

Human Computer Interaction (CSE 556A)

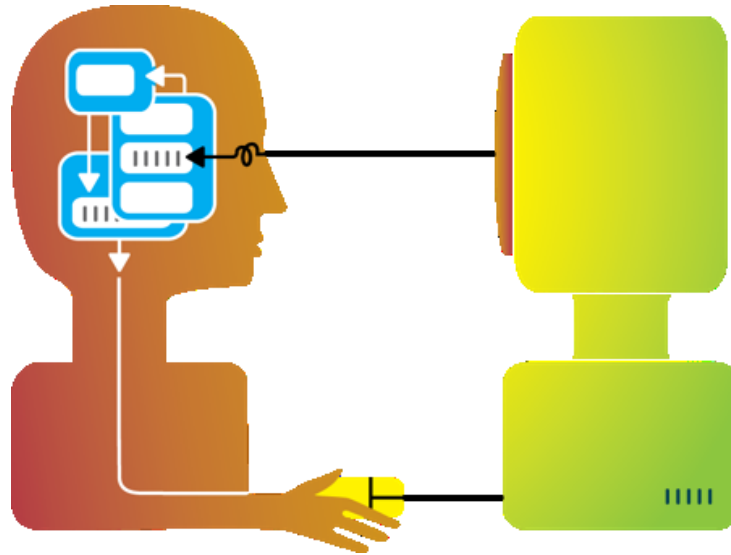
Kyle J. Harms

What is HCI?

The study of computers and people as a single system.

Much of HCI employs a set of engineering techniques focused on improving the performance of a computer-human system.

HCI as a discipline: a brief history



1945 - Vannevar Bush – As We May Think



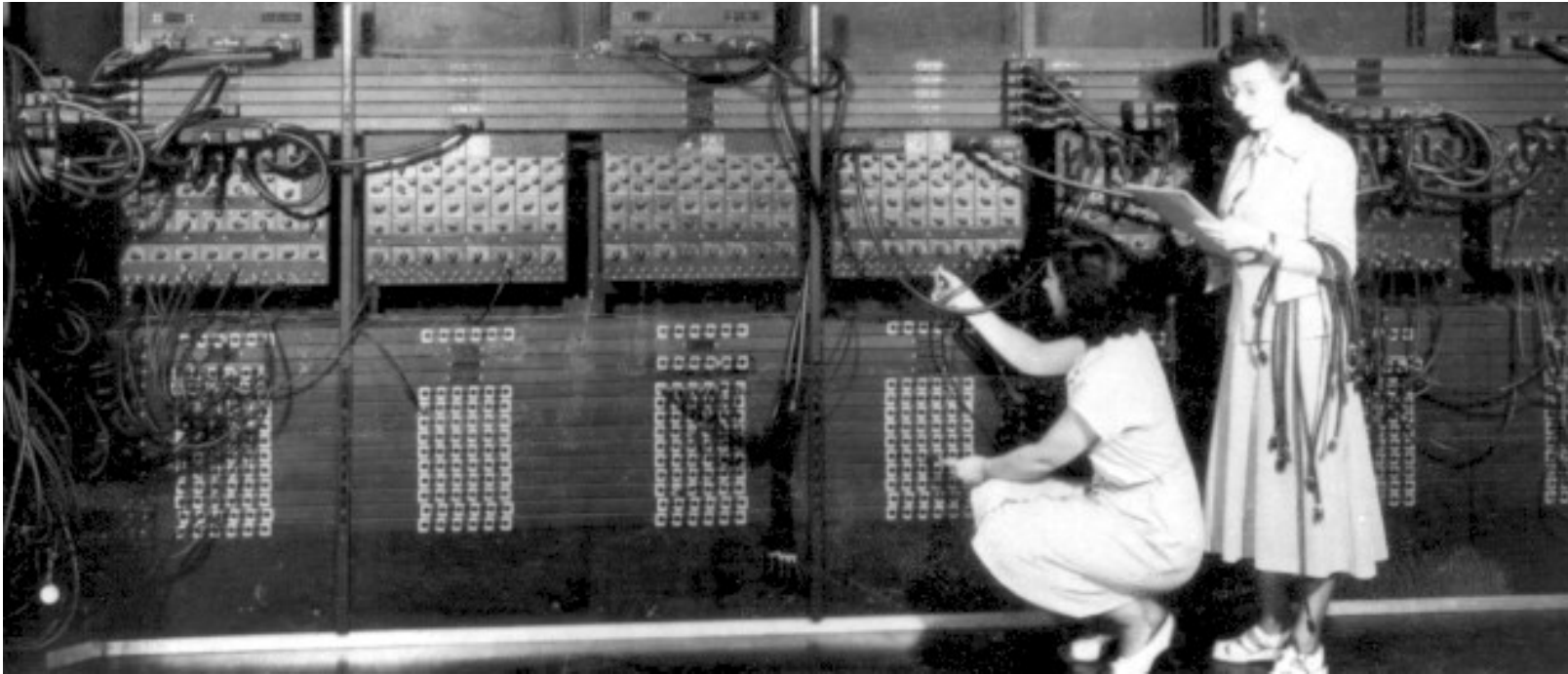
- “As we Think” – Atlantic Monthly
- Hypothetical MEMEX device
 - Idea based on microfilm-based machine
 - A device that stores all records
 - Cross references and indexing
 - Save a trail of where you’ve been
- Sounds like HTML and the web...

1946 - ENIAC



- Probably the first general purpose computer
- Developed for calculations for hydrogen bomb
- “I was astounded that it took all this equipment to multiply 5 by 1000” (ENIAC occupied 1800 Sq. Ft.)

Programming ENIAC

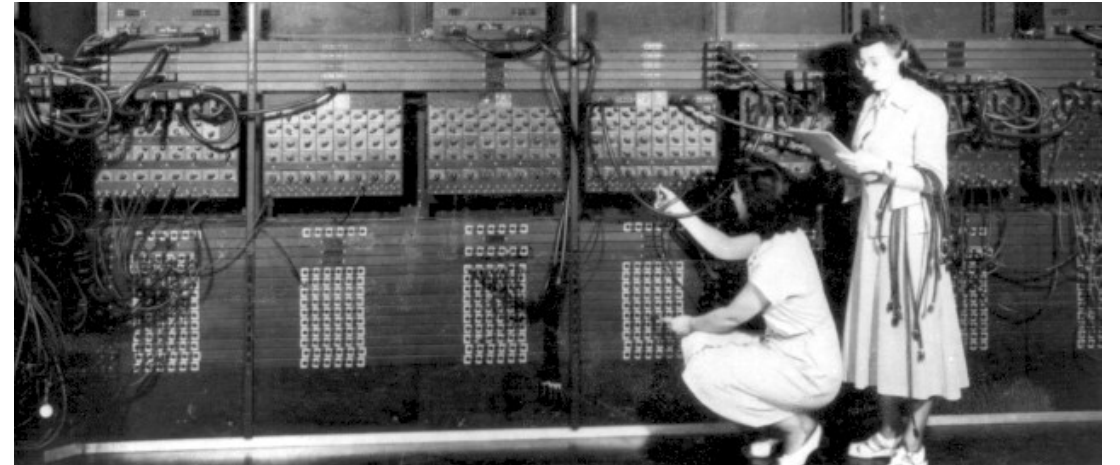


Programmed: plug board and switches
Input/output: cards, lights, switches, plugs

To compute *circumference* = $3.14 * \text{diameter}$;
Required rearranging a ton of path cords and then locating
three knobs on a wall and setting them to 3, 1, and 4.

Programming ENIAC

- Programming team of 6 woman
- Days to load program
 - Set switches
 - Set dials
 - Connect cables
- Vacuum tubes failed a lot



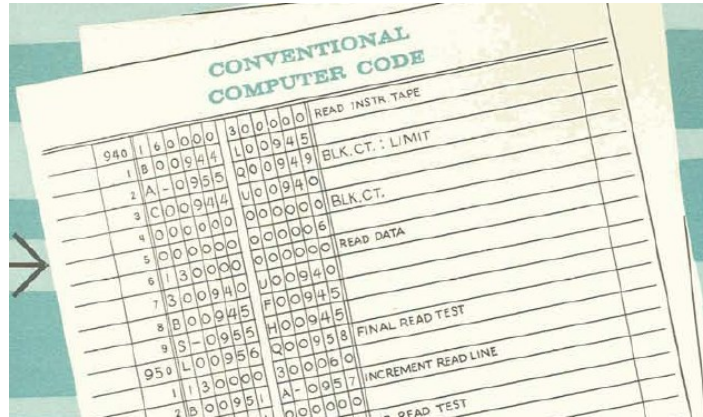
1951 – UNIVAC I



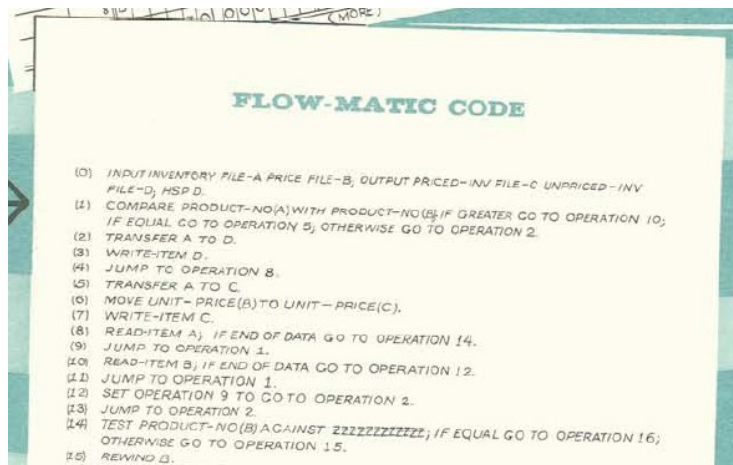
The UNIVAC I delivered to the U.S. Census Bureau.

Remington Rand eventually sold 46 machines at nearly \$1 million each (\$750,000 plus \$185,000 for a high speed printer.)

Programming UNIVAC



Line	Code	Binary	Operation
940	1 6 0 0 0 0	3 0 0 0 0 0	READ INSTR. TAPE
1	8 0 0 9 4 5	1 0 0 9 4 5	
2	A - 0 9 5 5	0 0 0 9 4 9	BLK. CT.: LIMIT
3	C 0 0 9 4 4	U 0 0 9 4 0	
4	0 0 0 0 0 0	0 0 0 0 0 0	BLK. CT.
5	0 0 0 0 0 0	0 0 0 0 0 6	
6	1 3 0 0 0 0	0 0 0 0 0 0	READ DATA
7	3 0 0 9 4 0	U 0 0 9 4 0	
8	8 0 0 9 4 5	F 0 0 9 4 5	
9	S - 0 9 5 5	H 0 0 9 4 5	FINAL READ TEST
950	L 0 0 9 5 6	Q 0 0 9 5 8	
1	1 3 0 0 0 0	3 0 0 0 6 0	INCREMENT READ LINE
2	0 0 0 9 5 1	A - 0 9 5 7	
	0 0 0 0 0 0	0 0 0 0 0 0	READ TEST



FLOW-MATIC CODE

(0) INPUT INVENTORY FILE-A PRICE FILE-B; OUTPUT PRICE-INV FILE-C UNPRICED-INV FILE-D; HSP D.

(1) COMPARE PRODUCT-NO(A) WITH PRODUCT-NO(B); IF GREATER GO TO OPERATION 10; IF EQUAL GO TO OPERATION 5; OTHERWISE GO TO OPERATION 2.

(2) TRANSFER A TO D.

(3) WRITE-ITEM D.

(4) JUMP TO OPERATION 8.

(5) TRANSFER A TO C.

(6) MOVE UNIT-PRICE(B) TO UNIT-PRICE(C).

(7) WRITE-ITEM C.

(8) READ-ITEM A; IF END OF DATA GO TO OPERATION 14.

(9) JUMP TO OPERATION 1.

(10) READ-ITEM B; IF END OF DATA GO TO OPERATION 12.

(11) JUMP TO OPERATION 1.

(12) SET OPERATION 9 TO GO TO OPERATION 2.

(13) JUMP TO OPERATION 2.

(14) TEST PRODUCT-NO(B) AGAINST ZZZZZZZZZZ; IF EQUAL GO TO OPERATION 16; OTHERWISE GO TO OPERATION 15.

(15) REWIND D.

- Interaction
 - Write a program
 - Submit it
 - Wait for results
- Programs no longer all in numeric form but commands still map very closely to what was going on in the circuitry.

UNIVAC 1 Predicting Outcome of the 1952 Election

8.30 P.M.

IT'S AWFULLY EARLY, BUT I'LL GO OUT ON A LIMB.

UNIVAC PREDICTS--with 3,398,745 votes in--

	STEVENSON	EISENHOWER
STATES	5	43
ELECTORAL	93	438
POPULAR	18,986,436	32,915,049

THE CHANCES ARE NOW **00 to 1** IN FAVOR OF THE ELECTION OF EISENHOWER.

- Filmed live by CBS
- “You're a very impolite machine I must say, but he's an awfully rapid calculator.” – Charles Collingwood, Reporter

Computer vs. Humans in 1951

- UNIVAC I: \$935,000
- Average worker salary: \$3350
- Cost of 1 computer \sim 280 workers for a year
 - Computer time is much more valuable than human time. It makes sense to focus almost entirely on the computer in isolation.

Good Software in 1951

- Optimize for computer time:
 - Make sure you are performing the right calculation (correctness)
 - Don't take any more computer time than you have to (efficiency)
- Correctness and efficiency seem familiar?

J.C.R Licklider

Proposed the need for HCI in 1960

Wash U alum in physics, math, and psychology



Image from <http://www.cpedia.com/wiki/Licklider/Mit>

Need for Interactive Computing

“Imagine trying, for example, to direct a battle with the aid of a computer on such a schedule as this. You formulate your problem today. Tomorrow you spend with a programmer. Next week the computer devotes 5 minutes to your program and 47 seconds to calculating the answer to your problem. You get a sheet of paper 20 feet long, full of numbers that, instead of providing a final solution, only suggest a tactic that should be explored by simulation.”

