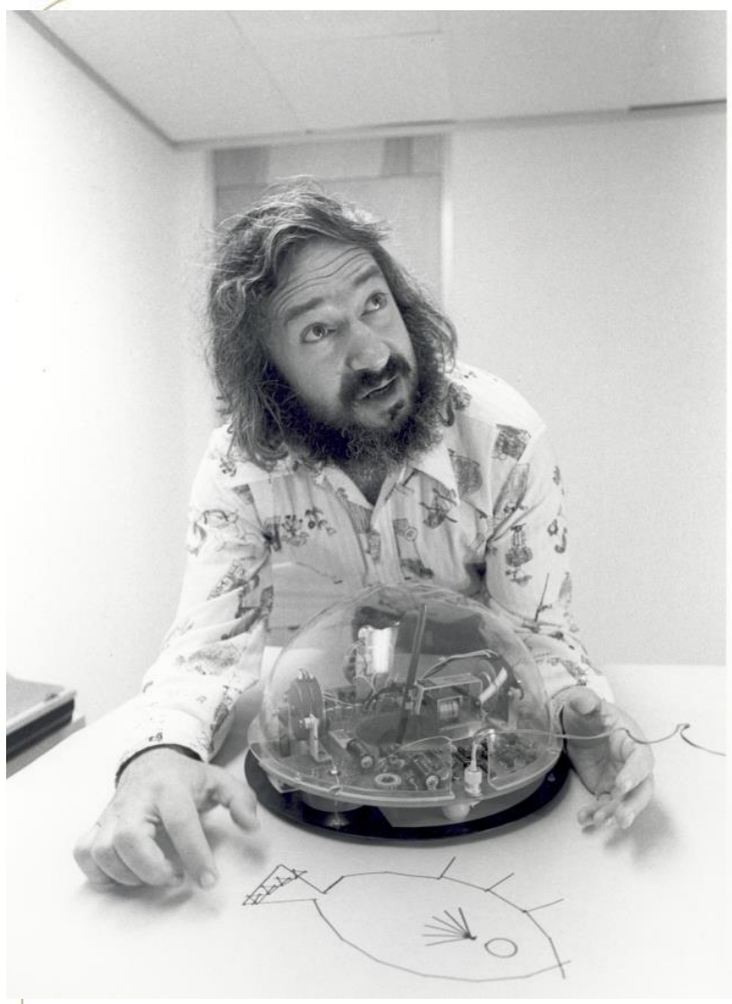
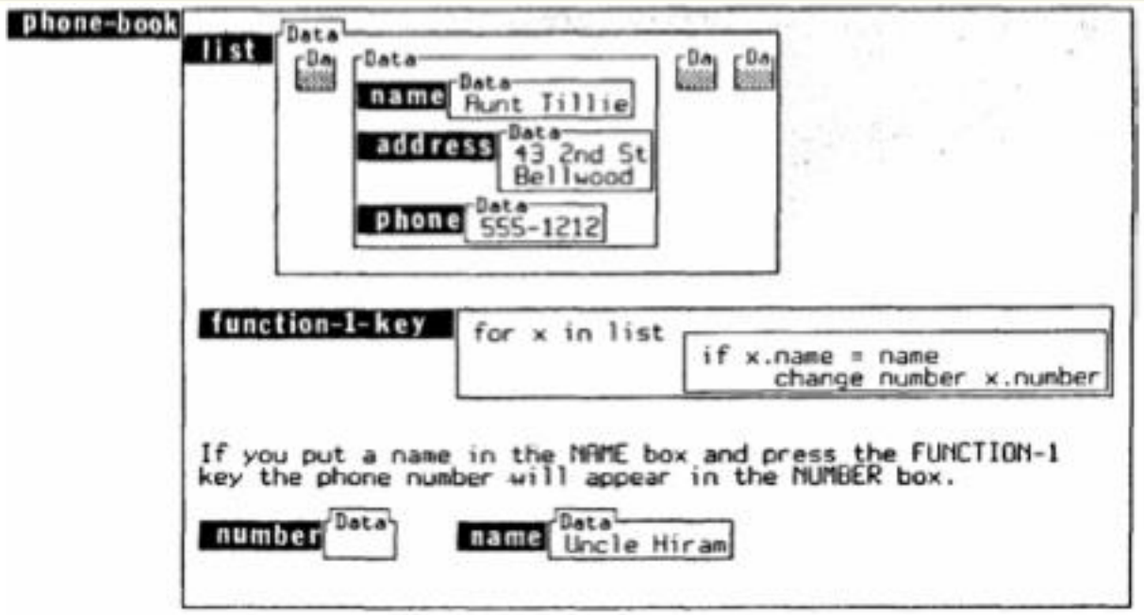
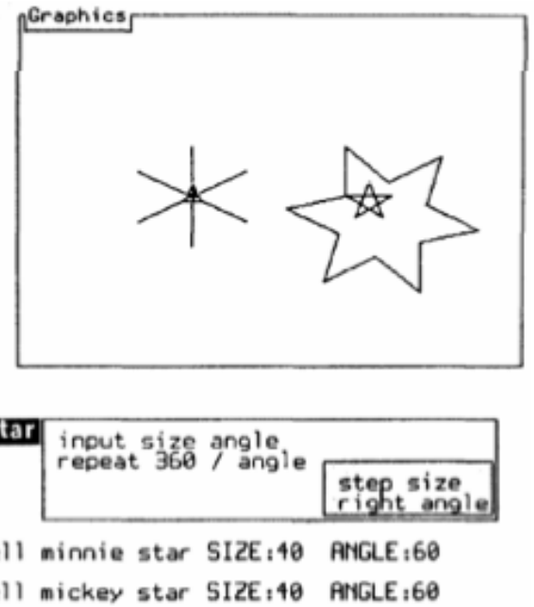
Guzdial Research

