

Preventing Error

SWE 632, Spring 2018

Today

- What causes errors?
- What design choices can help reduce the frequency of errors?
- What design choices can help users resolve errors more effectively?

Human Error

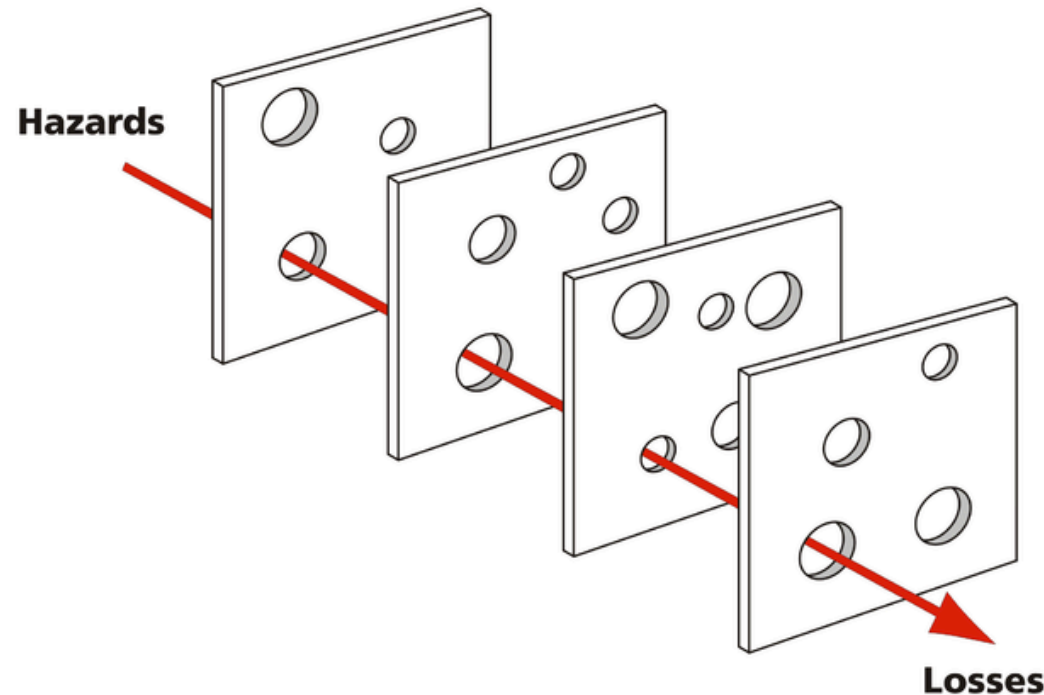
Root cause analysis

- Keep asking **why** to determine causes for erroneous actions, and the causes of these causes
- Example
 - 2010 F-22 crash that killed pilot
 - Official cause: pilot error - pilot failed to take corrective action
 - Why did the pilot not take the action?
 - Pilot was not receiving oxygen and was probably unconscious.

What causes disasters?

