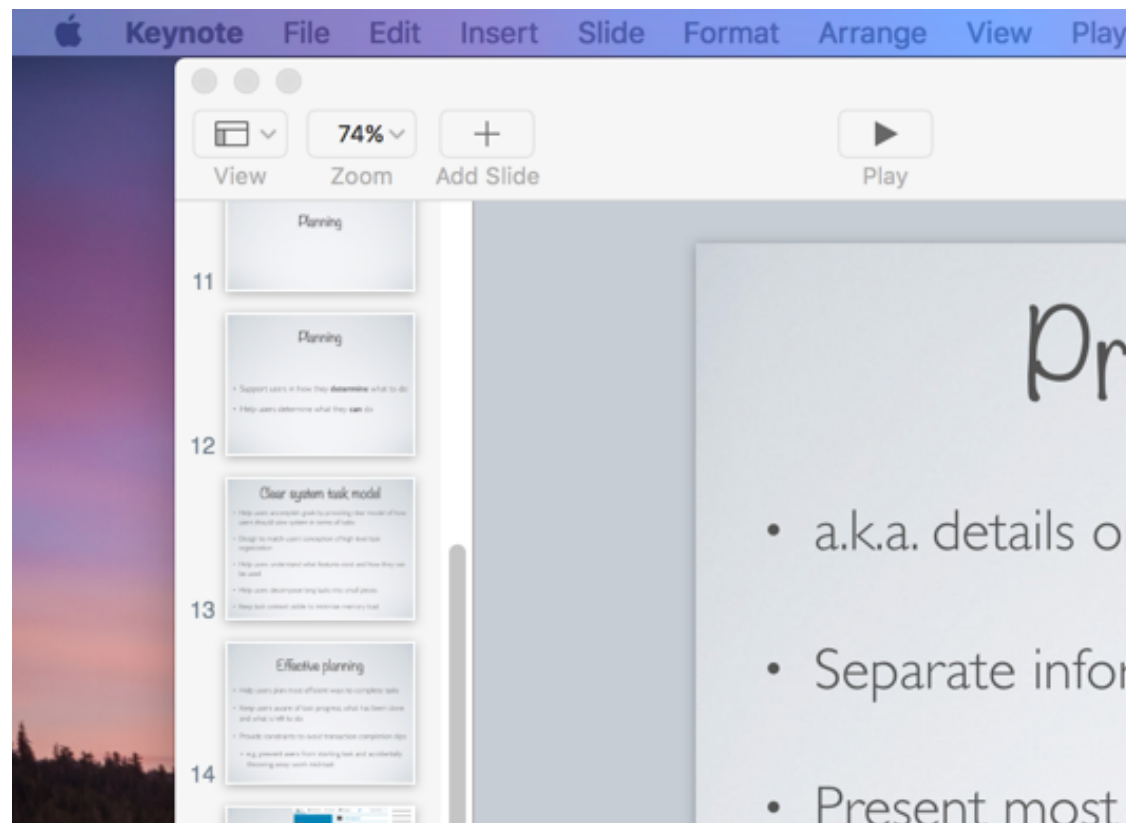
# Design implications of Fitt's law

- 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



System feedback

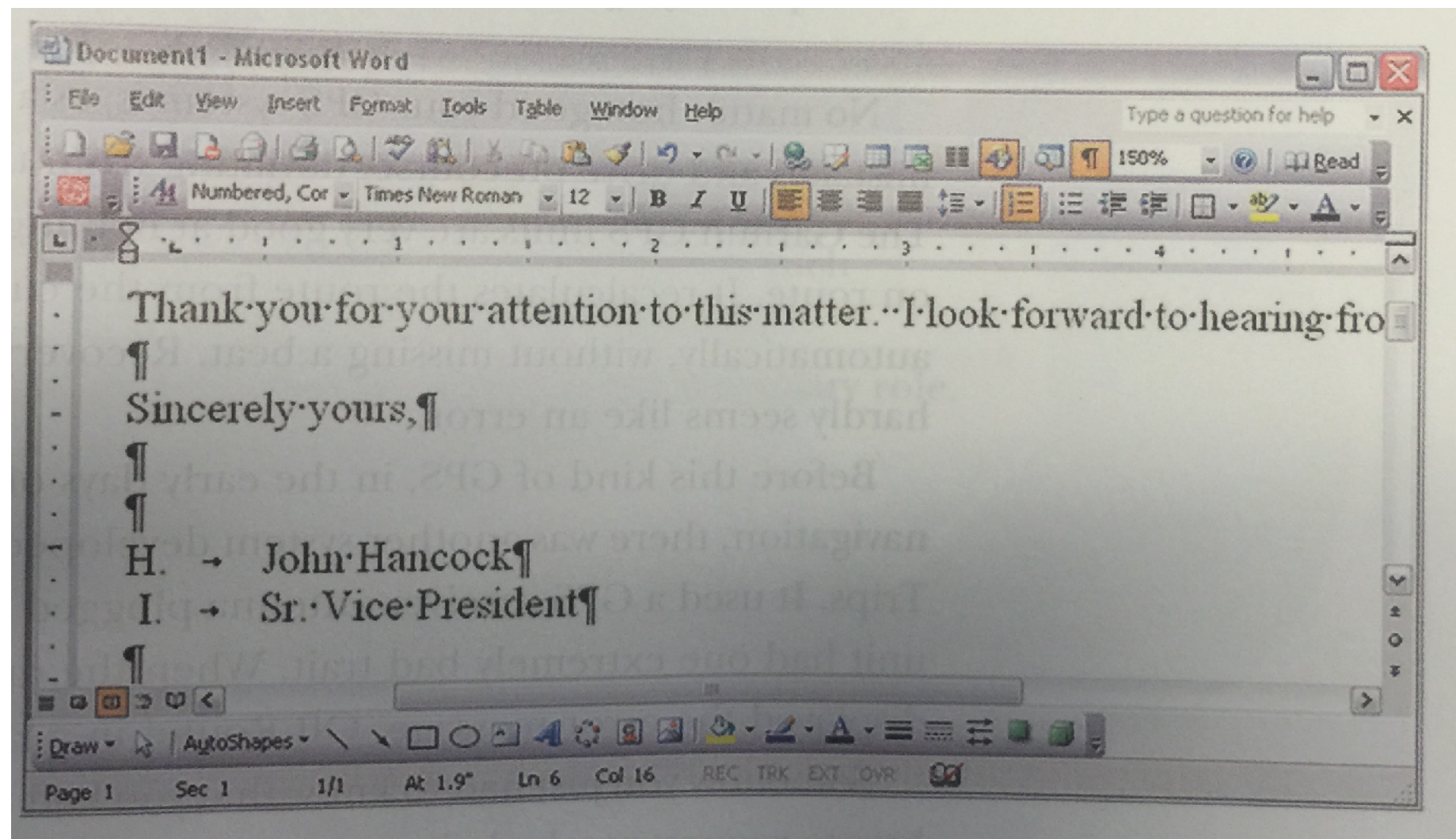