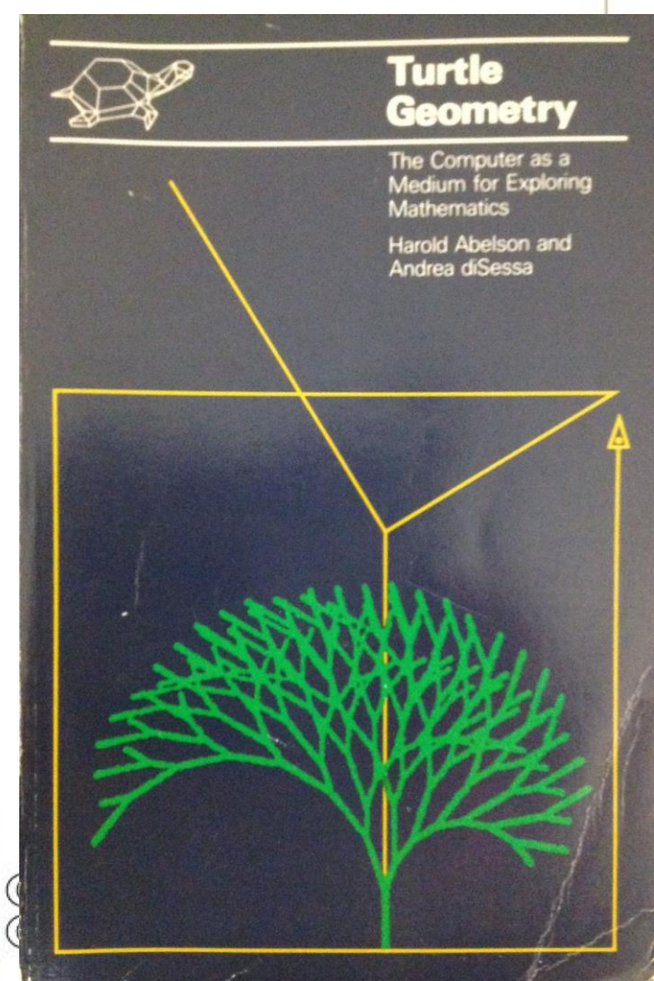
- **How do we support the use of programming as a medium for thought and communication?** How do we improve success in programming (especially for people who aren't computer scientists)?
- Past Research:
  - Media Computation
  - Subgoal Labelling of Worked Examples
- Current Research Questions:
  - How do we make programming a more successful medium for learning in non-CS subjects?
  - How do students learn programming?

Papert, Logo; DiSessa, Boxer; Kay, Smalltalk



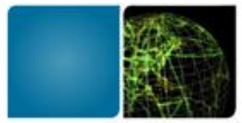






Georgia Tech

Media Computation

- *Fall 1999*:
  All students at Georgia Tech must take a course in computer science.
  - Considered part of General Education, like mathematics, social science, humanities…

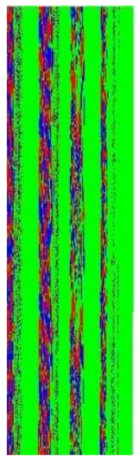
- 1999-2003: Only one course met the requirement.

- Overall pass rate: About 78%
  - Around 50% for Liberal Arts, Architecture/Design, and Business students

Media Computation: Teaching in a Relevant Context

- Presenting CS topics with media projects and examples
  - Iteration as creating negative and grayscale images
  - Indexing in a range as removing redeye
  - Algorithms for blending both images and sounds
  - Linked lists as song fragments woven to make music
  - Information encodings as sound visualizations