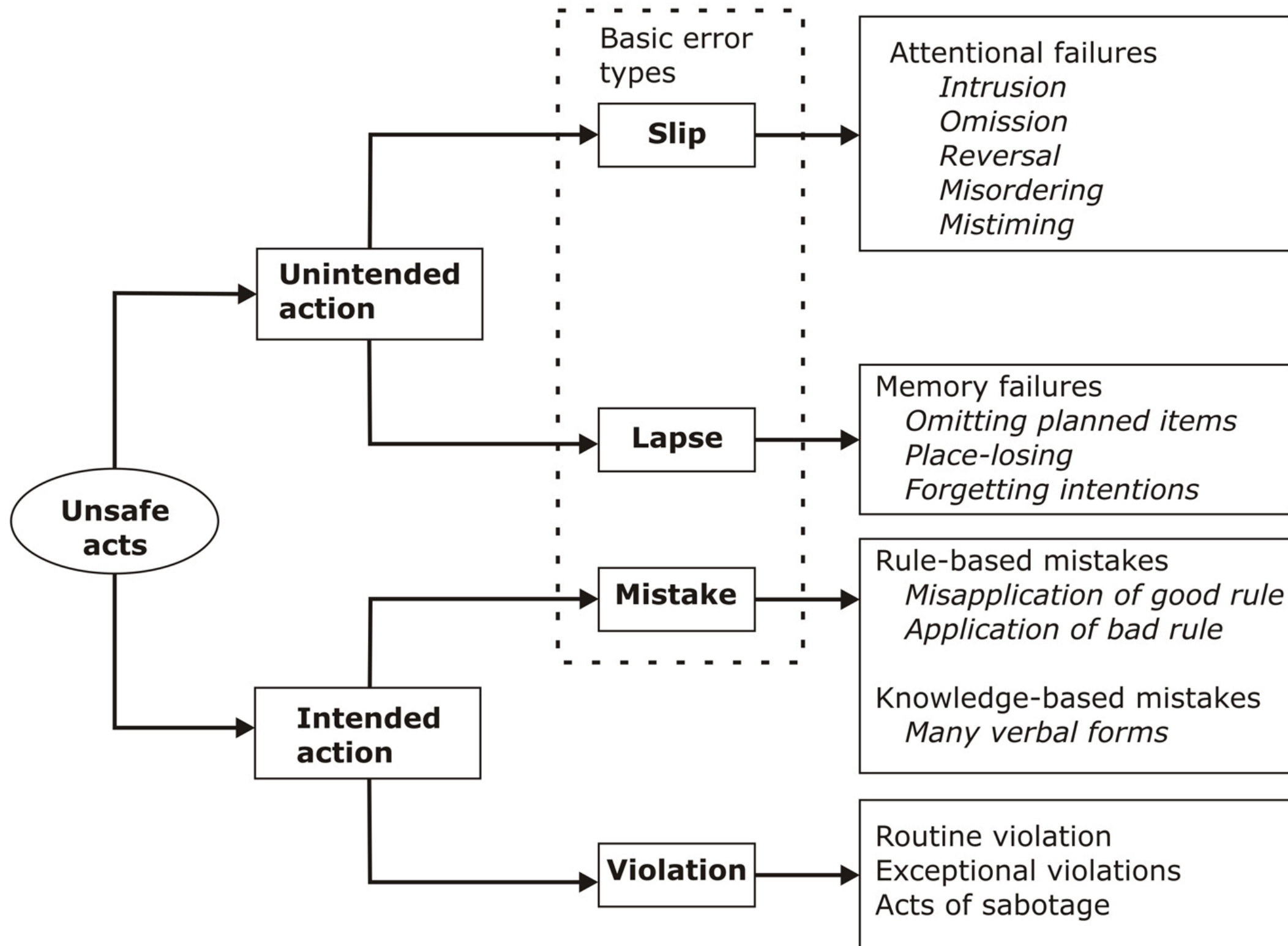
- Mechanical malfunction?
- Poor design?
- Human error?

Swiss cheese model



- Accidents must penetrate levels of system defenses
- Reduce accidents by
 - Adding more layers
 - Reduce the size and number of holes
 - Alert users when holes line up

Psychological types of unsafe acts



Violation

- Error occurred because user **intended** the erroneous output
- Routine violation - user always intends to do it
 - Noncompliance is so frequent it is ignored
 - E.g., running a red light
- Exceptional - only in some cases
- Sabotage - intended destruction

Mistakes

- User **formulated** the wrong goal or plan
 - Executing action will not achieve goal
- Rule based: appropriately diagnosed situation, but chose erroneous course of action
 - Example: Night club attendees blocked from leaving during fire because bouncers thought they
- Knowledge based: does not have correct information
 - Example: Skidding driver feels brake vibrations, believes indicates malfunctioning breaks and takes foot off break, stopping ABS

Memory Lapse

- Failing to do all steps of a procedure, repeating steps, forgetting the outcome of an action, forgetting the goal or plan
- Often caused by interruption
 - Time between when plan was formulated and plan was executed leads to forgetting plan
 - Take a pen out to sign form, get interrupted talking to someone, leave it on desk rather than put it back in bag

Slips

- Attentional failure - user **intended** to do correct action, but did not actually execute action
- Example: forgot to turn off the gas burner on the stove after cooking



Strong habit intrusion

- Performance of some well-practiced activity in familiar surroundings
- Intention to depart from custom
- Failure to make an appropriate check
- Example: start trip to frequent destination, forget going somewhere else

Omissions

- May be interrupted, forgetting intention to act
- “I picked up my coat to go out when the phone rang. I answered it and then went out of the front door without my coat.”

Perceptual confusions

- Take frequent action very often, leading to high System 1 automation
- Don't perform perceptual check to verify that System 1 action is the correct one to take
- Example: "I began to pour coffee into the sugar bowl"

Mistimed checks

- Highly automated System 1 activity that is interrupted
- Error in resuming activity because usually unconscious.
- Example - interrupted in the middle of tying shoes

Activity

- Think of the last unsafe act you performed.
- What was the underlying cause?