# System response times

- 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

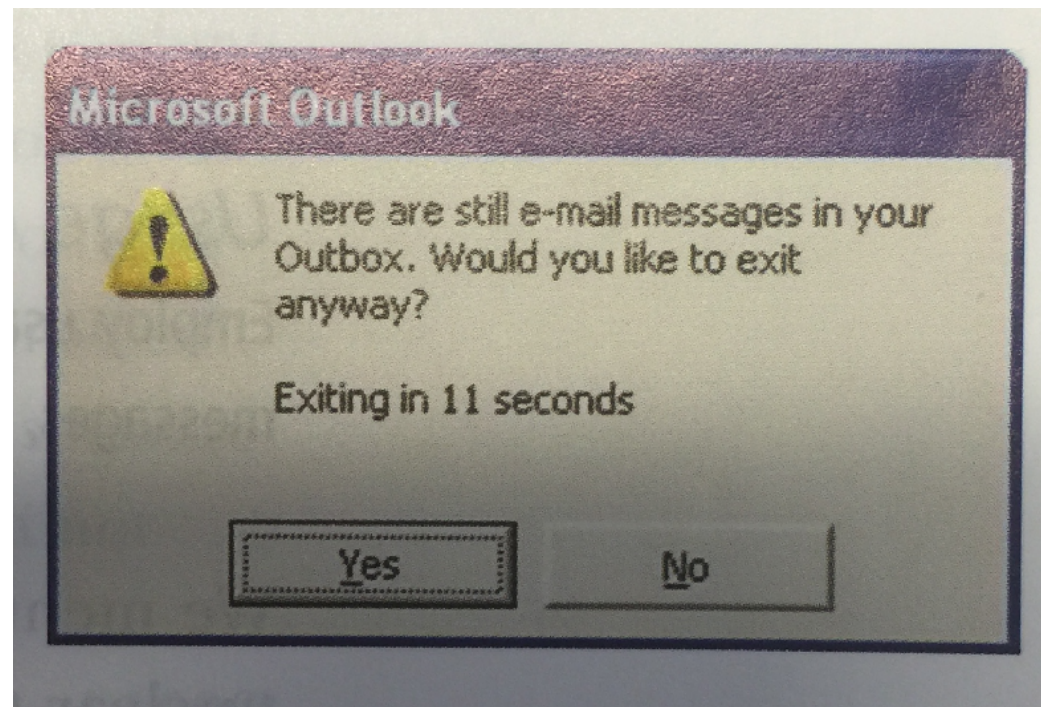
# Tone of feedback

- 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