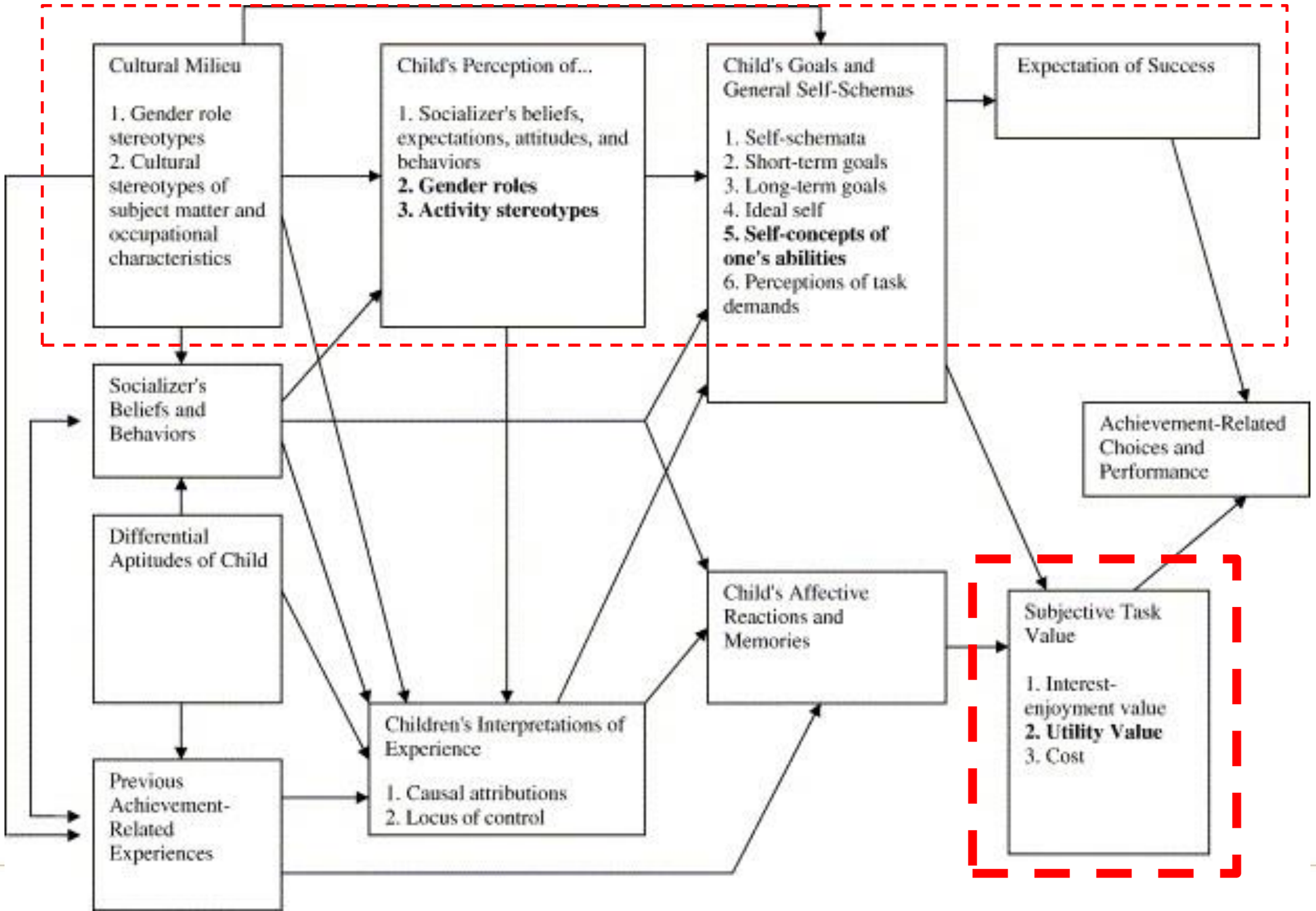



Results: Media Computation

Change in Success rates in CS1 "Media Computation" from Spring 2003 to Fall 2005 (Overall 85%)		
Architecture	46.7%	85.7%
Biology	64.4%	90.4%
Economics	54.5%	92.0%
History	46.5%	67.6%
Management	48.5%	87.8%
Public Policy	47.9%	85.4%

Similar results replicated at multiple institutions

Eccles (1983) model of achievement-related choices



Part 2: Subgoal Labeling

- Students are often overwhelmed when programming.

  "You've taught me so many details,

  I don't know which ones to use."

  (Clancy & Linn, 1990)

- How do we convey how to think about the *purpose* for the parts of the program? About *why* each part is there?

- Richard Catrambone (1994) invented a way to label the *subgoals* in examples provided to students.
  - Developed in statistics, chemistry, and physics
  - Never tested before in Computer Science

Example of Written Materials

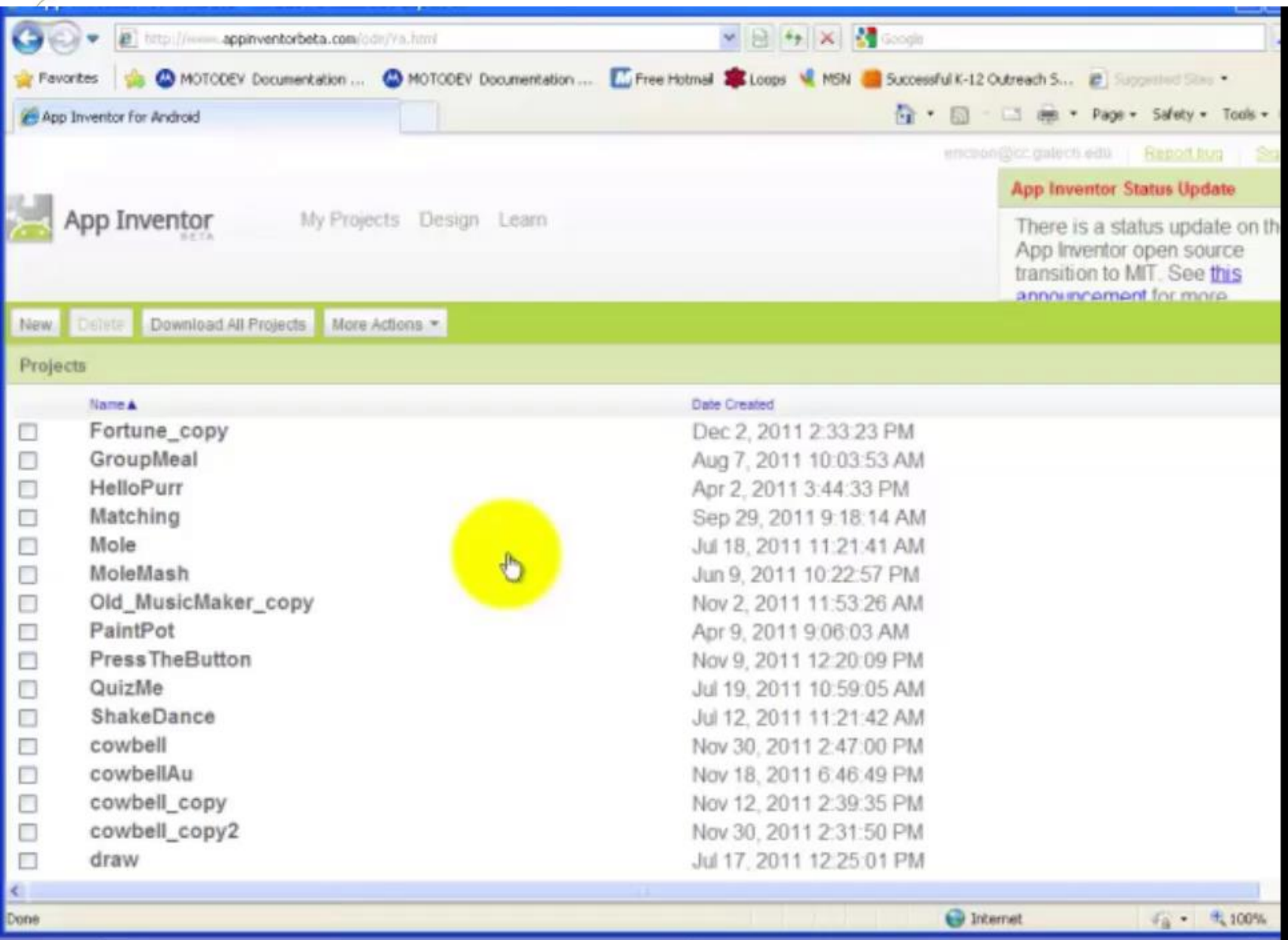
Subgoal

- Define Variables from Built-in
- Click on "Built-In" and "Definition" and pull out a def variable.
- Click on the "variable" and replace it with "fortuneList". This creates a variable called "fortuneList".
- Click on "Lists" and drag out a call make a list
- Click on "Text" and drag out a text text block and drop it next to "item". Click on the rightmost "text" and replace it with your first fortune.
- Handle Events from My Blocks
- Click on "My Blocks" and "Button1".
- Drag out a when Button1.Click.