- Licklider, 1960

J.C.R. Licklider (1960)

- Outlined “man-computer symbiosis”

“The hope is that, in not too many years, human brains and computing machines will be coupled together very tightly and that the resulting partnership will think as no human brain has ever thought and process data in a way not approached by the information-handling machines we know today.”

Licklider's Suggested Immediate Goals

- time sharing of computers among many users
- electronic i/o for the display and communication of symbolic and pictorial information
- interactive real time system for information processing and programming
- large scale information storage and retrieval

Licklider's Suggested Longer Term Goals

- Interactive Graphics
- Shared, collaborative displays
- Speech based interfaces

Ivan Sutherland – SketchPad(1963)



(Alan Kay on...) <http://www.youtube.com/watch?v=495nCzxM9PI>

(Lincoln Lab video...) <http://www.youtube.com/watch?v=T7dC98PNxyE>

Sketchpad Ideas that are still with us

- hierarchical structures (e.g. pictures and sub-pictures)
- object-oriented programming: master picture with instances
- constraints: specify details which the system maintains through changes
- icons: small pictures that represented more complex items
- copying: both pictures and constraints
- input techniques: efficient use of light pen
- world coordinates: separation of screen from drawing coordinates
- recursive operations: applied to children of hierarchical objects

The First Mouse (1964) – Douglas Engelbart



Doug Engelbart at the AFIP Fall Conference 1968

Document Processing

- modern word processing
- outline processing
- hypermedia

Input / Output

- the mouse and one-handed corded keyboard
- high resolution displays
- multiple windows
- specially designed furniture

Shared work

- shared files and personal annotations
- electronic messaging
- shared displays with multiple pointers
- audio/video conferencing
- ideas of an Internet

User testing, training

<http://www.youtube.com/watch?v=1MPJZ6M52dI>



Slide adapted from Saul Greenberg

Optimize for Skill or Use?

- Douglas Engelbart
 - Difficult interface = great power
- Lost funding due to difficulty of mastering the interface

Licklider's Suggested Longer Term Goals

- ✓ Interactive Graphics
- ✓ Shared, collaborative displays
- Speech based interfaces

We have the first hints of 2 out of 3 by the end of the 60s.

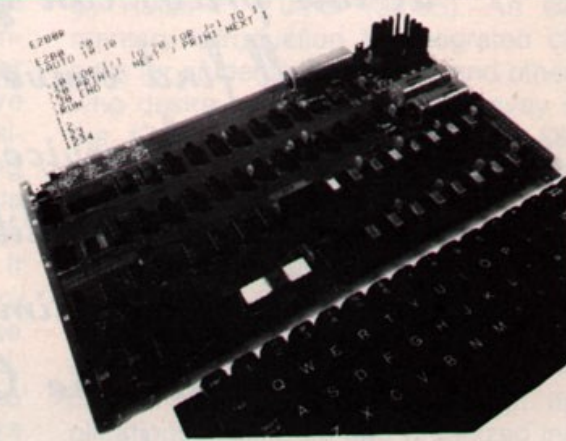
1976

The commercial world tends to lag behind research.

A BALANCE OF FEATURES

The APPLE-1 SYSTEM is a fully assembled, tested & burned-in microprocessor board using the 6502 microprocessor. The board contains processor & support hardware; **complete video electronics** for a 40 character/line, 24 line video display; **on-board RAM capacity of 8K BYTES**; software system monitor in PROM; and fully regulated power supplies. The Apple attaches directly to an ASCII encoded keyboard and a video monitor, allowing the efficient entry and examination of programs in hexadecimal notation. The use of the new **16-pin 4K RAM chips** results in low power and high density memory, which can be upgraded to the 16K chips when they become available (32K bytes on-board RAM!!)

A fast (1 kilobaud) cassette interface is available and includes a tape of **Apple Basic**. And ... Yes, Folks, **Apple Basic is Free!**



APPLE-1 **\$666.66**
*includes 4K bytes RAM

- | | |
|------------------------|--|
| Micro Interface | <ul style="list-style-type: none">• 6502 Microprocessor• Full video display electronics - 40 char/line, 24 line. Outputs composite video.• Has ASCII keyboard interface on-board.• Cassette interface board available. FAST - 1 Kilobaud. |
| Memory | <ul style="list-style-type: none">• Uses 16-pin 4K Dynamic RAMS.• 8K BYTE RAM capacity on-board!• Upgradable to 16K RAM chips.• Software system monitor in PROM |
| Basic | <ul style="list-style-type: none">• Apple Basic ... pseudo-compiled, FAST, FREE. |
| Power | <ul style="list-style-type: none">• Fully regulated power supplies on-board. |

DEALER INQUIRIES INVITED

APPLE COMPUTER COMPANY

770 Welch Road, Suite 154
Palo Alto, California 94304

Phone: (415) 326-4248

CIRCLE NO. 42 ON INQUIRY CARD

JULY 1976

1980:Early Speech and Gesture



The Architecture Machine Group at MIT

<http://www.youtube.com/watch?v=RyBEUyEtxQo>

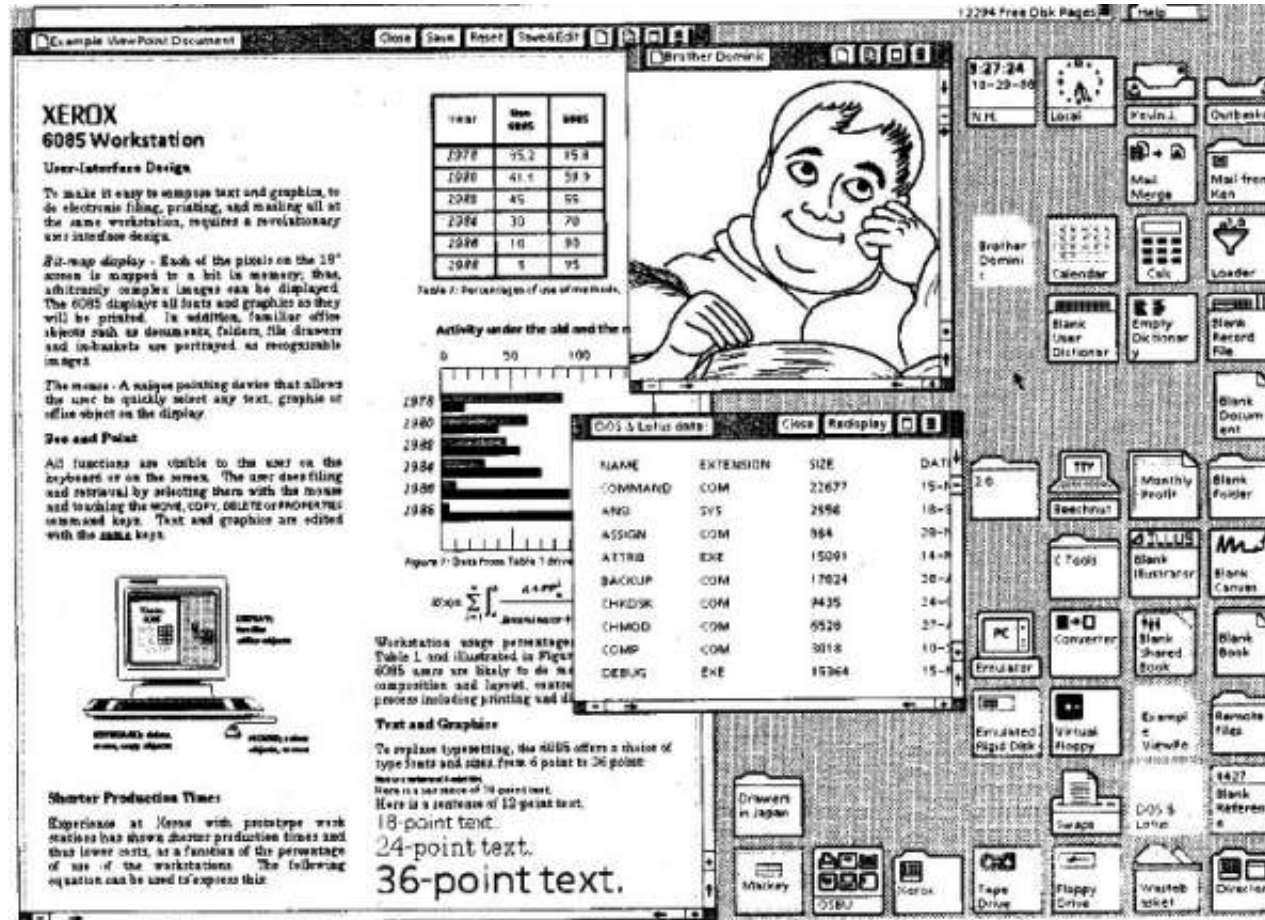
Licklider's Suggested Longer Term Goals – 1980s

- ✓ Interactive Graphics
- ✓ Shared, collaborative displays
- ✓ Speech based interfaces

Computer vs. Humans in 1981

- Apple Lisa: \$10,000
- Average salary: \$ 16,855 (men) and \$7928 (women)
- Cost of 1 computer \sim 0.6 to 1.26 workers for a year
 - The balance is shifting and companies are starting to think about prioritizing for both human and computer.

Xerox Star (1981)



Bringing in interactive graphics: <http://www.youtube.com/watch?v=QYvxgNhUwBk>

Xerox Star-1981

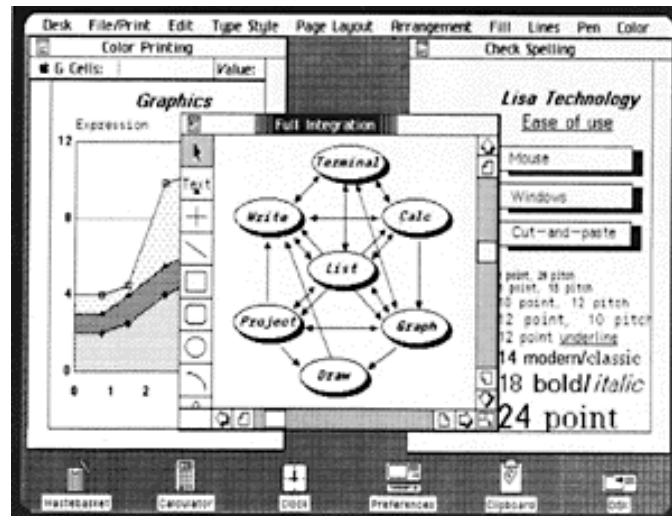
- First use of usability testing in design
- First commercial personal computer designed for “business professionals”
- First comprehensive GUI used many ideas developed at Xerox PARC
 - familiar user’s conceptual model (simulated desktop)
 - promoted recognizing/pointing rather than remembering/typing
 - property sheets to specify appearance/behaviour of objects
 - what you see is what you get (WYSIWYG)
 - small set of generic commands that could be used throughout the system
 - high degree of consistency and simplicity
 - modeless interaction
 - limited amount of user tailorability

Xerox Star (continued)

- Commercial failure
 - cost (\$15,000);
 - IBM had just announced a less expensive machine
 - limited functionality
 - e.g., no spreadsheet
 - closed architecture,
 - 3rd party vendors could not add applications
 - perceived as slow
 - but really fast!
 - slavish adherence to direct manipulation

Commercial Machines: Apple Lisa (1983)

- based upon many ideas in the Star
 - predecessor of Macintosh,
 - somewhat cheaper (\$10,000)
 - commercial failure as well



Quicken 1.0 for DOS (1984)

Print/Acct		Edit	Shortcut	Report	Activity	Graph	F1-Help	
Date	Num	Payee	Memo	Category	Payment	C	Deposit	Balance
BEGINNING								
10/02		Opening Balance				X	10,000 00	10,000 00
2009		[Testing Accou→						
10/02	0001	Wikipedia Foundation			1,000 00			9,000 00
2009		Charity						
10/02		Memo:						
2009		Cat:						
Ctrl-F8 to Expand Qcards								
accesses menu>								
Ending Balance: \$9,000.00								

Quicken 8.0

Quicken (1984)

"... in the first instance of the Usability Testing that later became standard industry practice, LeFevre recruited people off the streets... and timed their Kwik-Chek ([Quicken](#)) usage with a stopwatch. After every test... programmers worked to improve the program.") [Scott Cook](#), [Intuit](#) co-founder, said, "... we did usability testing in 1984, five years before anyone else... there's a very big difference between doing it and having marketing people doing it as part of their... design... a very big difference between doing it and having it be the core of what engineers focus on."

- From Wikipedia Entry on Usability Testing

Getting Quicken to Market (late 1980's - early 1990's)

- The competitors:
 - Tech Savviest Audience Member using competitor's personal finance software
 - Least Tech Savvy Audience Member using Quicken without a manual.
- The challenge:
 - Write and print a personal check.
- Results
 - Quicken users won in 3-5 minutes. Users of competitor's system typically gave up.