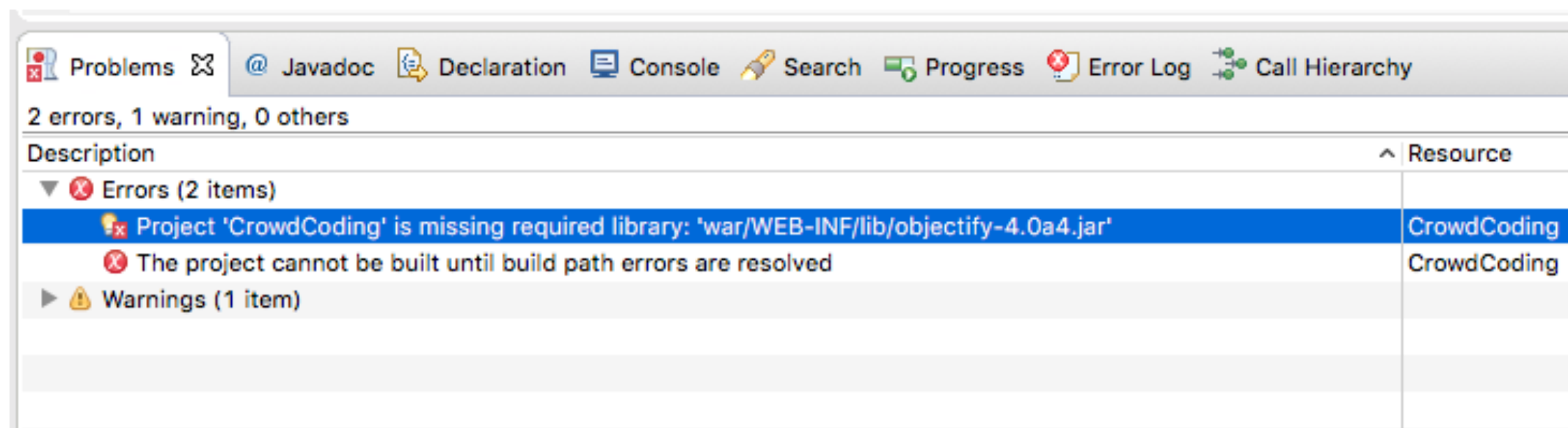
- Clarity - support clear understanding of outcome
- Precise wording
- Completeness - include enough information to fully understand outcomes