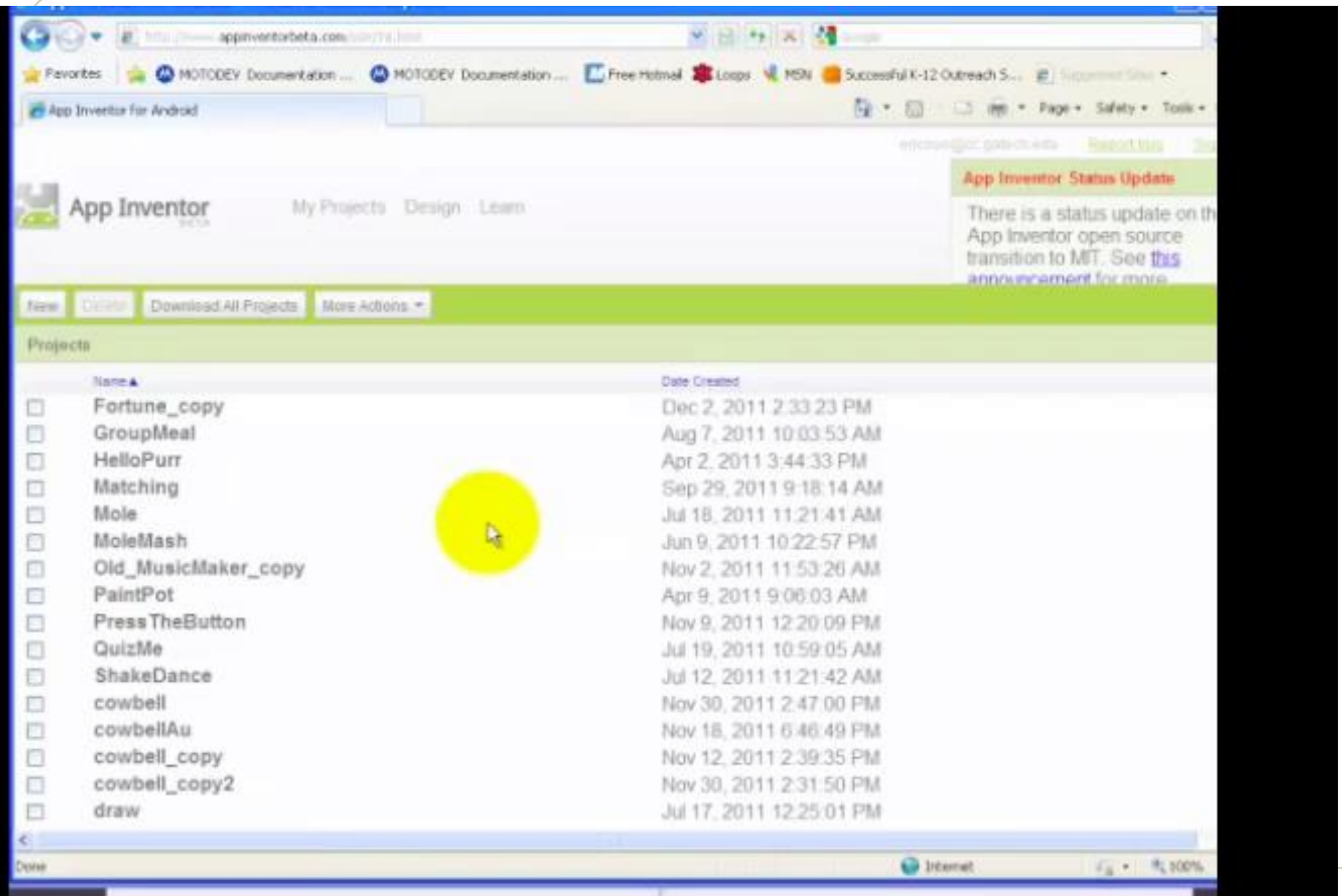
Non-subgoal

- Click on "Built-In" and "Definition" and pull out a def variable.
- Click on the "variable" and replace it with "fortuneList". This creates a variable called "fortuneList".
- Click on "Lists" and drag out a call make a list
- Click on "Text" and drag out a text text block and drop it next to "item".
- Click on the rightmost "text" and replace it with your first fortune.
- Click on "My Blocks" and "Button1".
- Drag out a when Button1.Click.