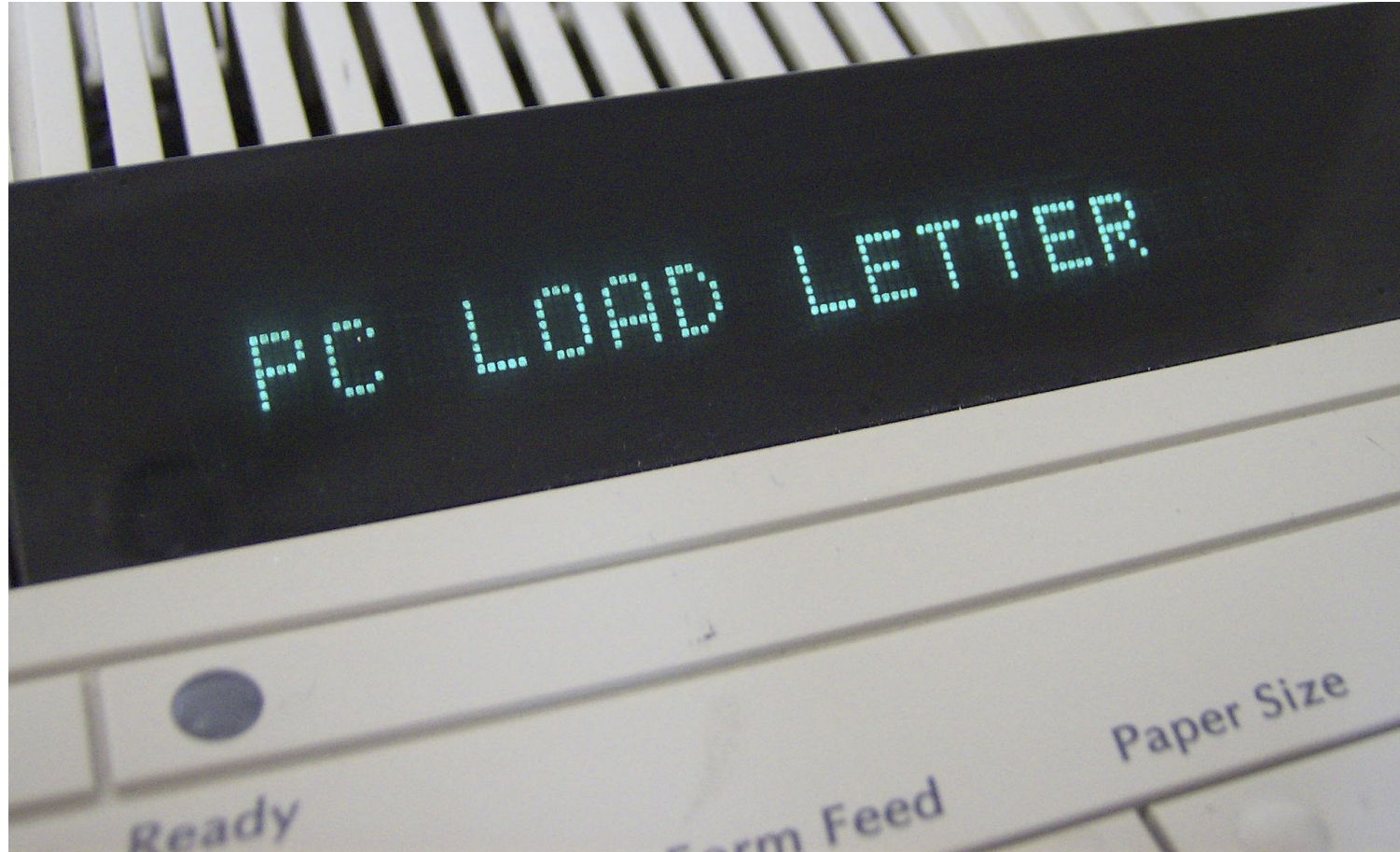
Getting Quicken to Market (late 1980's - early 1990's)

- Intuit's revenues grew from \$6 million in 1988 to \$33 million in 1990.
- By 1993, revenues are at \$200 million.
- Usability was a key ingredient in their success. Other companies started to applying usability techniques during design.

Computer vs. Human Costs

- Typical PC: ~1,000
- Average worker salary: \$42,800 (full time male), \$34,700 (full time female)
- Cost of 1 computer \approx 2.6% of one worker for one year.
 - It no longer makes sense to optimize exclusively for computation time. We need to also optimize for human time.

We don't always put the human first... 1990s

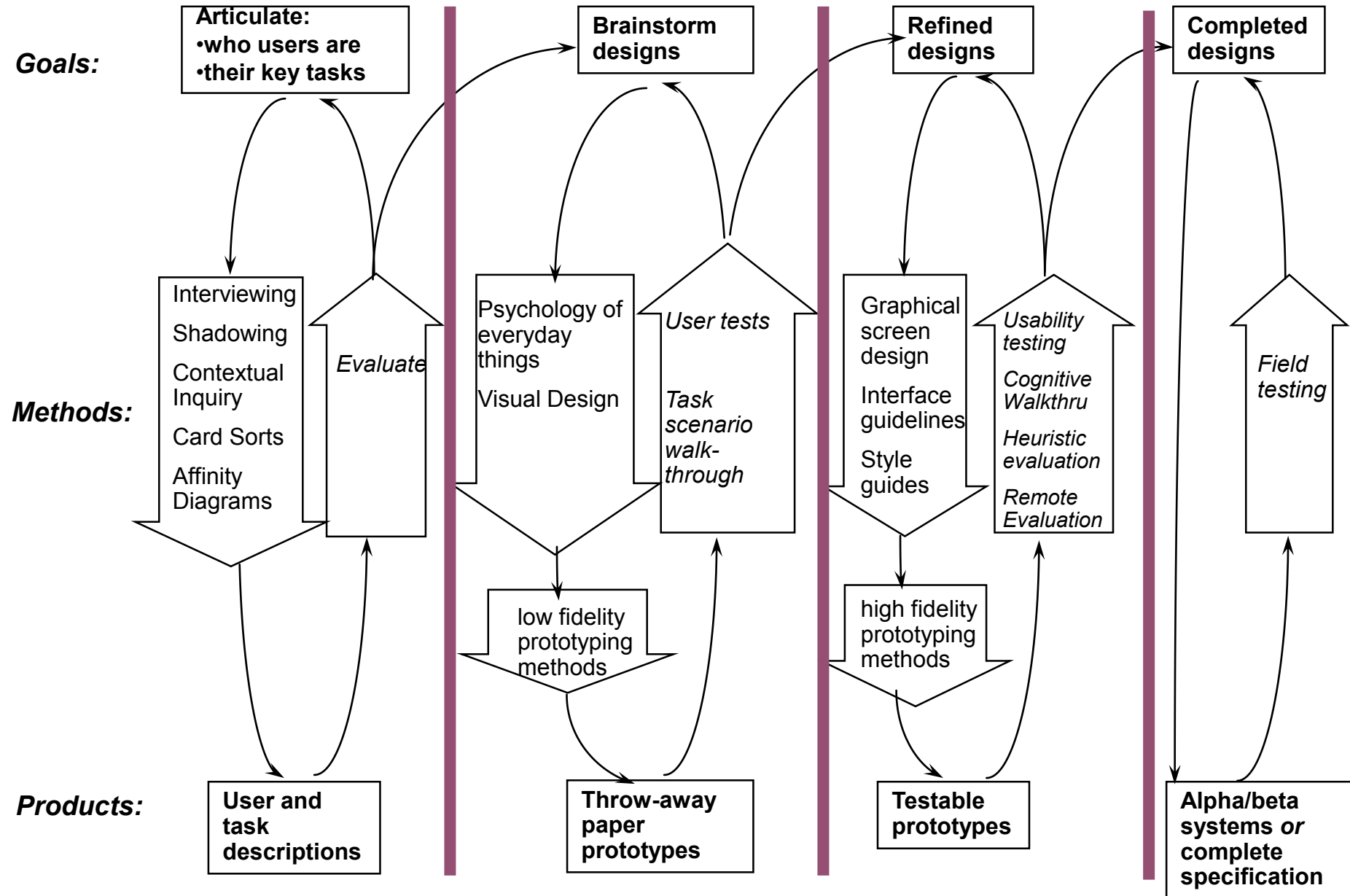


Today

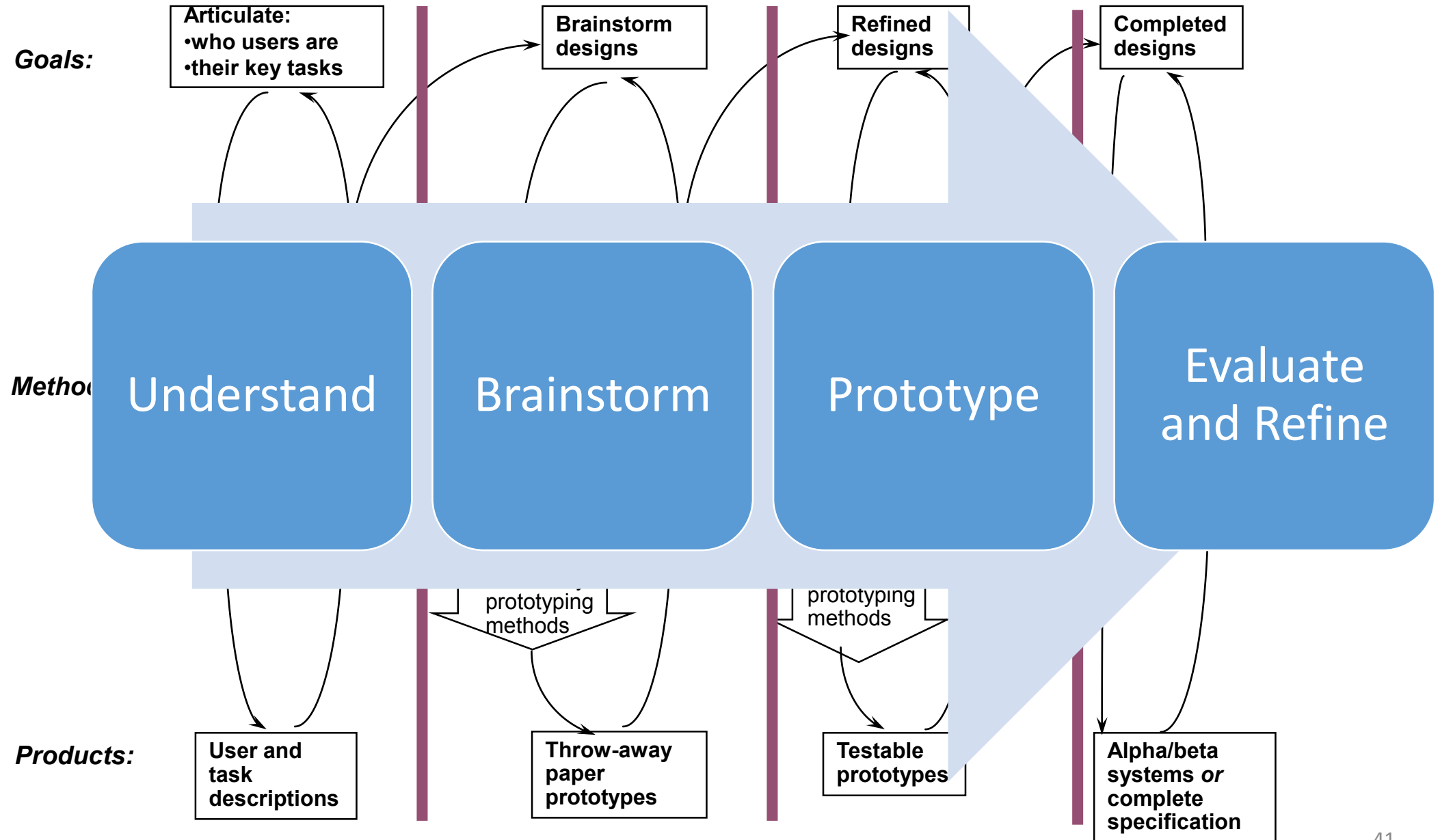
- Most of the big software development companies do usability work
 - Some have special divisions and labs
 - A few integrate usability into development teams
- We've developed methods and processes for evaluating human-computer systems.
- But, not yet pervasive.