# 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, I 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

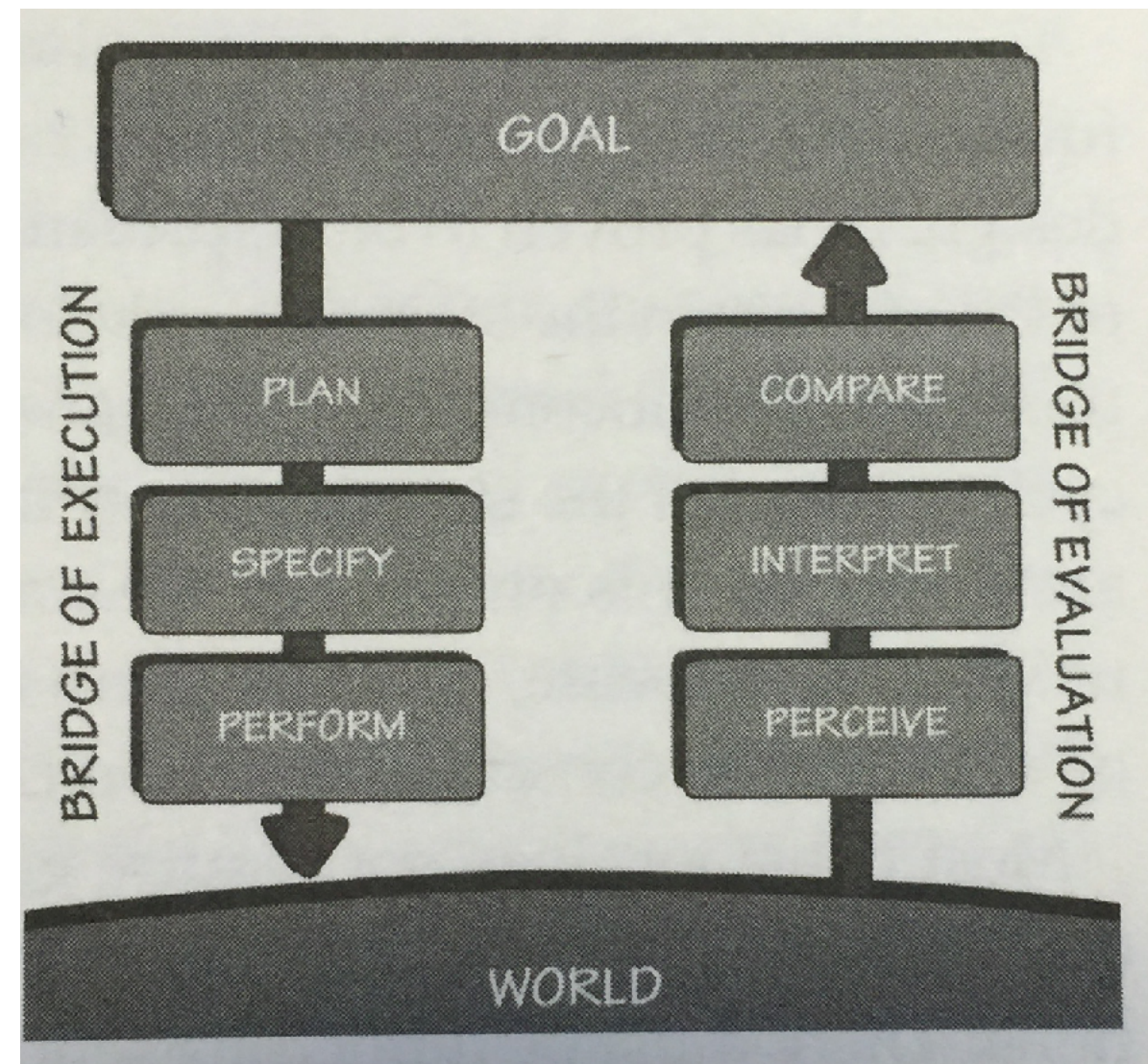
# In Class Activity

- 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.