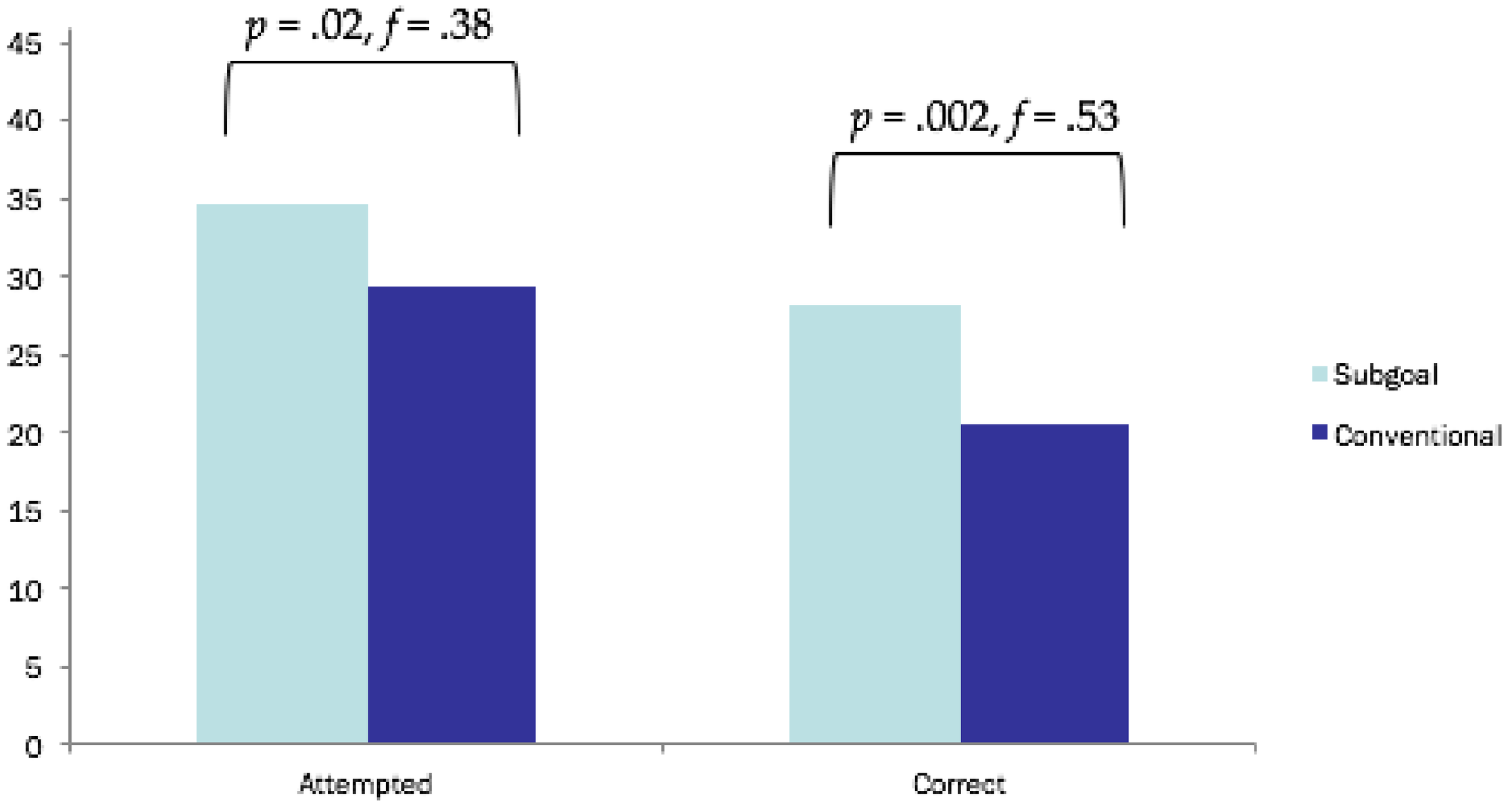
Original Video



Video with Subgoals



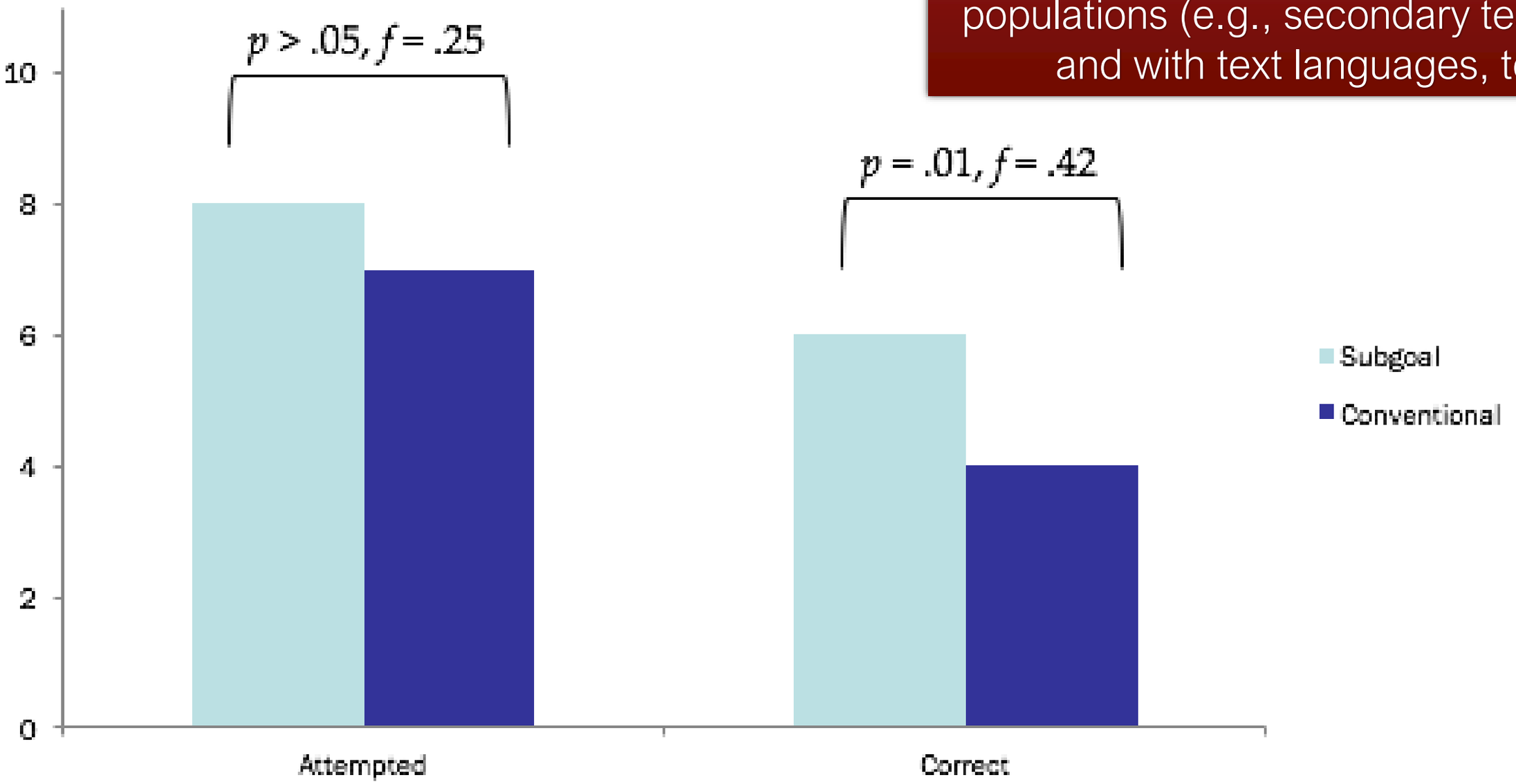
Results: Understanding



Results: Retention

Significant results for transfer as well.