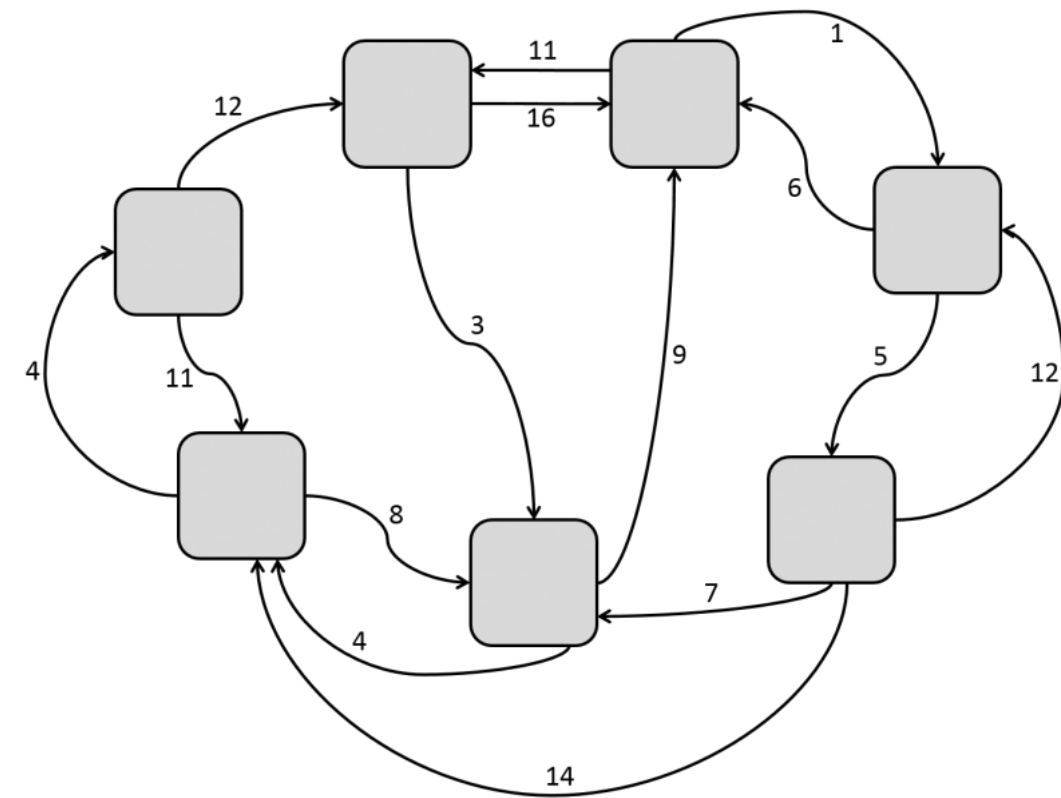
Designing for error

Designing for error

- Humans are not automatons and will never behave like automatons
- Easy to design for the situation in which everything goes well
- But important to think about what might go wrong and how the interaction design can ameliorate issues

IFT perspective

- IFT perspective
 - User exploring patches topology in search of prey
 - Always making a decision about whether a patch is the right place to hunt and changing as new information arrives
- Breaks down when user actions transform the state of the application
 - Patches and topology no longer fixed
 - Visiting a configuration of the system by clicking "Send" on the email editor is a not an undoable action



Some strategies for designing for errors

- Understand the cause, and fix it
- Make it possible to reverse errors
- Offer feedback that enables users to discover and correct errors
- Don't treat actions as errors, but as manipulations

Understand the causes of errors

- What errors occur? What type are they? How can they be prevented?
- Frequent contributing factors
 - Ambiguous or unclear information about the state of the system
 - Lack of an effective conceptual model
 - Inappropriate procedures
- Must design for users as they exist, rather than users as you'd like them to behave

Interruptions

- Interruptions are a frequent cause of error
- User may be using your interface perfectly, with the correct plan to get to their goal
 - What happens if, in the middle of the task, they answer a phone call?
 - Or if they run out of time, and come back the next day?

Designing for interruptions

- Help user resume task, by remembering where they were in task, what steps have been completed, and what steps remain
- Reduce the number of steps
- Use forcing functions to force users to do forgettable action (e.g., take card from **before** picking up cash)

Example: interruptions

- In groups of two or three
- Imagine a user was interrupted while using one of your project apps
- What errors might this create?
- What challenges might users experience when resuming?
- How could you change your design to address these issues?