Direct manipulation

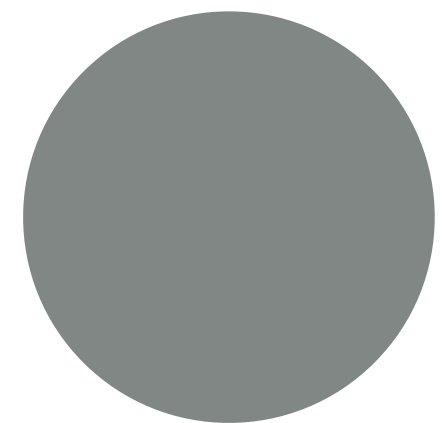
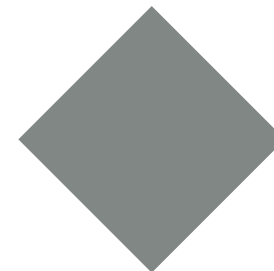
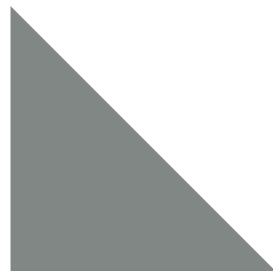
# Motivation

- 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)

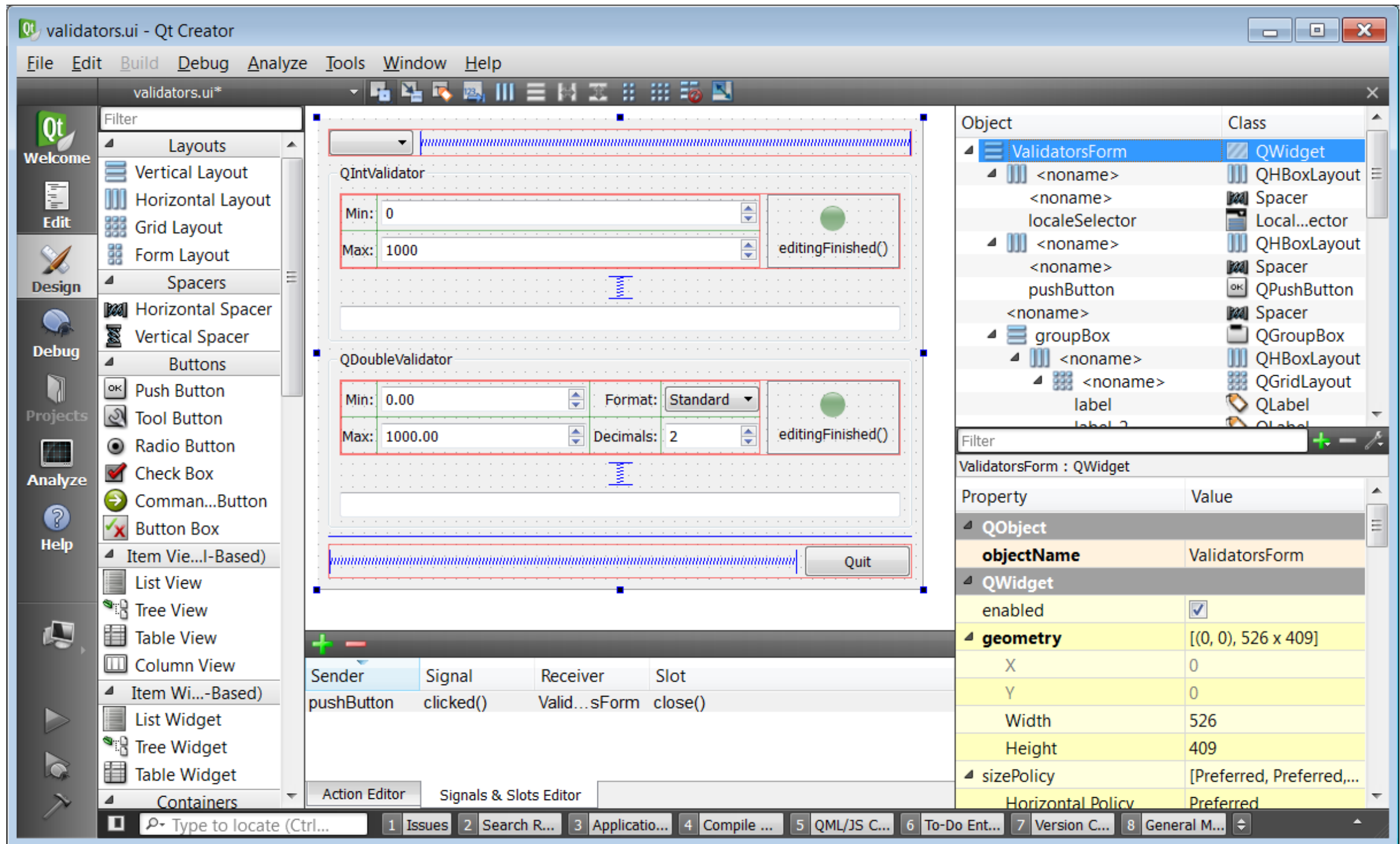




# Benefits

- 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

FlyCalc - WIG2004.XLS

File Edit View Insert Format Tools ?

100% Support Chat OFF

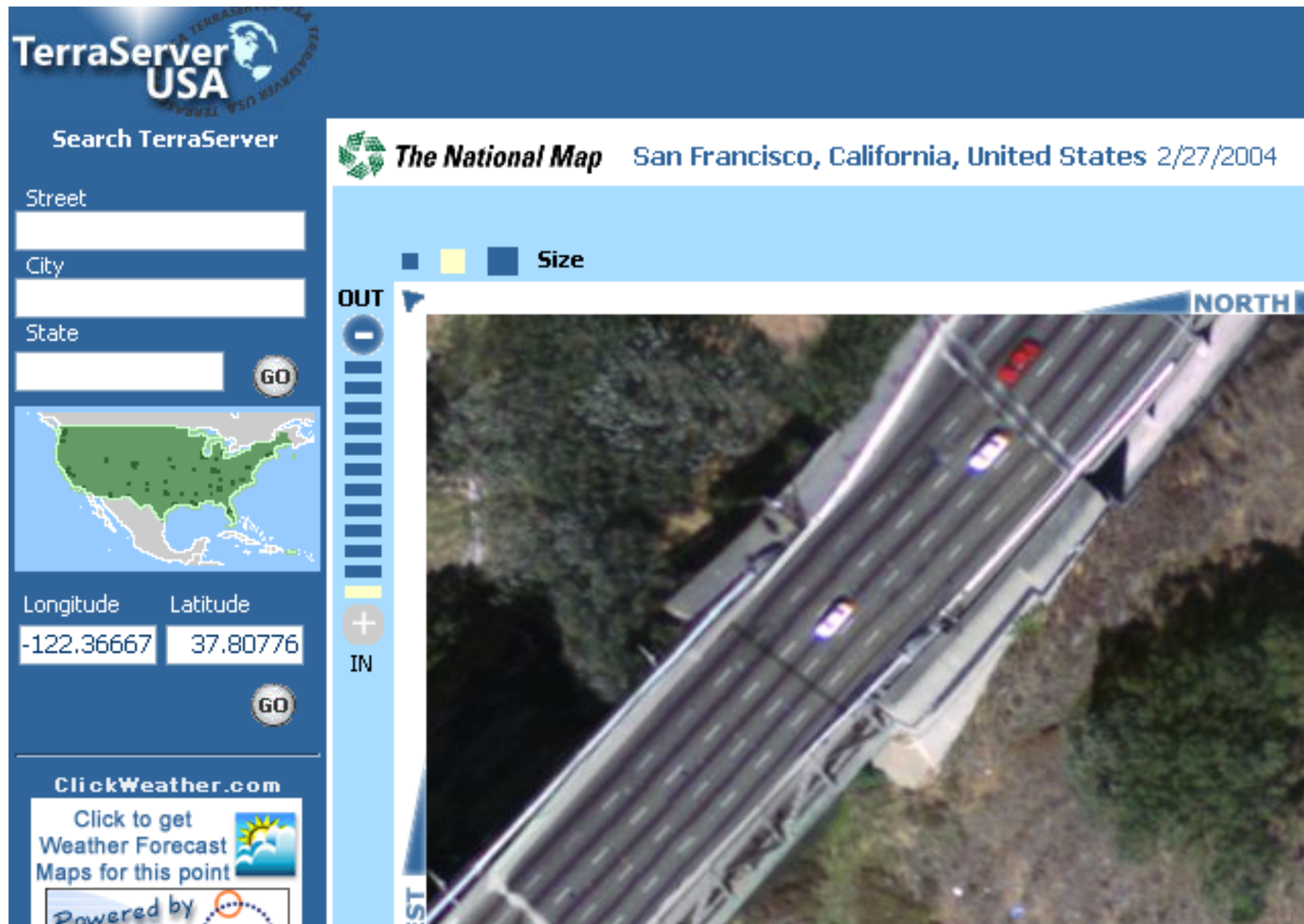
Formula :