Effect has been found with different populations (e.g., secondary teachers), and with text languages, too.





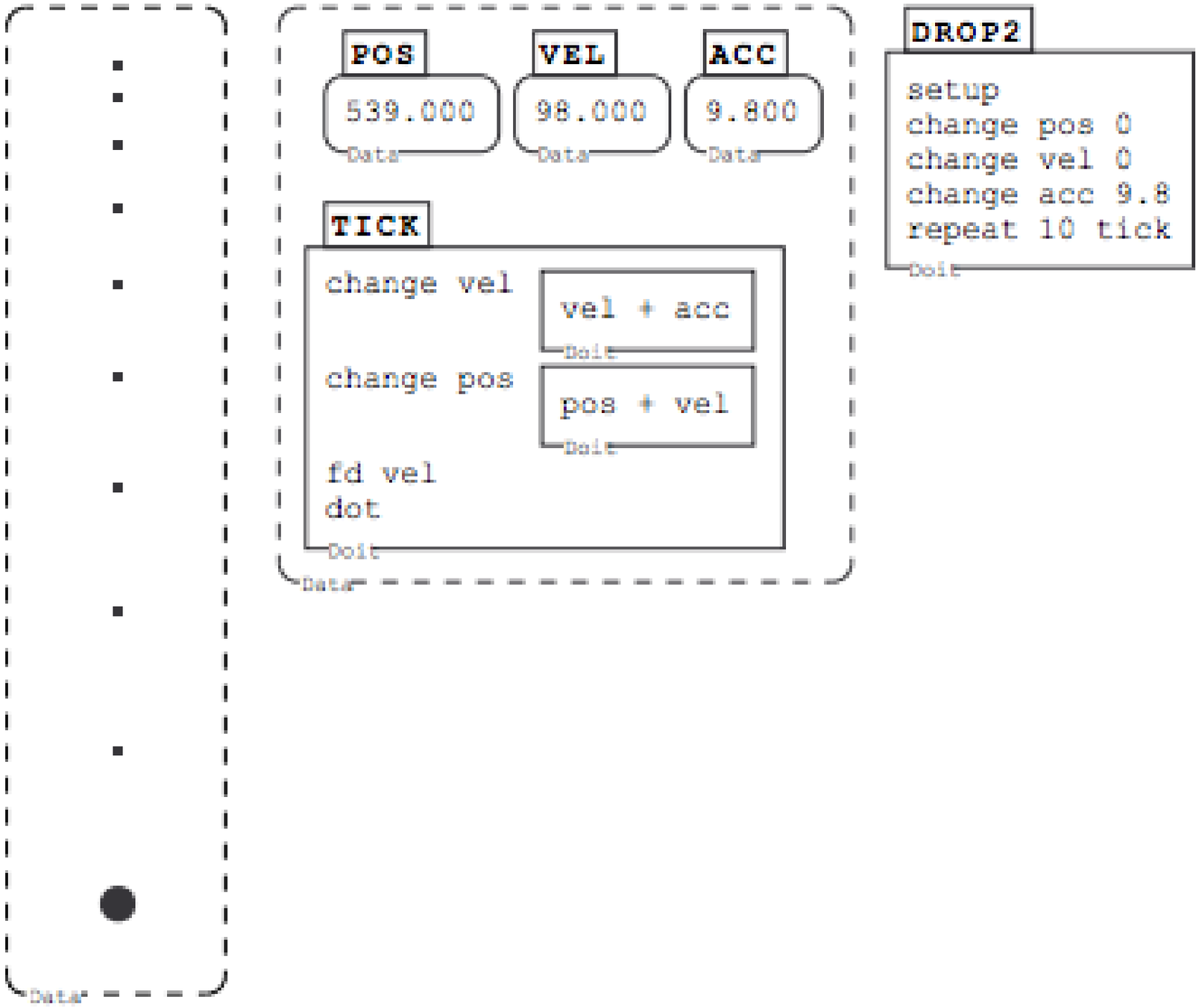
Why do subgoal labels work?

- Ashok Goel's **Structure-Behavior-Function (SBF)** model of design knowledge
  - **Structure** is code
  - **Function** is the purpose for the code, what goal it achieves.
  - **Behavior** is how the code dynamically achieves the goal
  - *Behavior is hardest for students*
- Subgoal labels, across different programs, connect S, B, and F.

Current Research Questions

- How do we make programming a successful medium for learning in non-CS subjects?
  - Code has value as a learning medium
  - Targeting Pre-Calculus and Economics
- How do students learn programming?

Code is Different



Bruce Sherin

Code is Temporal & Causal

Researcher: Can you tell me how long it took the rock to get to the ground?

Student B: It would be about one second

R: Okay, where did you get that from?

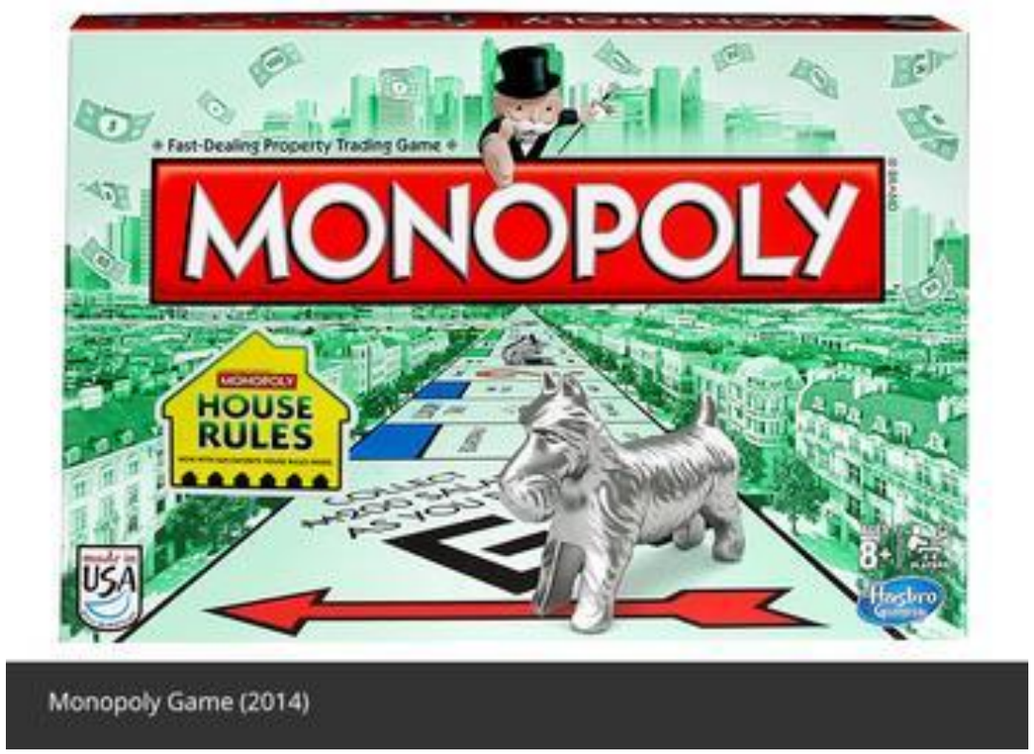
B: If the acceleration is 30 feet per second per second, then per second it will be going 30 feet per second, then it will just take a little longer for it to get to the ground.