Design Process

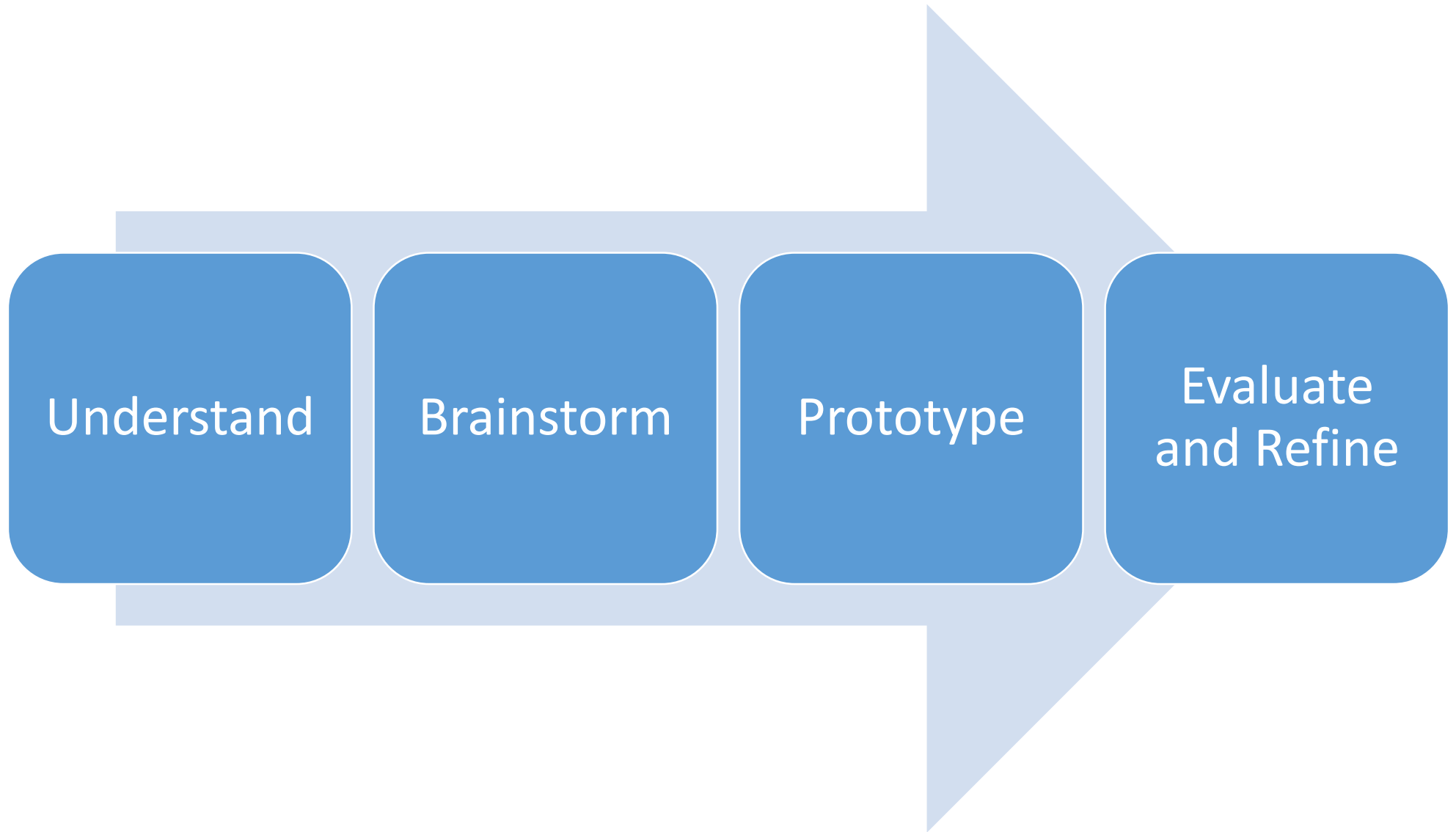
An interface design process

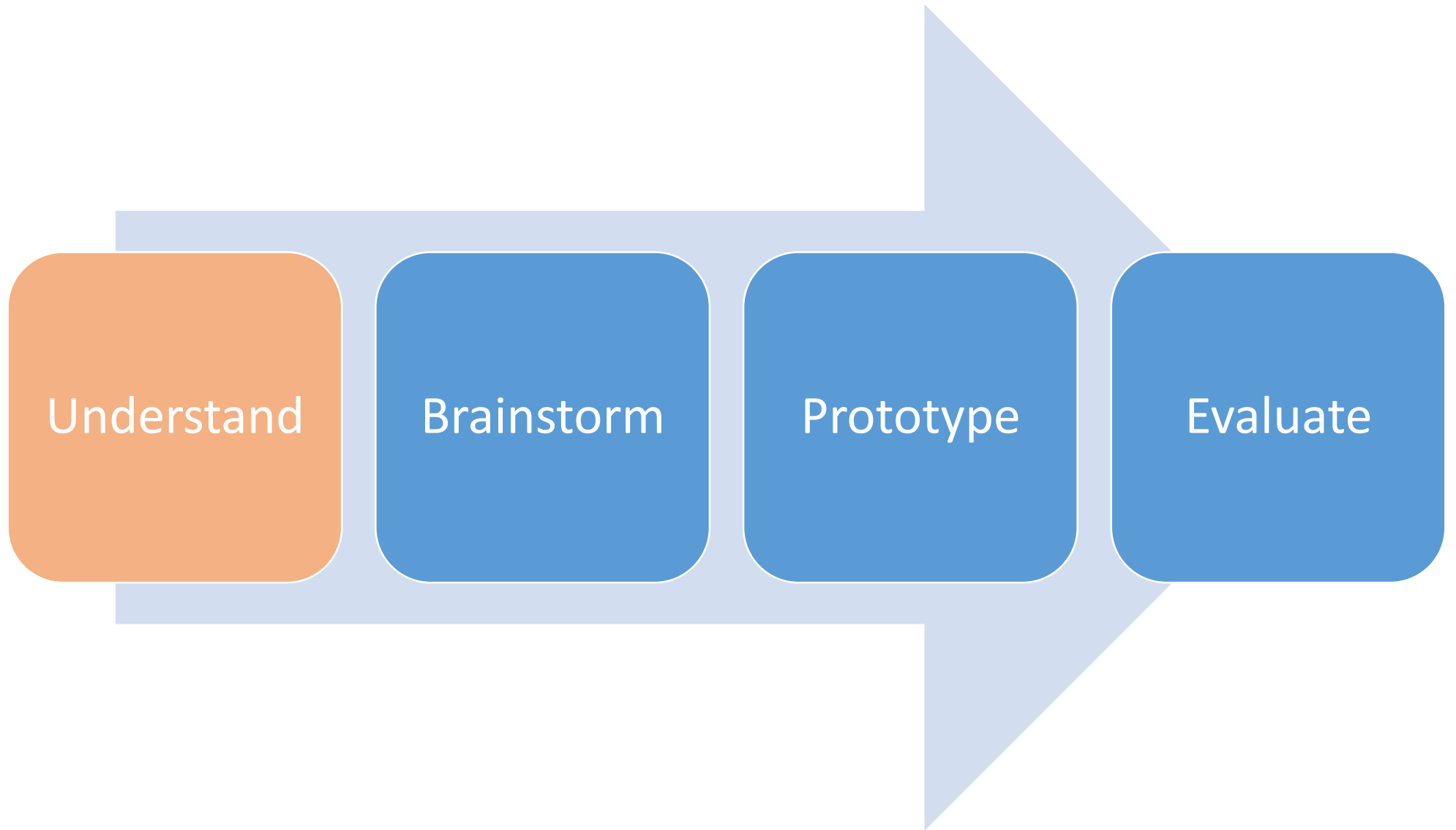


An interface design process



Some examples from my research lab's work...





Research Question

- There are numerous sources of code examples in varying forms available on the web.
- How can we better enable inexperienced programmers to take advantage of them?
- Specifically, when shared programs contain output of interest, can inexperienced programmers determine which lines of code are responsible for the behavior of interest?

Expected Use



I want that
part...where he
bends to pick up the
banana.



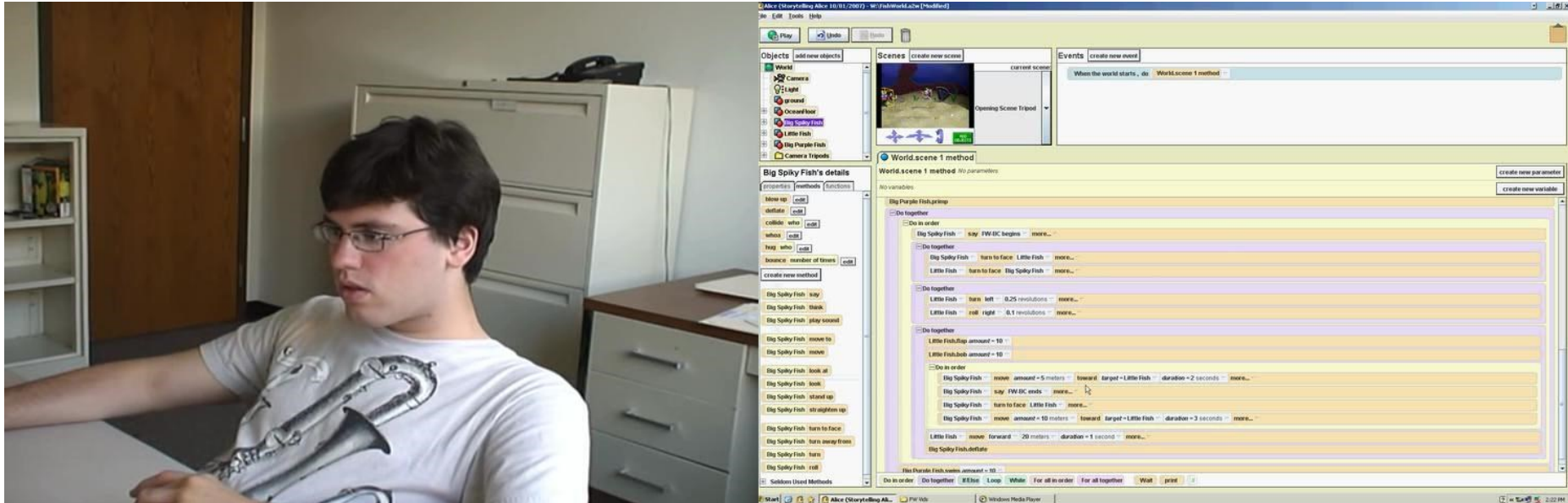
Tasks Mirror Expected Use

- Bounding Tasks
 - Denote begin, end of highlighted functionality
- Modification Tasks
 - Modify highlighted functionality
- 5 tasks for each of 4 programs.



For this task you will have to modify the Fish World.

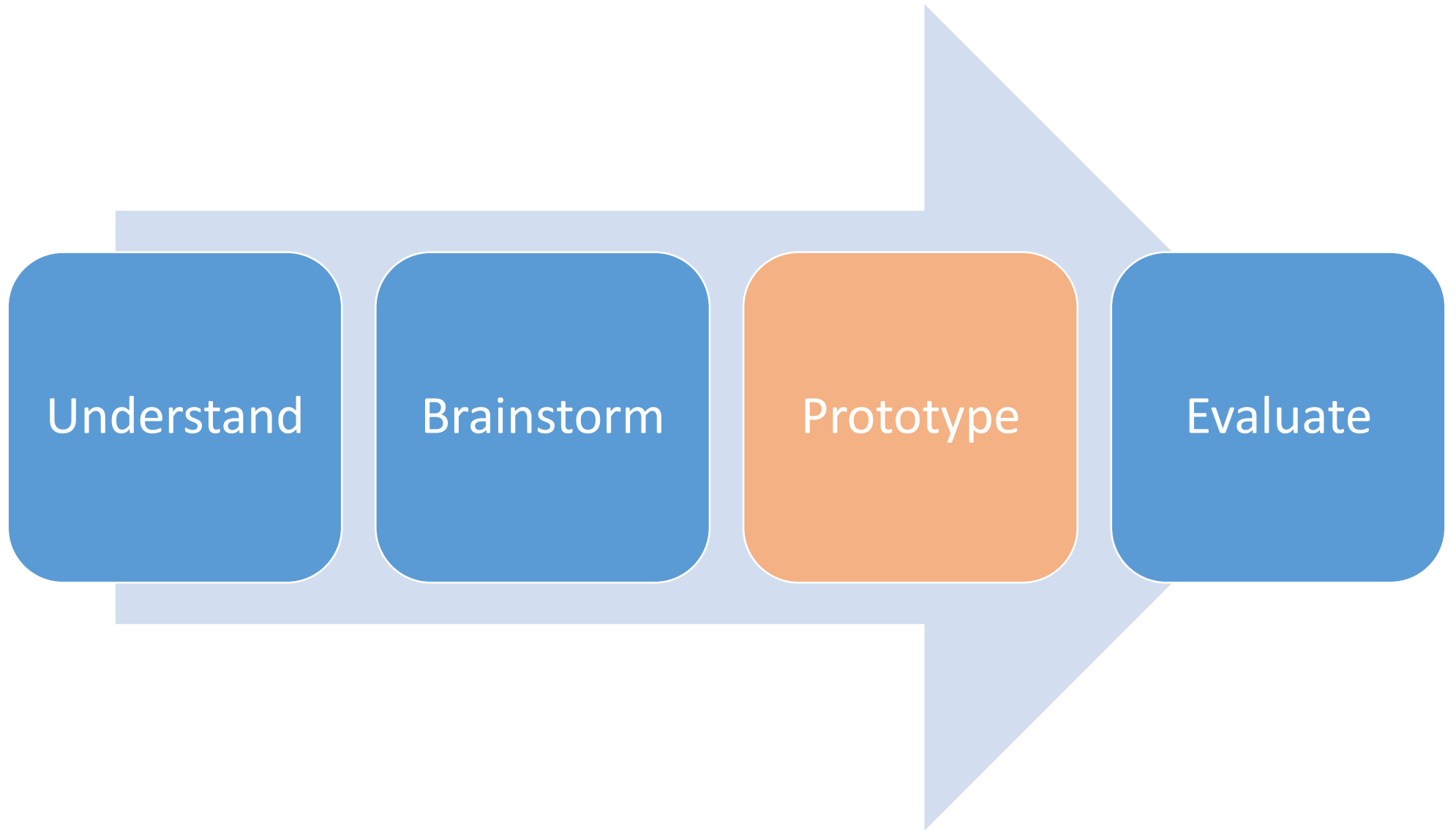
Task Data Collection



- Computing history, programming experience
- Video/audio recording of users, screen
 - Talk-aloud protocol

This is hard for novices.

- 41% correct answers
 - 33% correct bounding
 - 72% correct modification
- Bounding task time range: 01:02 – 26:20
- Modification task time range: 01:07 – 27:29



Research Question

- We know that inexperienced programmers attempt to perform a bi-directional search in order to identify code statements responsible for output of interest.
- How can we best support this mapping in a programming environment?

Version 1

The screenshot displays the Looking Glass 3.0 beta software interface. The main window is titled "Looking Glass 3.0 beta.0025 C:\Documents and Settings\ckelleher\My Documents\LookingGlass\MyProjects\KissExample.a3p". The interface is divided into several panels:

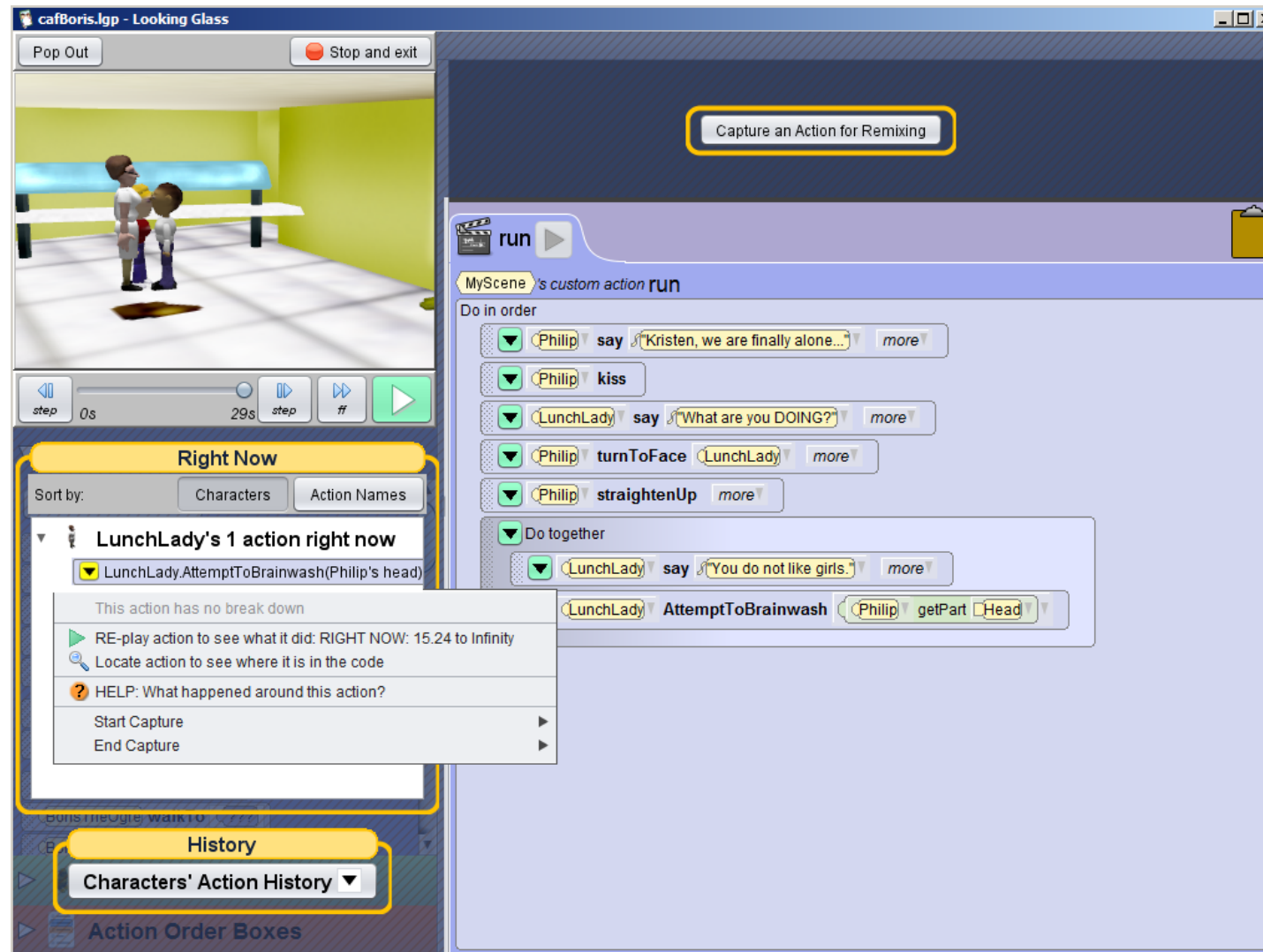
- Top Panel:** A timeline slider showing "0.00 seconds" to "36.00 seconds". A green box above the slider asks "What was happening at 31.58 seconds?".
- Left Panel:** A small 3D scene window titled "The scene at 31.58 s." showing two characters in a room. Below it, a panel titled "Everyone's actions at 31.58 s." lists actions for "lunchLady" and "philip".
- Right Panel:** A large script editor showing a "scene.run()" procedure. The script includes several blocks: "showTitle", "say", "kiss", "turnToFace", "do together" (containing "AttemptToBrainwash_lg", "delay", and "say"), and "walkOffscreen".