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1		788	355	564	399	413	897	444	523	413						
2		800	923	233	307	864	355	90	877	864						
3		657	788	755	444	455	478	432	405	455						
4		599	866	233	201	413	361	455	233	413	Sep 2005	Oct 2005	Nov 2005	Dec 2005	Jan 2006	Feb 2006
5		899	755	673	311	780	400	614	754	780						
6			334	953	888	214	644	789	361	978						
7		233	644	766	446	231	977	577	453	847	455	507	690	700		788
8		577	533	968	897	541	977	475	358	975	355	478	361	400		355
9											742	267	599	700		564
10	Bryant Park	965	365	233	708	564	344	78	359	997	352	215	836			399
11	Keokuk	670	607	233	846	980	544	613	523	877	405	233	754			413
12	Westport	855	732	908	556	352	315	635	413	864	455	413	780			897
13	Temple	607	244	641	908	561	555	314	467	900	378	723	382			444
14	Lockhart	222	645	999	182	388	905	814	444	190	432	455	614			523
15	Stonington	344	756	600	481	339	489	144	399	307	444	201	311			413
16																
17	Subtotal		5455	4380	5088	5002	4521	4866	4084	4342	5687	3718	3890	5477		4796
18																
19	U.K. Factories															
20																
21	Clacton	855	315	908	556	352	556	635	413	864	455	413	780			980
22	Perge	506	605	860	222	459	222	521	897	355	478	361	400			670
23	Runcorn	670	544	233	846	980	846	613	523	877	405	233	754			2242
24	Worcester Park	344	489	600	481	339	481	144	399	307	444	201	311			899
25	Wapping	855	315	908	556	352	556	635	413	864	455	413	780			600
26	Tooting Bec	506	605	860	222	459	222	521	897	355	478	361	400			600
27	Belham	222	905	999	182	388	182	814	444	90	432	455	614			797
28	Wigan	670	544	233	846	980	846	613	523	877	405	233	754			800
29	Ashby de la Zouche	855	315	908	556	352	556	635	413	864	455	413	780			413
30	Bude	607	555	641	908	561	908	314	467	900	378	723	382			361
31	Looe	344	489	600	481	339	481	144	399	307	444	201	311			455
32	Scunthorpe	674	677	790	650	666	679	677	566	756	567	685	433			900
33																
34	Subtotal		5073	4761	5982	5078	4750	5078	4433	4478	5441	3896	3233	5086		7167
35																
36	Canadian Factories															
37																
38	Deception Bay	344	489	600	600	481	339	521	897	355	478	361	233			846
39	Mississauga	855	315	908	600	481	339	481	855	315	908	556	352			481
40	WIG															