R: Why?

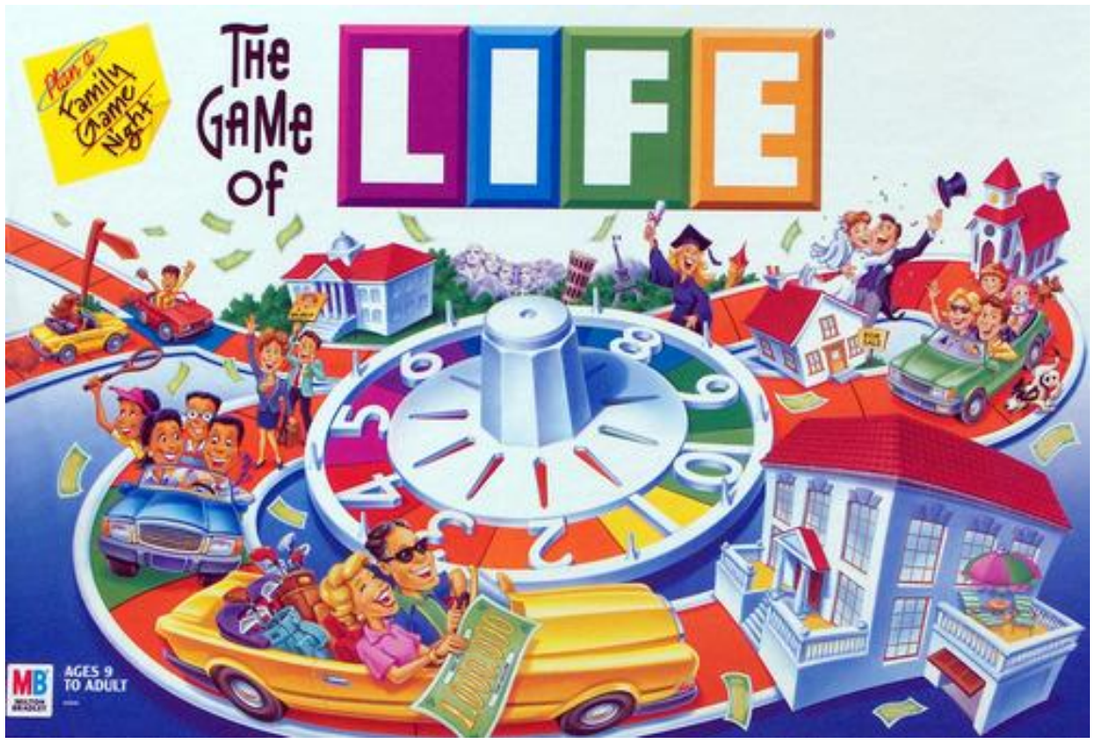
B: Because you have to divide the, to get the average velocity, which is how fast it's going, and how you can measure how far it's gone, you have to...let's see...it will be going, it will be going 15 meters per second. Maybe two seconds, I guess.

R: Why?

B: Because...1.5 seconds. Because, by the time it's accelerated the second second, it will be going about 45 feet per second, so it'll have to be between the first and second second that it hits the ground.



Monopoly Game (2014)



Teaching economics and systems-thinking.

First experiences with economic systems

Interleaving for Thinking about Systems

What do we want from an economic system?

Play the Board Game

Define what you mean by “win.”
Can you come up with the perfect winning algorithm?

Scale the Basic Board Game

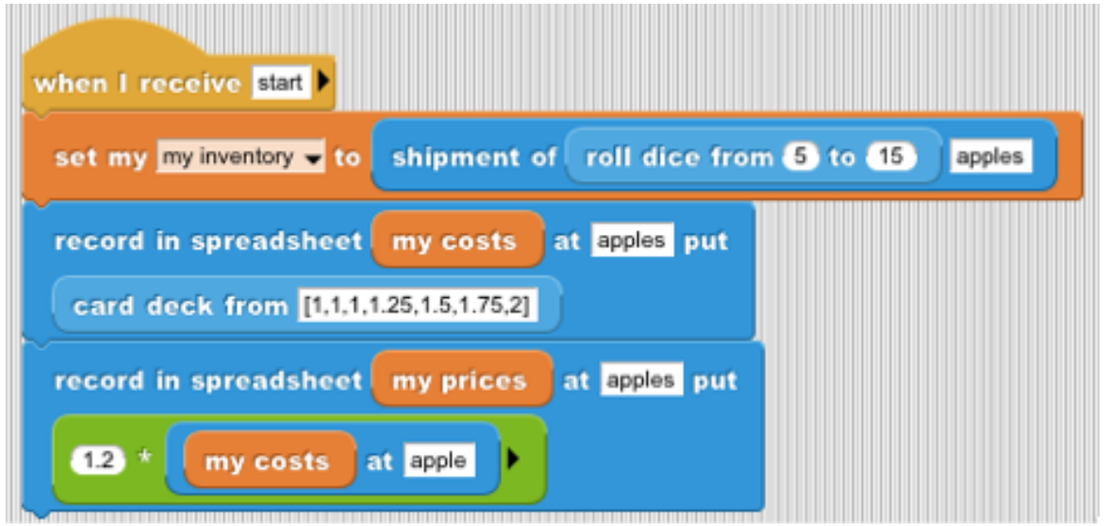
Put your algorithm in a game board simulation. Does your algorithm work the same over 1000’s of plays?