Offer feedback for user actions

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

Tone of feedback

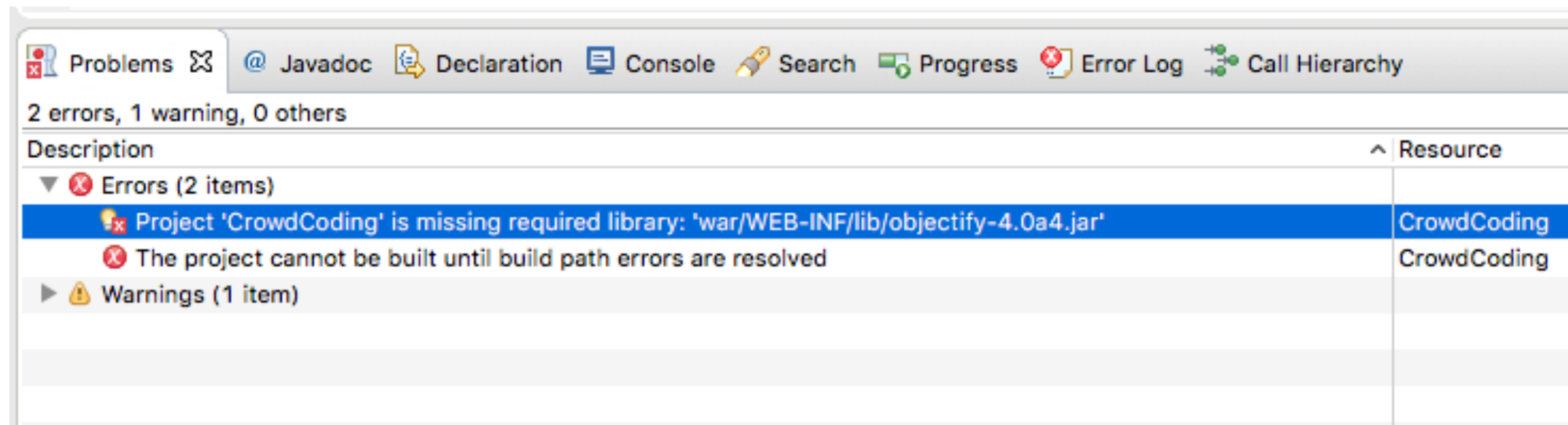
- Establishes relationship with user
- Important not to take user feel “stupid”
- Make the system take blame for errors
- Be positive, to encourage
- Provide helpful messages, not cute messages
- Avoid violent, negative, demeaning, threatening terms (e.g., illegal, invalid)

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]