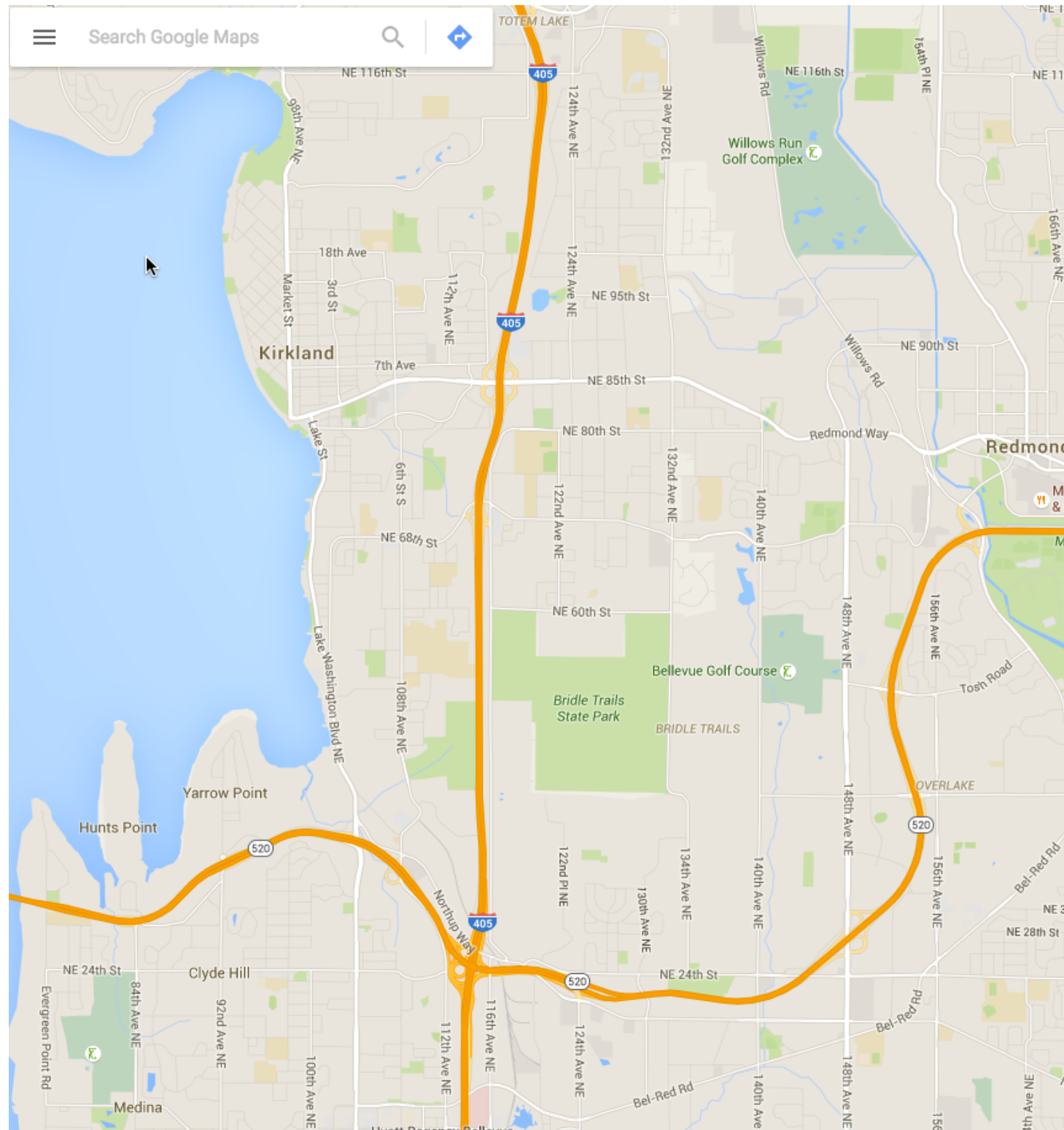
FlyCalc 1.1 - Copyright Natium 2003-2006



# 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  
Friday ↔ Dec 19  
Monday

Economy 1  
cabin 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  
[www.justfly.com](#)

[View Deal](#)

Ad

**\$227**

American Airlines



American Airlines



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

[View Deal](#)

[Show details](#)

Economy

**\$227**

American Airlines



American Airlines



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

[View Deal](#)

[Show details](#)

Economy

# 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