The script blocks are numbered 1 through 7. The "do together" block is highlighted with a green box and labeled "zoom into Do Together". The "kiss" block is labeled "explore kiss". The "say" block in the "do together" block is labeled "You do not like girls.". The "walkOffscreen" block is labeled "this.lunchLady walkOffscreen".

Version 1 Problems

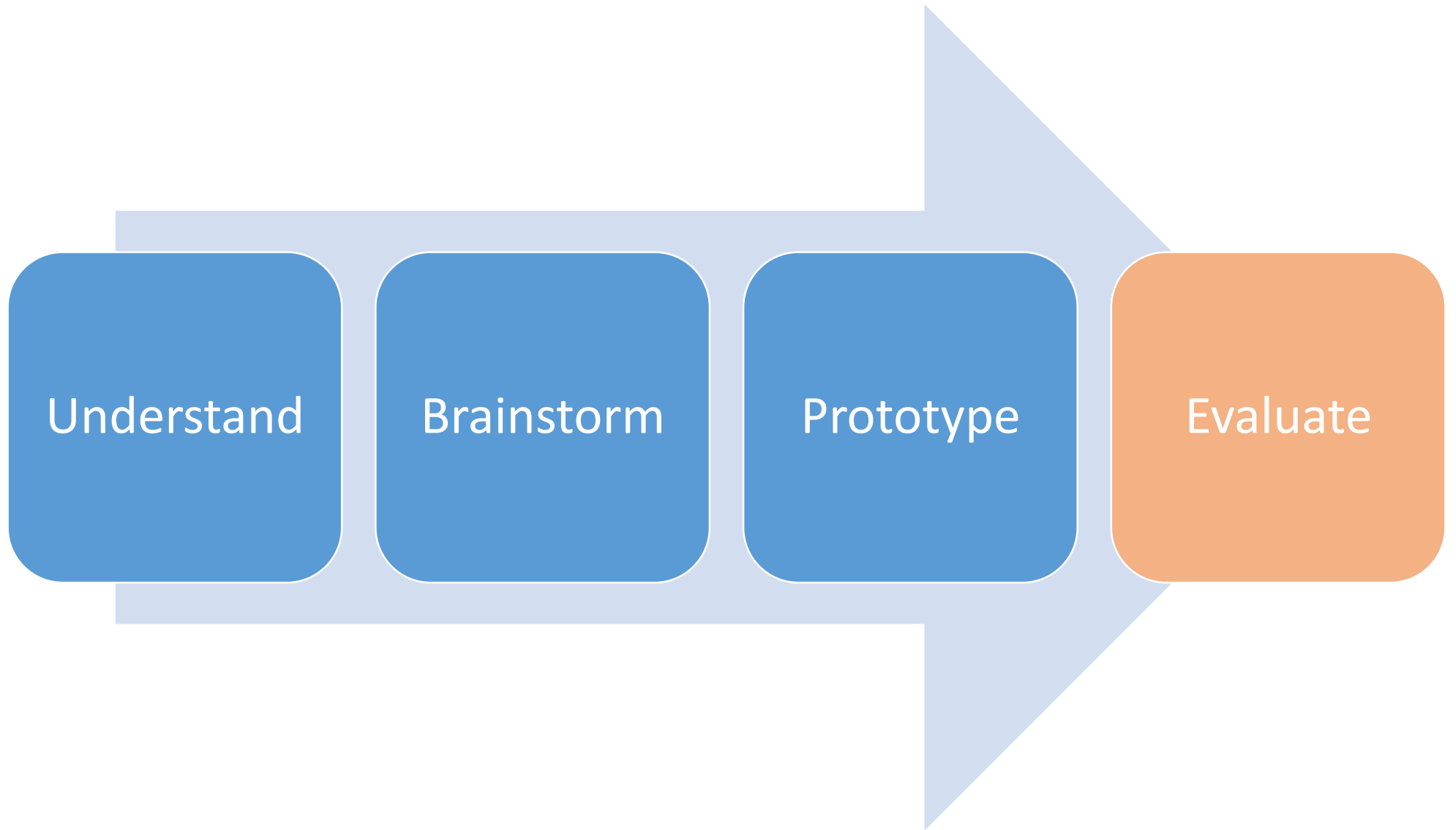
- Once people get into the code, they prefer to reason from there.
 - While our prototype allowed navigation in the same method, users must go from the output again to reach code in a different method.
- Recorded executions can't separate out multiple statements occurring simultaneously.
 - This can lead to incorrect conclusions about statements occurring in parallel.

Supporting Reasoning from Code



Version 2 Problems

- Too many different paths to finding a particular action became overwhelming for users.
- The overlay approach no longer has value, but costs a fair bit in performance.



Research Question

- Do inexperienced programmers learn new programming concepts more effectively using code puzzles or tutorials?

Programming Completion Problems: Code Puzzles

The image shows a programming puzzle interface for a game titled "alien-and-ufo.lgp - Looking Glass (0)". The interface is divided into several panels:

- Panel C (Top Left):** A 3D scene showing a green alien standing next to a flying saucer on a grey, rocky surface.
- Panel D (Bottom Left):** A control panel with a "Play Correct" button and a "Play Mine" button. Below them is a red and white checkered icon and the text: "Use all of these actions to put the animation back in the correct order." A dashed box contains a "Do together" block with a "flying saucer beam up alien" block inside it. Below this is a "flying saucer fly away" block. A purple arrow points from the "Do together" block to the "Repeat" block in Panel B.
- Panel B (Center):** A "custom action My Story" block with a yellow background. It contains the following code:

```
alien walk 0.5 meters
Repeat 2 times
  alien kick
  flying saucer shake
loop
flying saucer fly above alien
flying saucer turn RIGHT 2.0 rotations duration 2.0 seconds
```
- Panel E (Right):** A "Play" window showing the 3D scene with the alien and the flying saucer. Below the scene is a progress bar with seven colored squares (six green, one orange, one grey). A blue arrow points from the progress bar to the "Fast Forward" button.
- Panel F (Bottom Right):** A "Fast Forward" button.

Integrated Tutorials



Study: 34 participants, randomly assigned

Control

1. 6 tutorial training tasks
2. 4 transfer tasks
(programs with all method calls but no programming constructs – add missing constructs to complete)

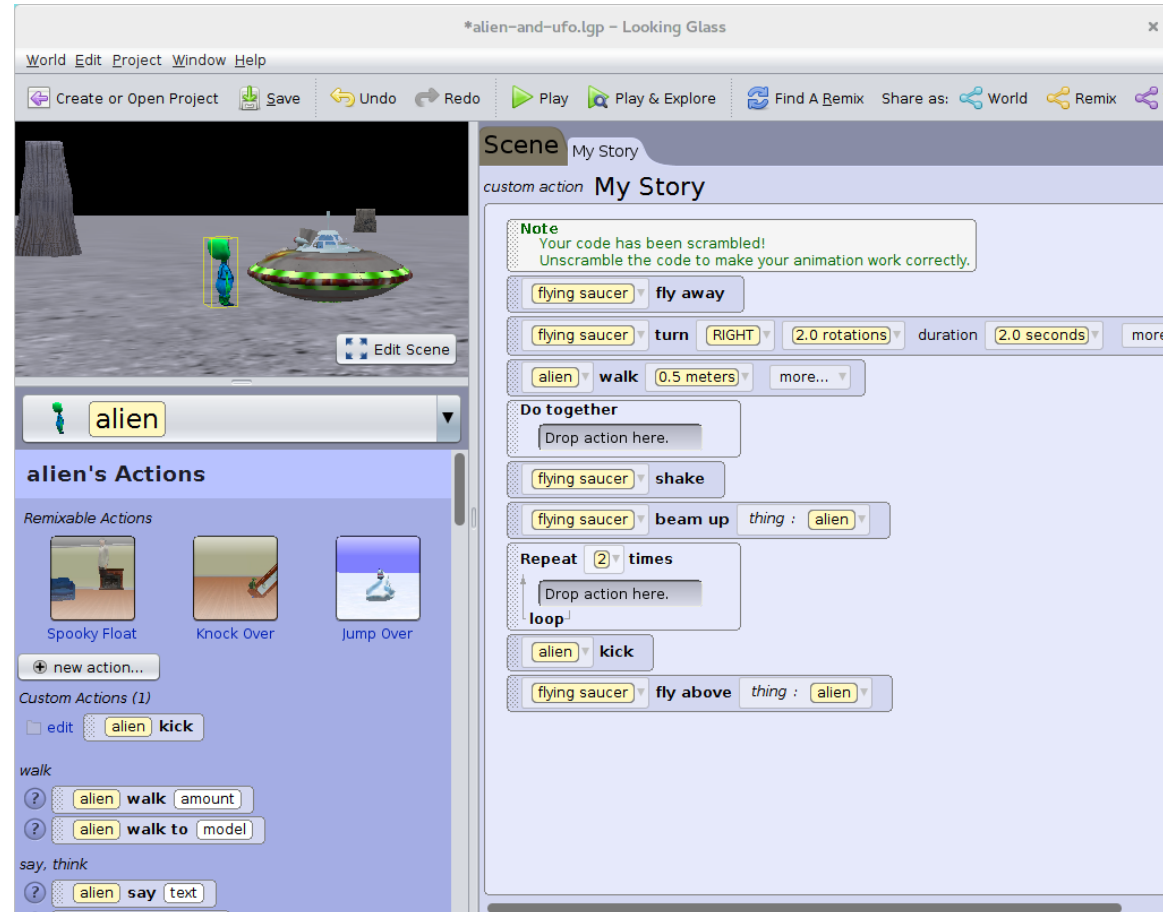
Experimental

1. 6 puzzle training tasks
2. 4 transfer tasks
(programs with all method calls but no programming constructs – add missing constructs to complete)

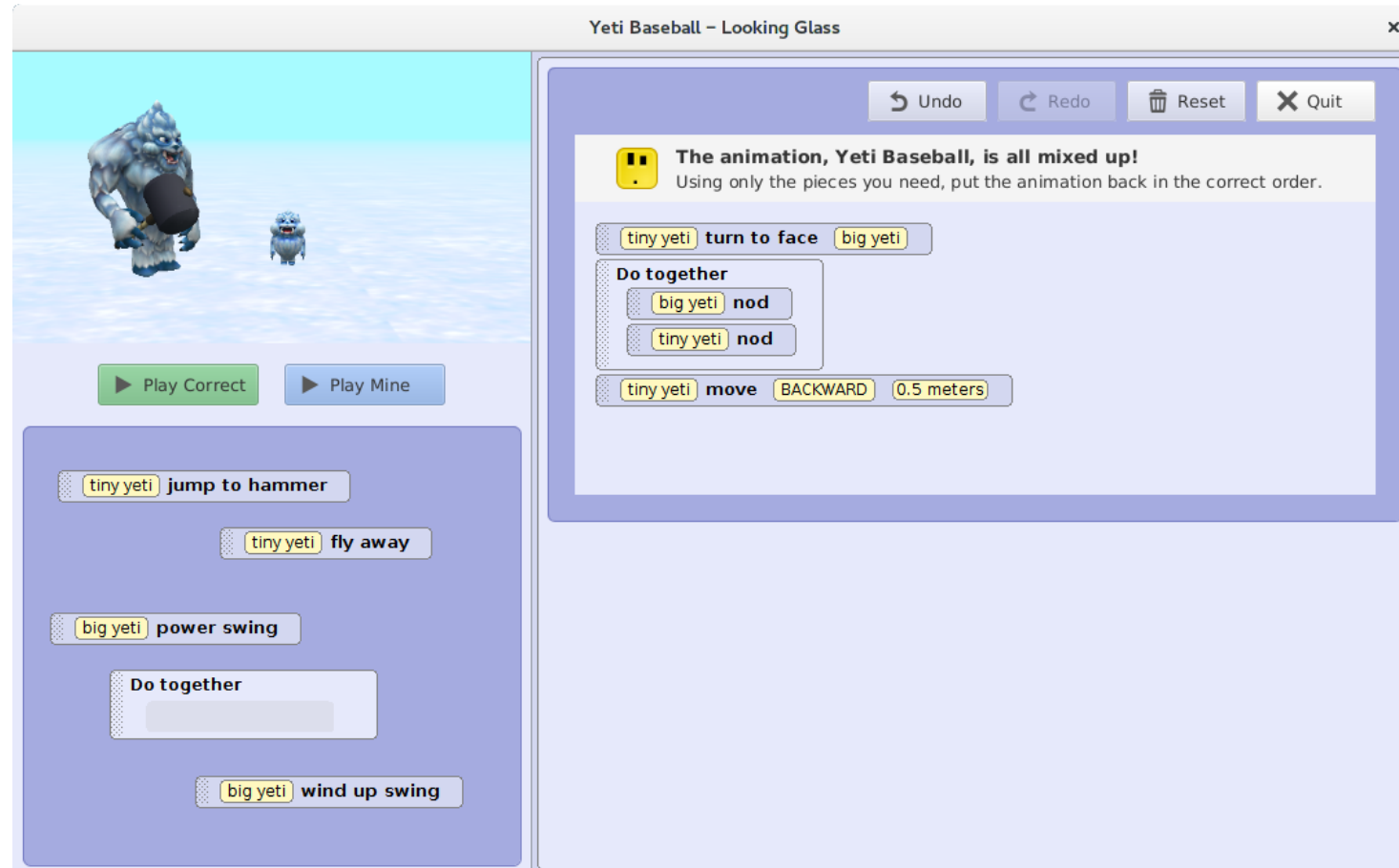
Highlight Results

- Puzzle users perform 26% better on transfer tasks.
- Puzzle users complete training 23% percent more quickly.
- More evidence of mental engagement using puzzles than tutorials.