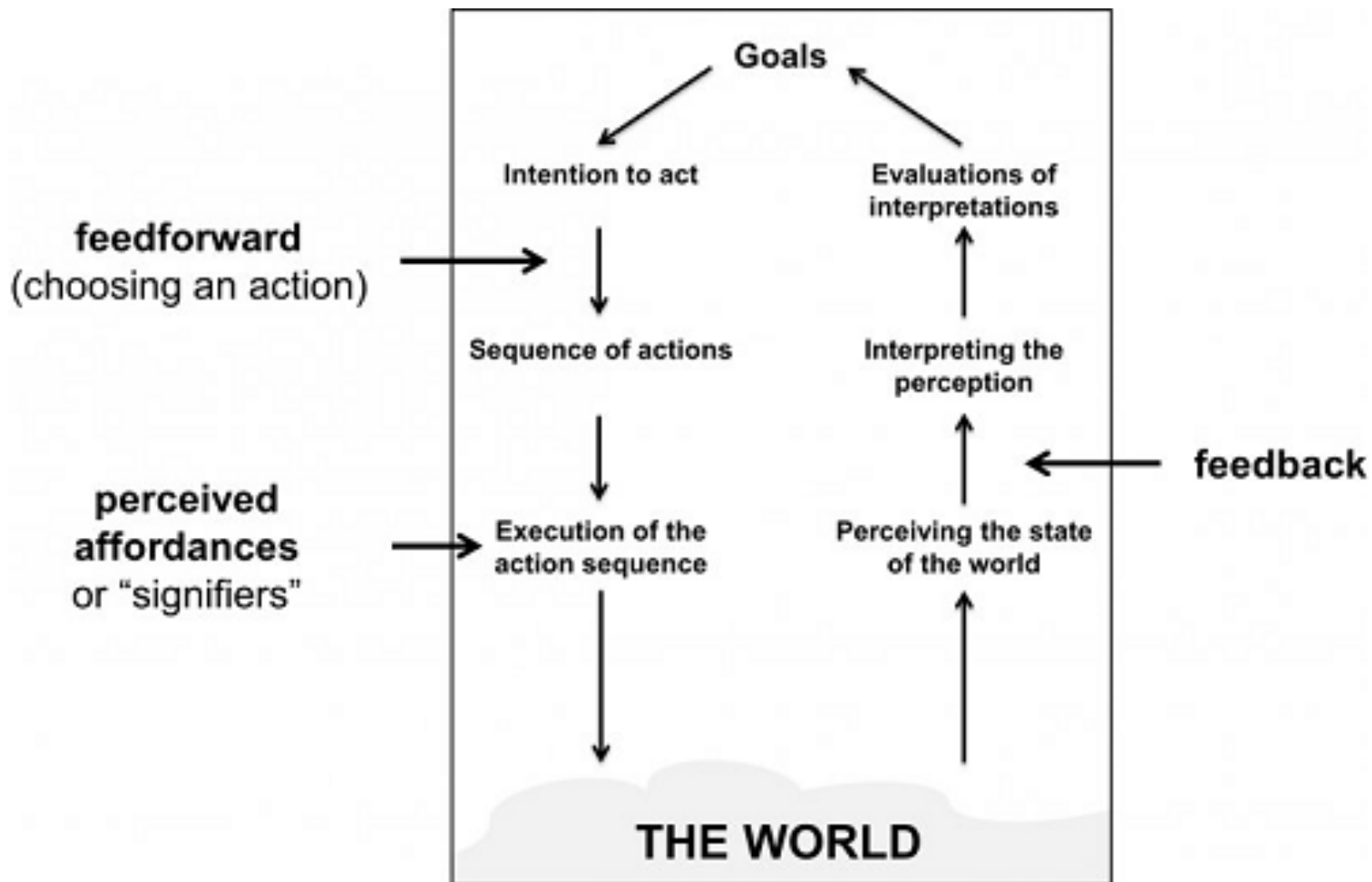
Show users how to fix errors

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



Direct manipulation

Motivation



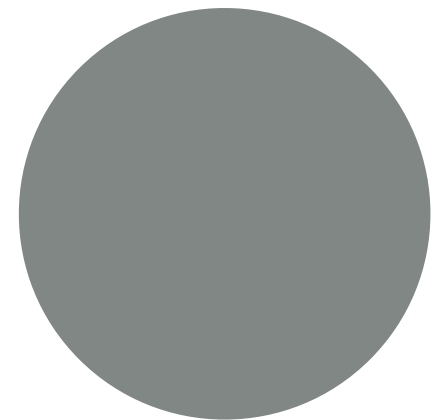
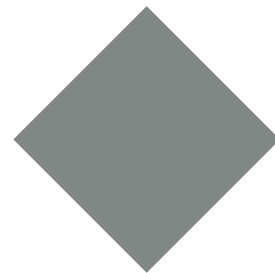
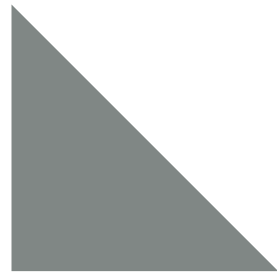
- User is trying to do a task, manipulating a [model] of world
- Hard to plan out long sequence of actions in advance
- Gulf of execution: hard to know if took correct action
- Gulf of evaluation: hard to understand if successfully manipulated world
- Hard to compare hidden world to desired world

Key questions

- What is the cost of an error?
 - Is it low cost or high cost?
 - Is it undoable?
- What feedback is necessary for user to realize the system is not in the desired state?

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 - Kayak



Advice: **BUY** [Learn more](#) [i](#)

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



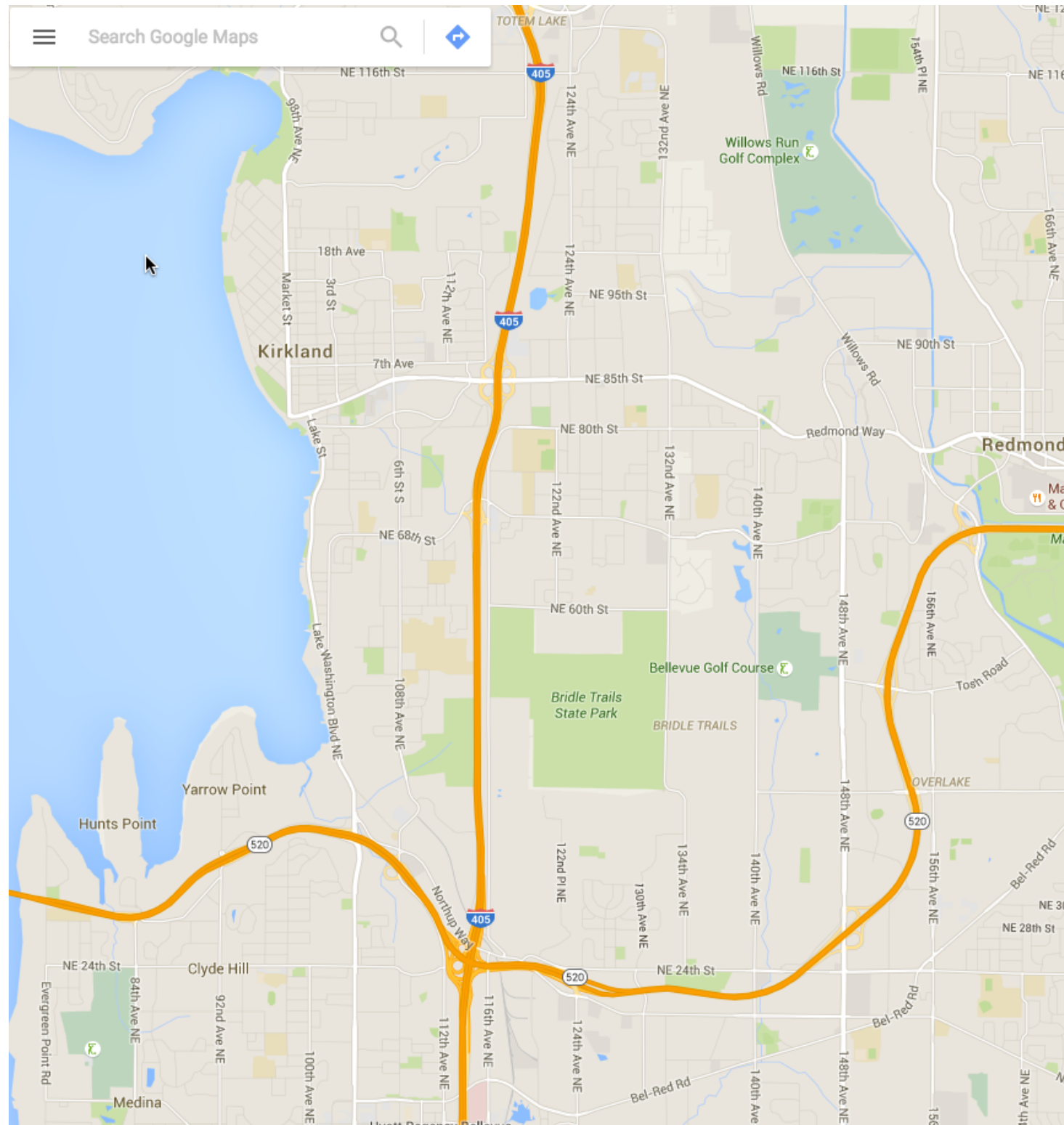
8:12p DCA → **9:26p** ORD 2h 14m nonstop
11:55a ORD → **2:42p** DCA 1h 47m nonstop

[View Deal](#)

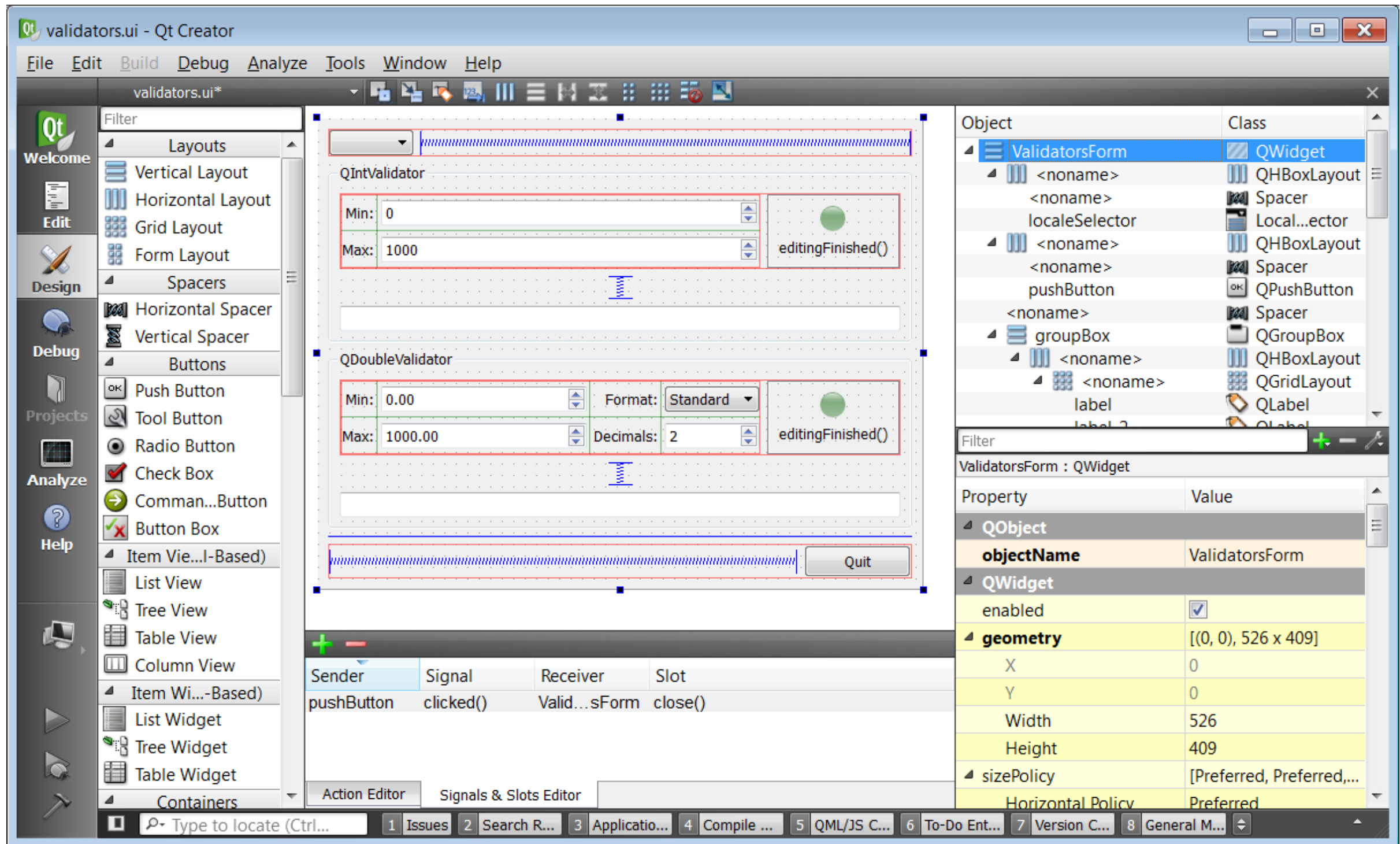
[Show details](#)