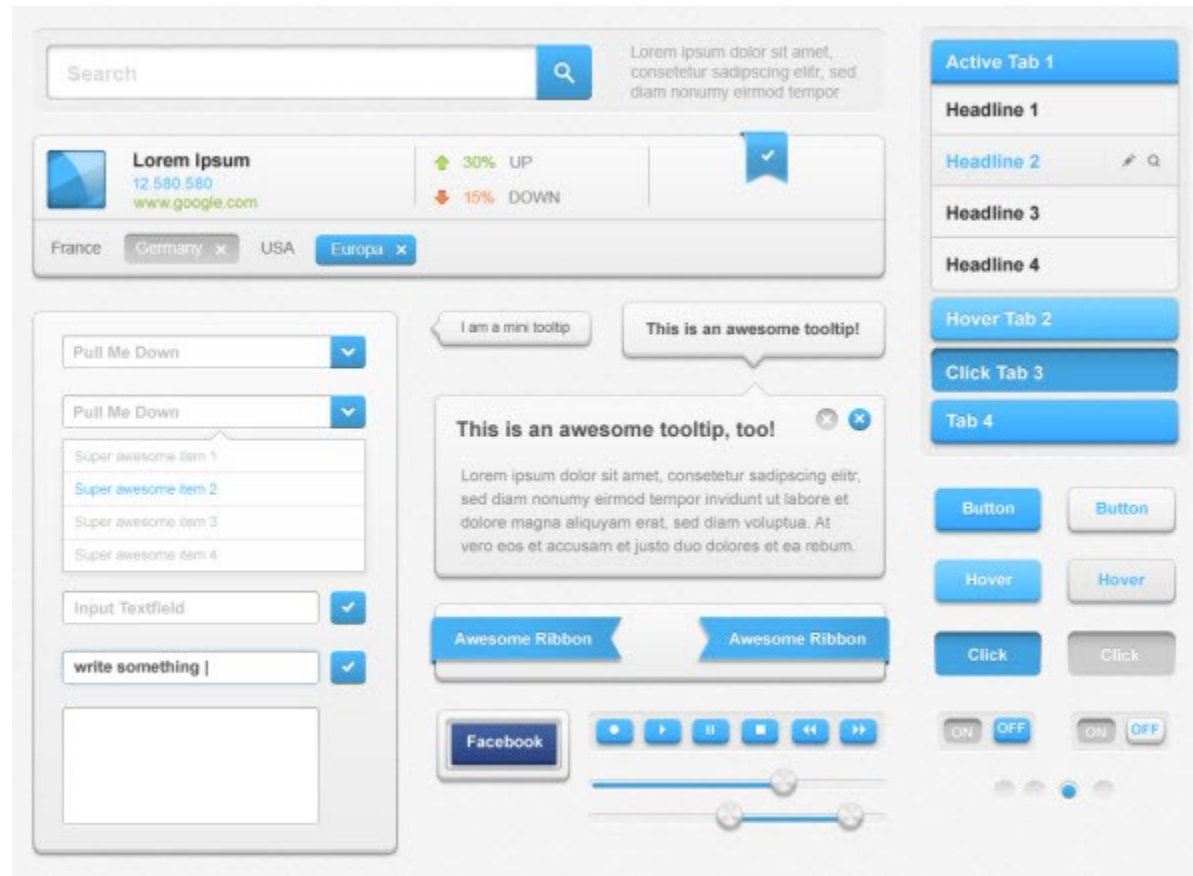
Your First Idea...



The Result... Demo

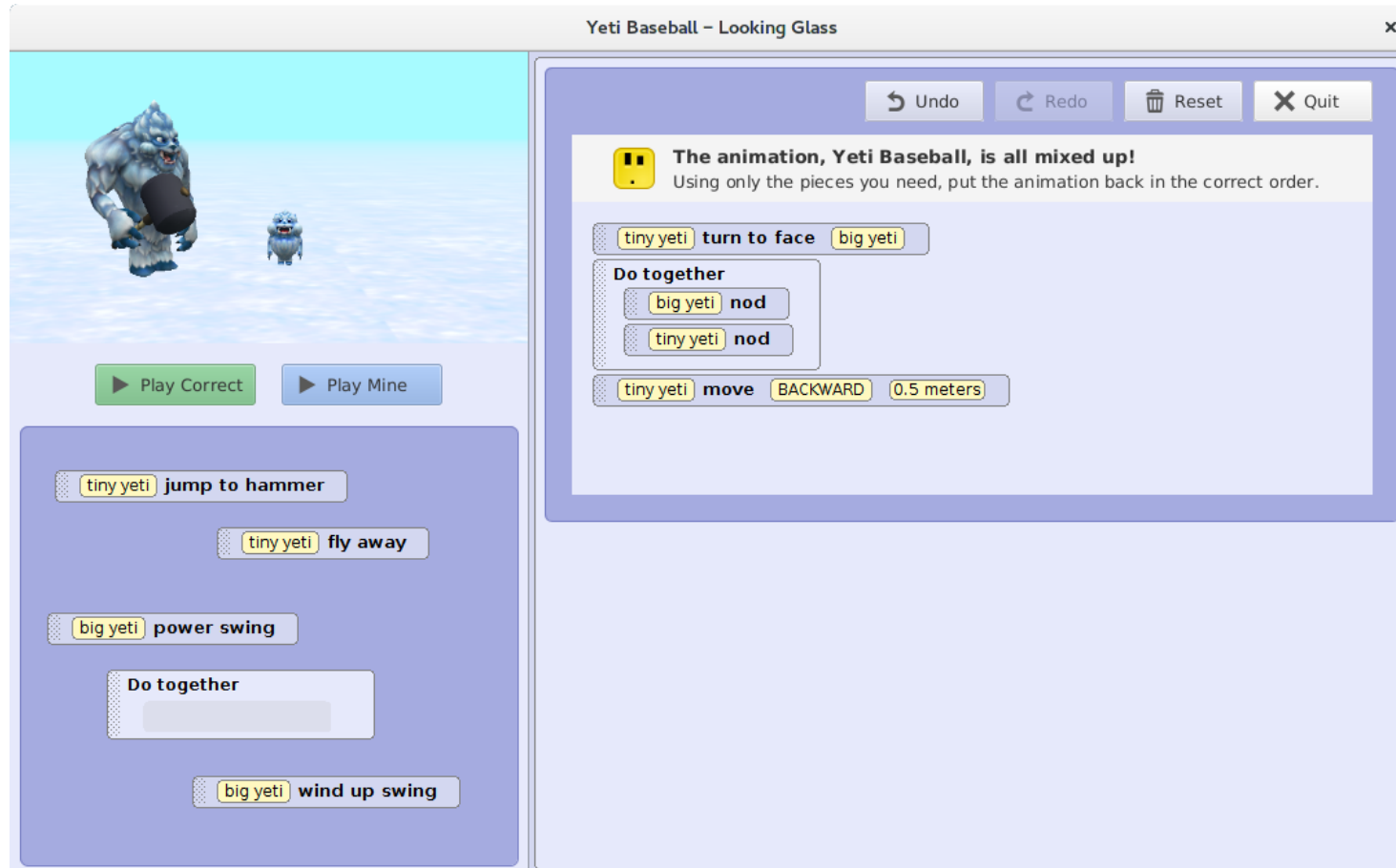


HCI: Not Just About GUIs

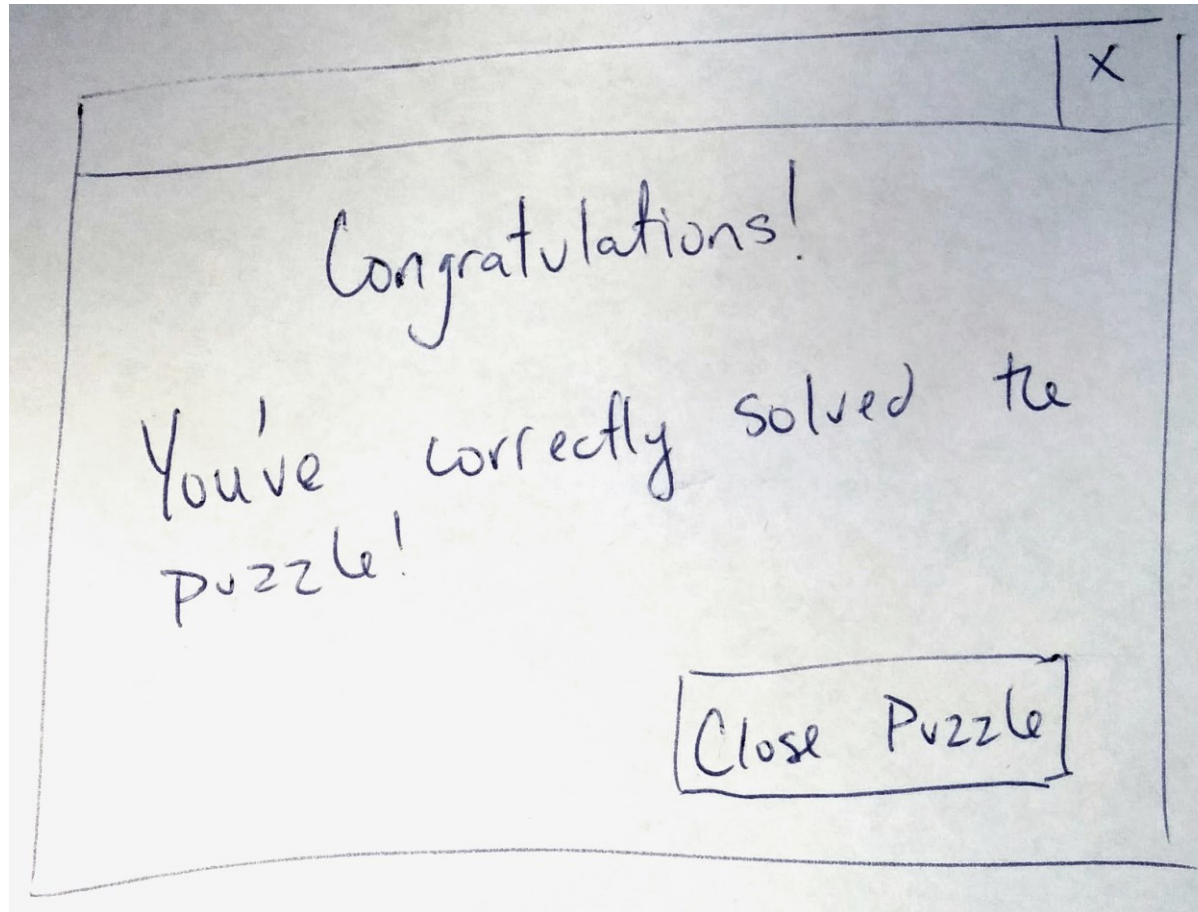


<http://www.webdesigndev.com/20-best-free-gui-sets-use-projects/>

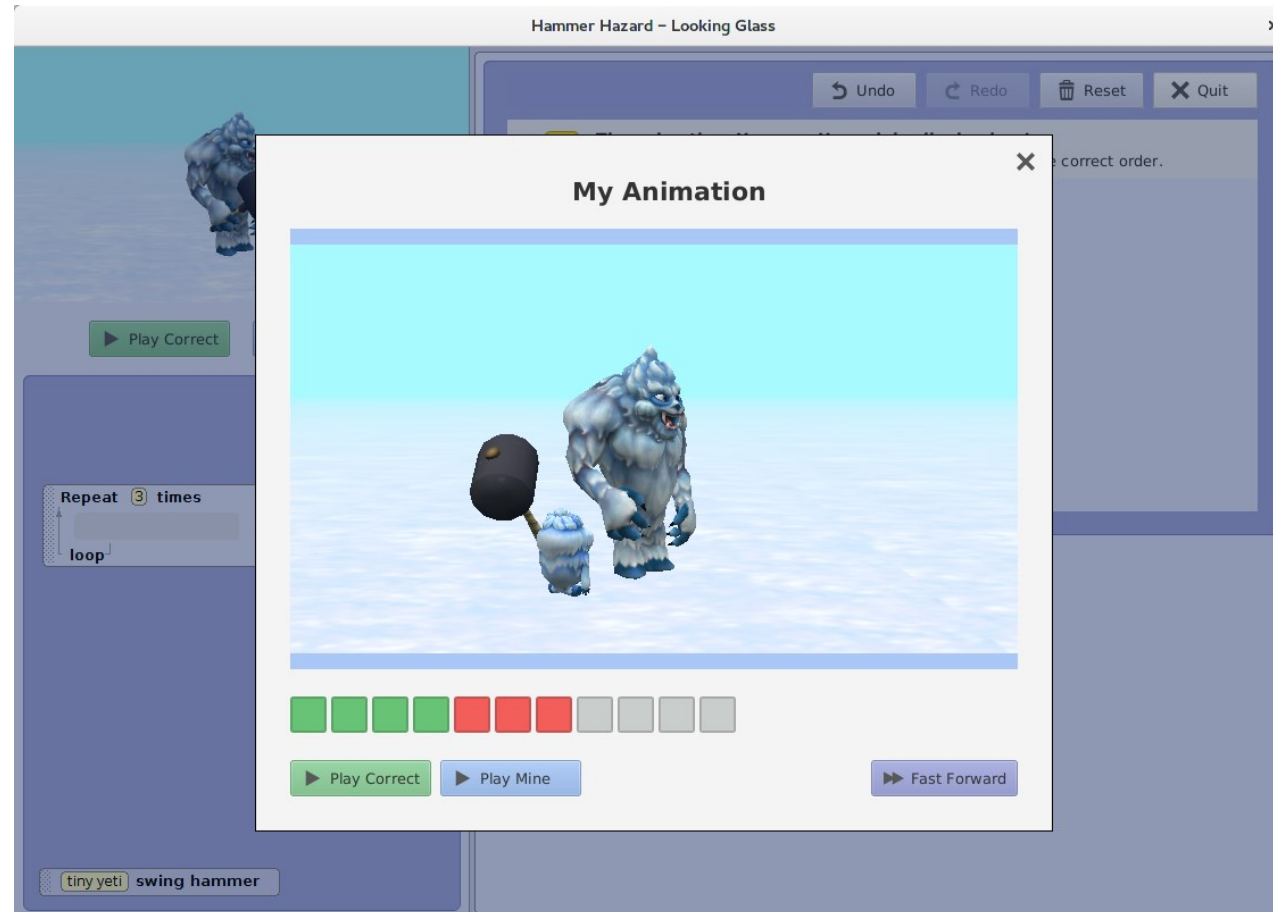
Algorithm to check if puzzle is correct



You didn't solve it... um... help?



It's not just about interfaces...



APIs – Scale an Image

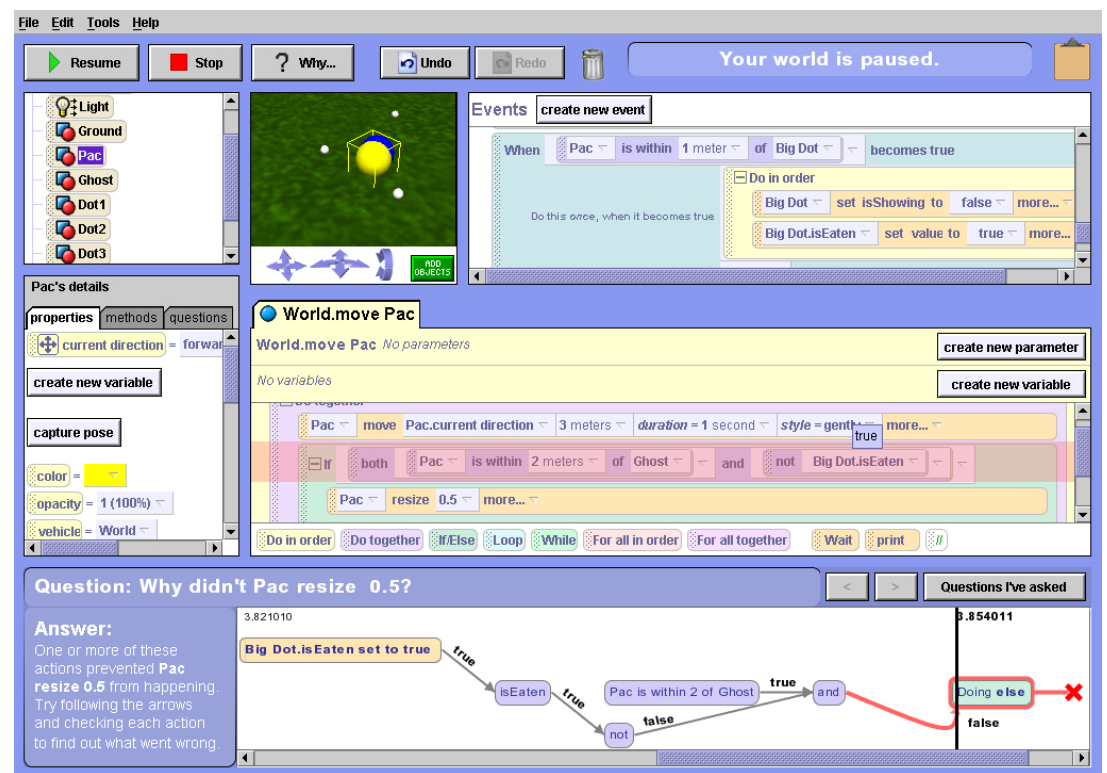
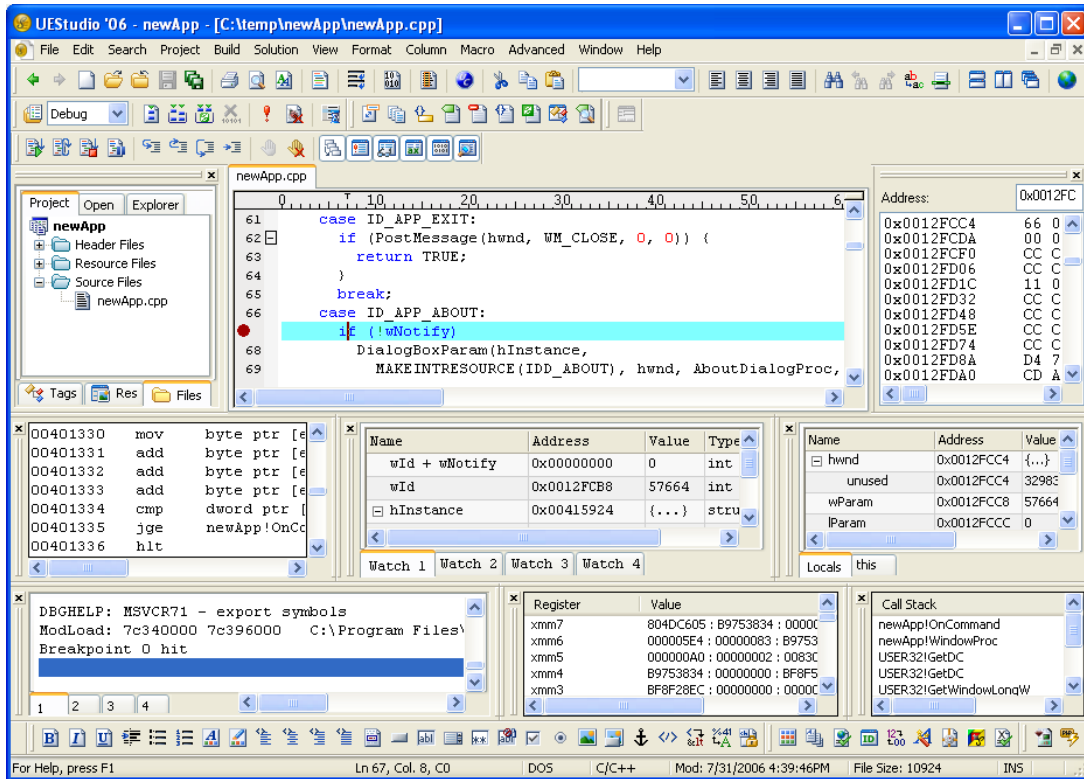
```
private float xScaleFactor, yScaleFactor = ...;
private BufferedImage originalImage = ...;
public void paintComponent(Graphics g) {
    Graphics2D g2 = (Graphics2D)g;
    int newW =
(int)(originalImage.getWidth() * xScaleFactor);
    int newH =
(int)(originalImage.getHeight() * yScaleFactor);
    g2.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
RenderingHints.VALUE_INTERPOLATION_BILINEAR);

    g2.drawImage(originalImage, 0, 0,
newW, newH, null);
}
```

```
from PIL import Image

i = Image.open("/tmp/c.jpg")
i.thumbnail([220, 133], Image.ANTIALIAS)
i.save('/tmp/c-thumb.jpg', quality=90)
```

Who are we helping?
The computer or the human?



Whyline

The screenshot displays the Whyline software interface, which is used for debugging and understanding the execution of a program. The interface is divided into several panels:

- Top Panel:** Contains buttons for "Resume", "Stop", "Why...", "Undo", "Redo", and a status bar indicating "Your world is paused."
- Left Panel:** A list of objects in the simulation, including "Light", "Ground", "Pac", "Ghost", "Dot1", "Dot2", and "Dot3". Below this is the "Pac's details" section, which includes tabs for "properties", "methods", and "questions". The "properties" tab shows attributes like "current direction = forward", "color", "opacity", and "vehicle".
- Center Panel:** A 3D visualization of the Pac-Man simulation, showing a yellow Pac-Man character moving on a green field with white dots and a ghost.
- Right Panel:** The "Events" section, which shows a sequence of events and actions. The first event is "When Pac is within 1 meter of Big Dot becomes true", followed by a "Do in order" block containing "Big Dot set isShowing to false" and "Big Dot.isEaten set value to true".
- Bottom Panel:** A "Question: Why didn't Pac resize 0.5?" section. It provides an "Answer:" explaining that one or more actions prevented the resize. Below the answer is a "Question: Why didn't Pac resize 0.5?" section, which shows a trace of the execution. The trace starts with "Big Dot.isEaten set to true" (true), followed by "isEaten" (true), "Pac is within 2 of Ghost" (true), and "and" (true). The trace then shows "Doing else" (false) and "false".

<https://www.cs.cmu.edu/~NatProg/movies/WhylineH264.mov>

Administrivia

About Me

- Kyle Harms
- 6 year PhD Student in Computer Science (AbD)
- Worked at Boeing as a Software Engineer for 5 years
- Contact
 - kyle.harms@wustl.edu
 - Checked Monday-Friday 8am-5pm.
- Office Hours
 - By Appointment
- Feedback Welcome

You?

- Share a bit about yourself
 - Your Name
 - What you do? Are you a full-time or part-time student?

Course Website

- <http://research.engineering.wustl.edu/~harmsk/teaching/cse-556a/>
 - Syllabus
 - Calendar
 - Required Reading
 - Bring Materials to Class
 - Project Details

In this class...

- How to get actionable and accurate information by interacting with potential users of a system.
- How to use a variety of techniques to analyze and improve an interface without user involvement.

A Word About Laptops/Tablets/Phones

- Research suggests that there is a negative correlation with in-class laptop usage and grades.
- Students who use laptops in class report lower levels of engagement and learning.
- Laptop usage significantly increases distractions for other students.
- “But I use my laptop to take notes...”
 - 83.3% take notes, 81% send email, 68% send instant messages, 43% surf the internet, 25% play games.
 - If you are going to use a laptop, sit in the back.

Reading Summaries (15%)

- Introduce a new method, technique etc.
- Practice using it and discuss pitfalls in class.
- Summaries are due by noon the day of class.
- 1-2 paragraphs
 - Did you read the material?
 - Do you have any misconceptions about the material?
- Questions are required.

In-Class Designs (5%)

- Scenarios that provide an opportunity to practice a new technique.
- Devil is definitely in the details, so we need to be able to get there.

Group Design Project (65%)

- Semester-long project that will enable you to employ user-centered design practices from the requirements stage through a running digital prototype.

Exam (15%)

- July 26, 2016

Project Pitches

- <http://research.engineering.wustl.edu/~harmsk/teaching/cse-556a/project/#pitch>
- 2-3 minute “pitch” in class Thursday.
- Flyer and slides due by noon Thursday (June 16).

Project Domain

- A totally new project or add to an existing project
- Often people choose to pitch a project based on an interest.
 - Crew team member pitching a training app for other rowers(?)
- An observed problem
 - Hospital volunteer pitching a tool to help patients feel more connected to their families.
- Or a frustration
 - None of the personal finance management software really works for me.

Caution

- If you start with a personal motivation (totally fine), make sure that you demonstrate that there is a substantial group that has a similar problem and will benefit

Project Scoping



Don't take on too much.

Can you imagine someone working full time building a working prototype in about 2 weeks?

That's about the right scope.

Focus Implementation on the UI

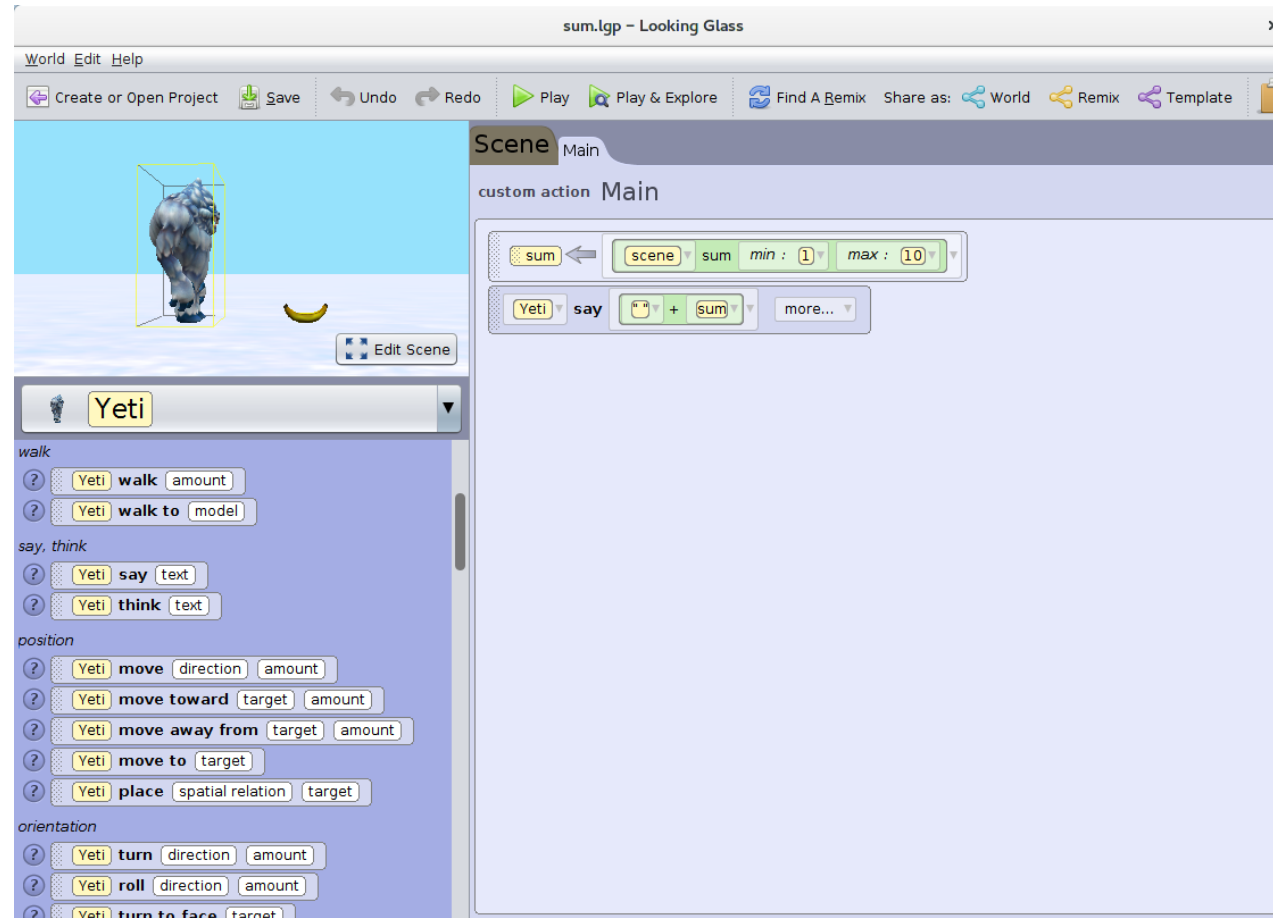
- This class is mostly about how to design and test effective user interfaces.
- Super clever or involved backend implementations take away from that focus. Feel free to leverage open source projects and other code you can find.