Economy

Example - Google Maps



Example - GUI builder



Example - Spreadsheets

FlyCalc - WIG2004.XLS

File Edit View Insert Format Tools ?

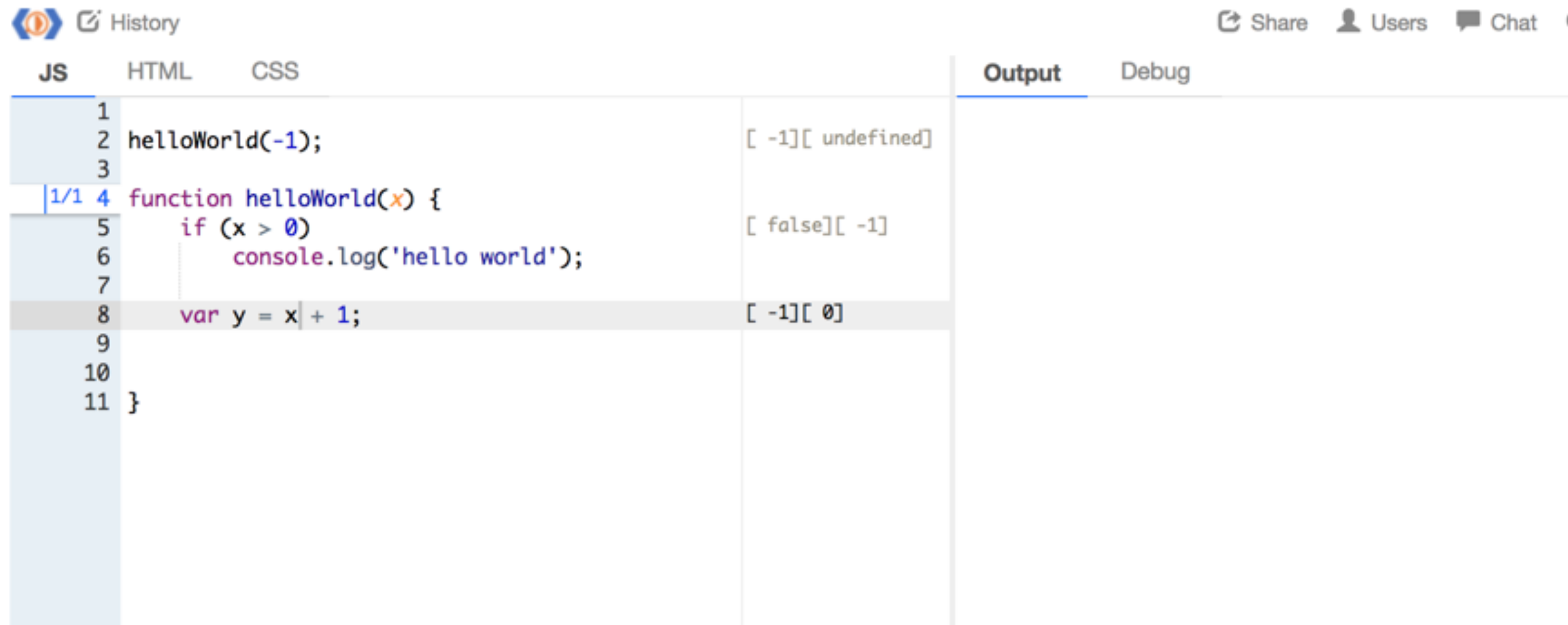
100% Support Chat OFF

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: live programming



The image shows a live programming interface with a code editor on the left and an output/debug console on the right. The code editor has tabs for JS, HTML, and CSS, with JS selected. The code in the JS tab is as follows:

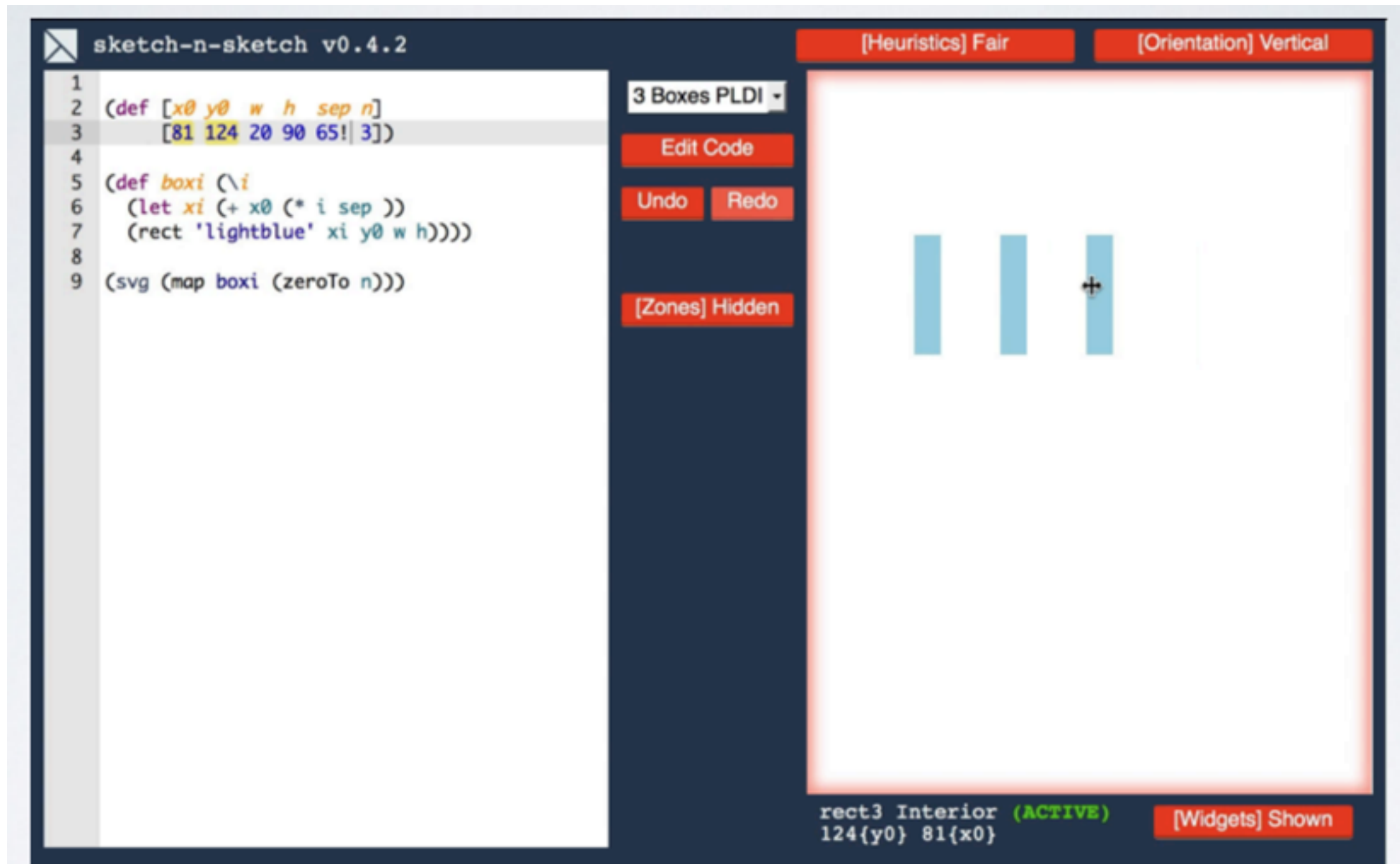
```
1  
2 helloWorld(-1);  
3  
1/1 4 function helloWorld(x) {  
5     if (x > 0)  
6         console.log('hello world');  
7  
8     var y = x + 1;  
9  
10  
11 }
```

The output/debug console on the right has tabs for Output and Debug, with Output selected. It shows the following output:

Line	Output
2	[-1][undefined]
5	[false][-1]
8	[-1][0]

The interface also includes a History icon, a Share icon, a Users icon, and a Chat icon in the top right corner.

Example: edit constants by editing output



Chugh et al. [PLDI '16]

In-class activity

In Class Activity: Direct Manipulation Programming Interactions

- In groups of 2
 - Design a system for writing 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 exhibits intended behavior