Example Pitch

An Interface for Asynchronous Document Synchronization in Looking Glass

Kyle J. Harms

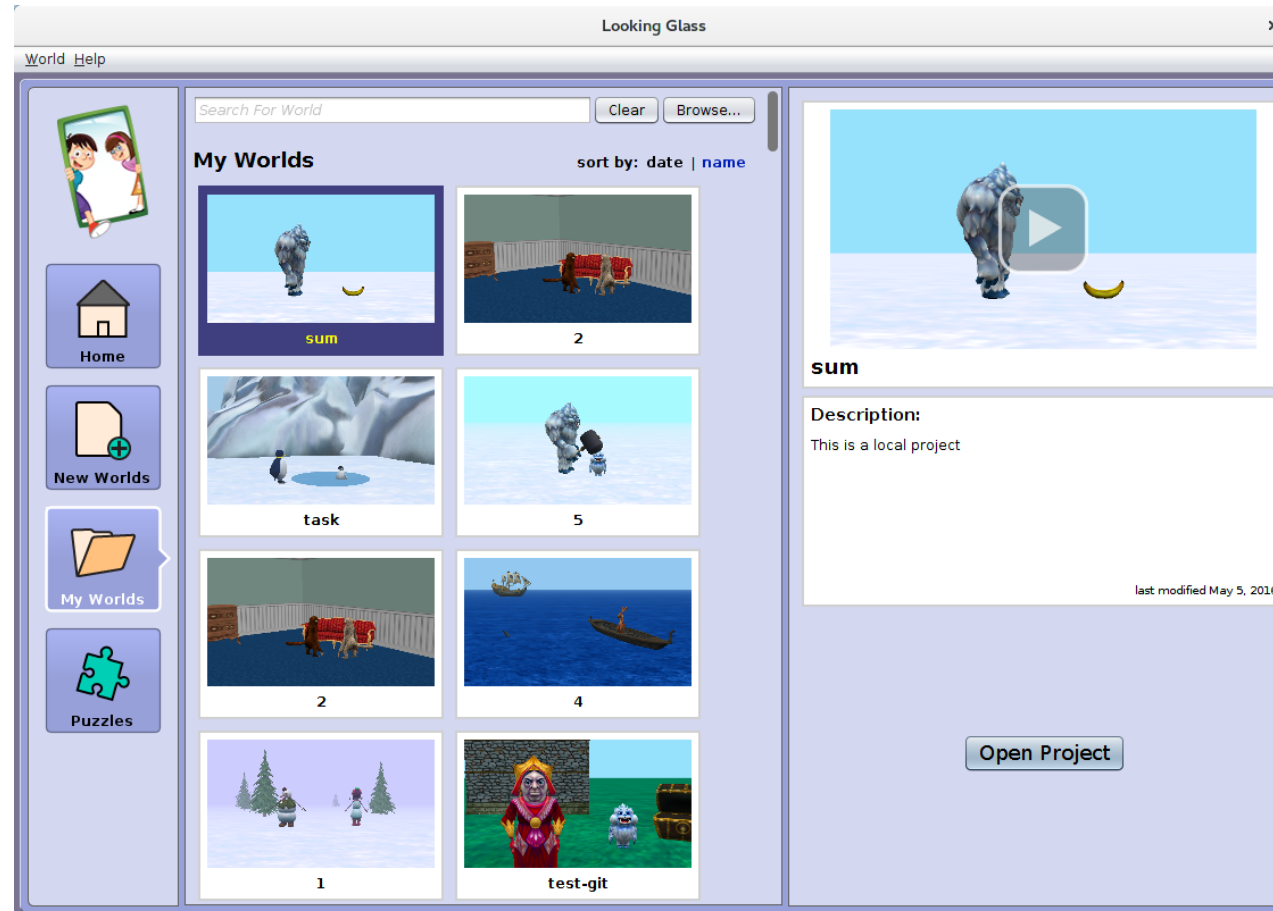
Looking Glass



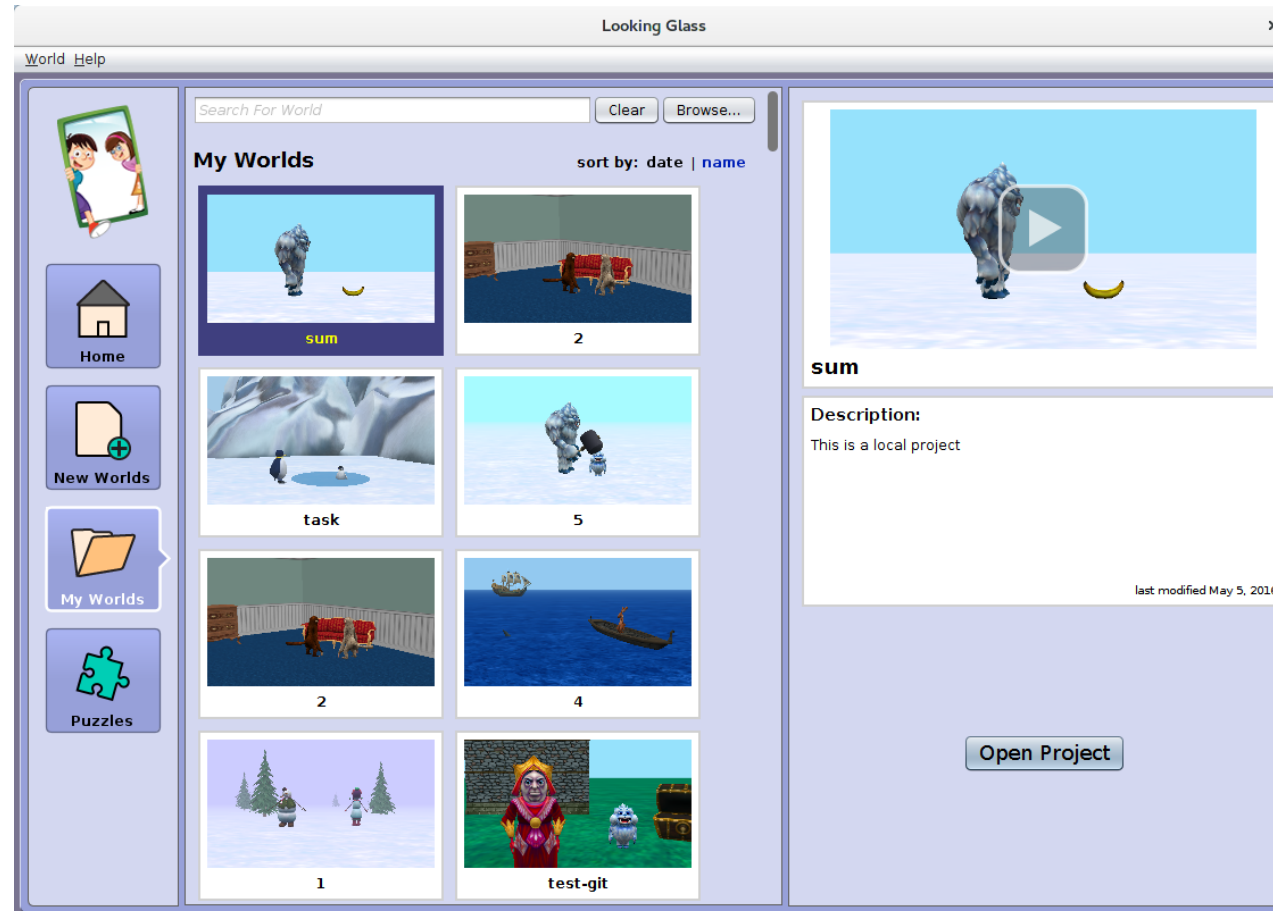
The Problem

- Population
 - Middle School Kids
- Problem
 - Projects are local to the computer you created them on.

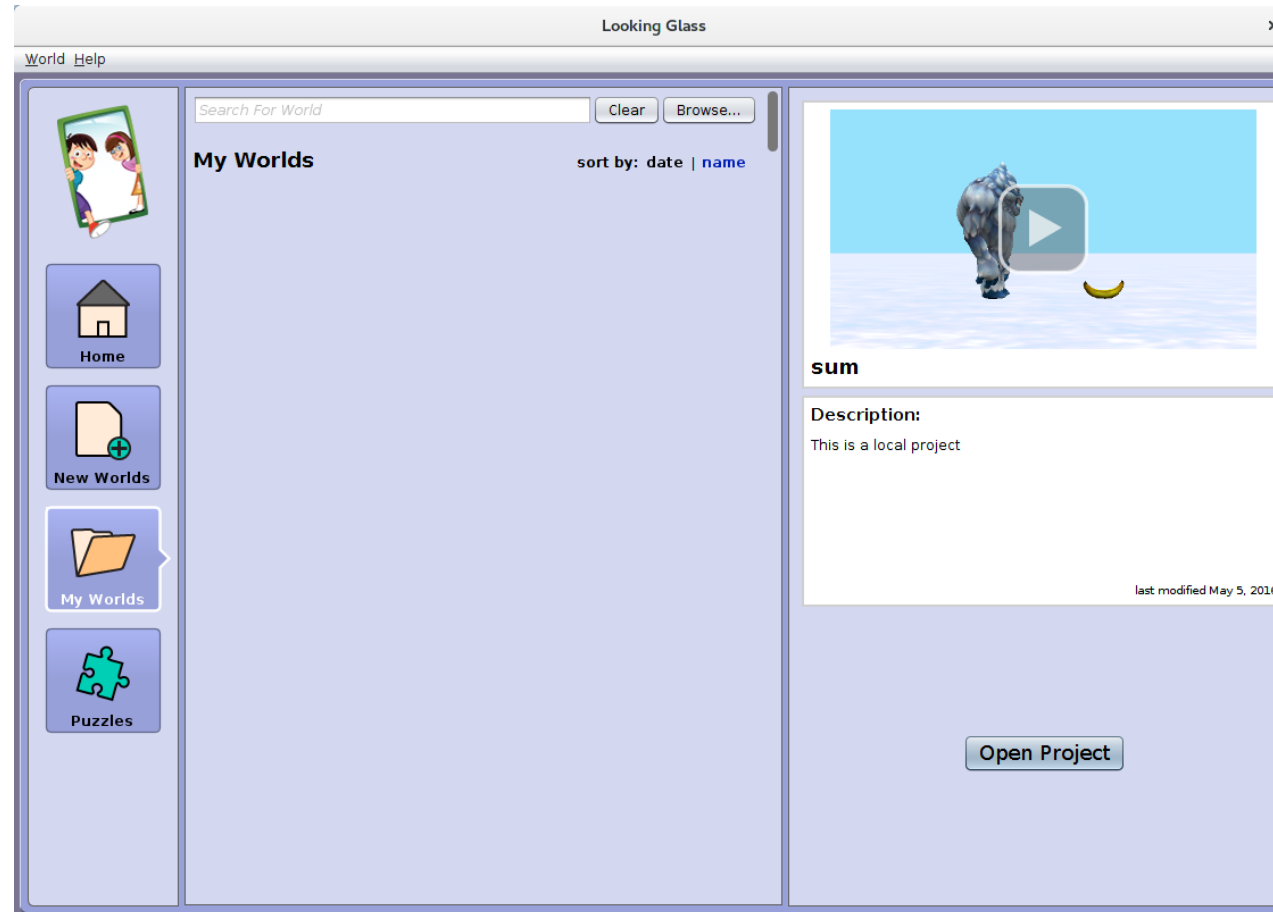
Your Projects/Programs/Files



Computer 1: All my projects



Computer 2: Where are all my projects?



Proposal

- Develop an interface for handling file synchronization in Looking Glass
- Prototype
 - Java source code is too difficult to rapidly prototype in test
 - Develop prototype with HTML & Javascript (Bootstrap?)
 - Fake loading projects by showing screenshot of program

Feasibility

- Middle School Kids are out for summer and I'm a Scout Leader
 - (Not really 😊)
- Development with HTML & Javascript is quick and free.
- Mostly UI work
 - No backend needed, we can fake synchronization.

Advice

Advice

- Read the Project Pitch on the website.
- Practice your pitch.
 - Do not go over 3 minutes.
 - Always number your slides
- Be concise...
 - What is the problem?
 - Who is the target audience?
 - Is the scope appropriate for this class?
- Send me your pitch and slides as PDFs before noon.