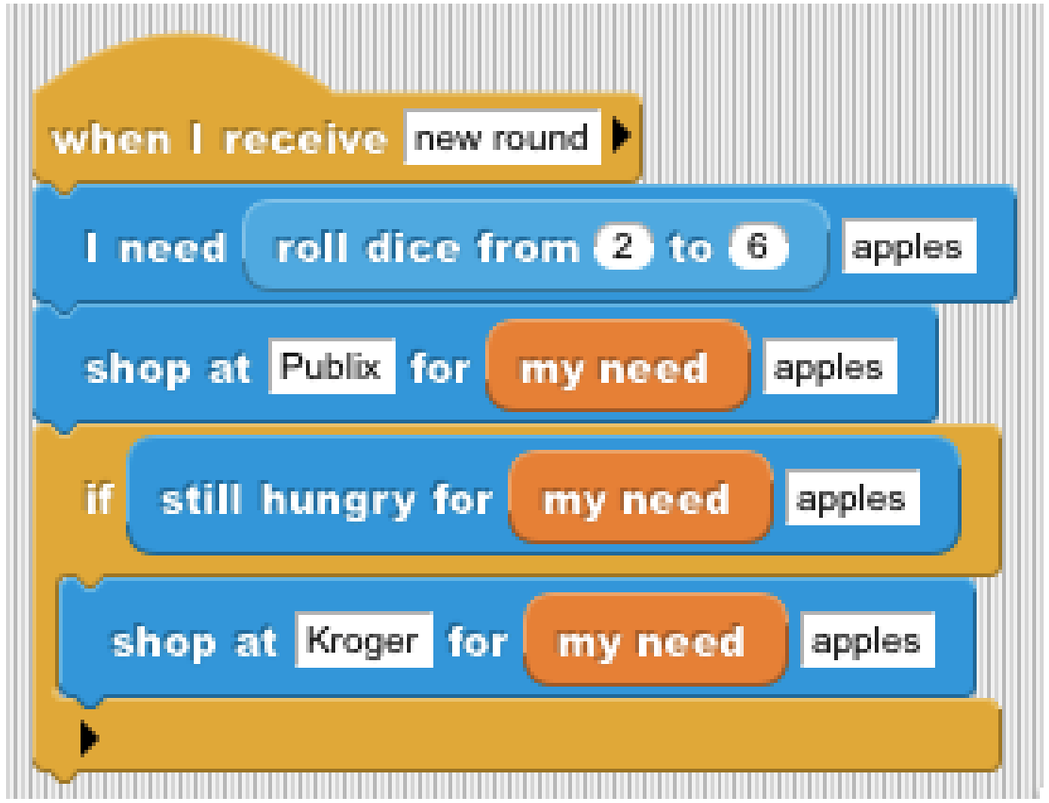
Redesign the Board Game Rules

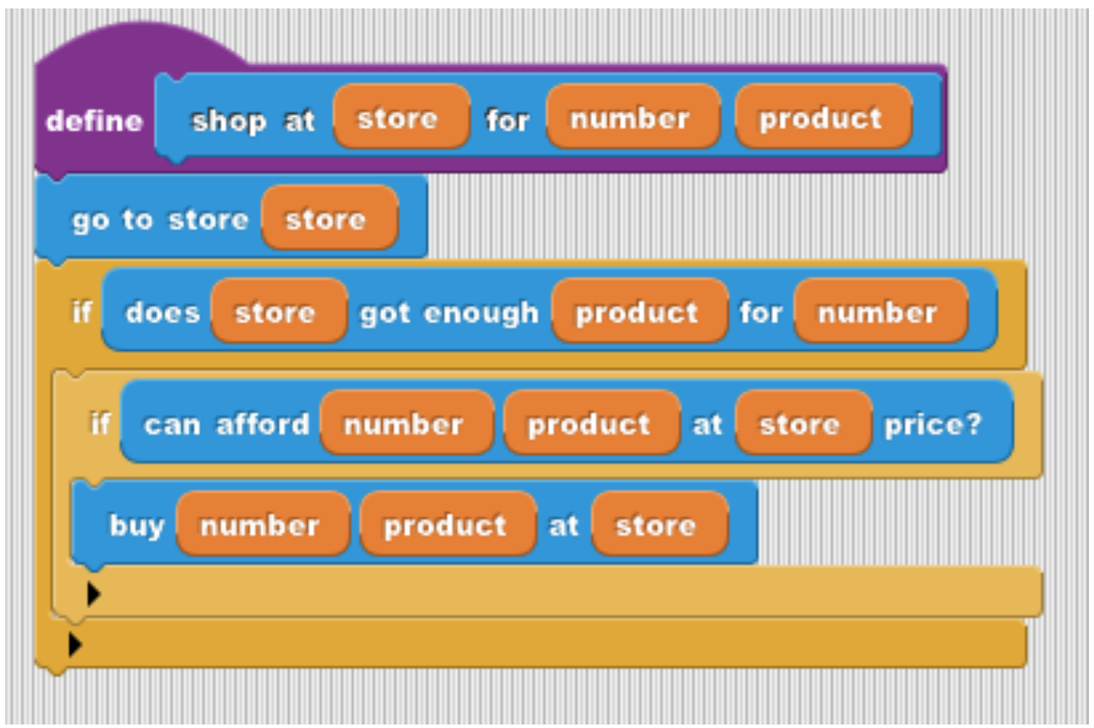
Scale the Redesigned Board Game Rules

What do we want from an economic system?

Discrete Event Simulations, in Blocks Language







Using Theory to Explain Learning Programming

- **Teach the syntax and semantics of languages**
  - Needs context (for motivation and function/goals) and concrete worked examples to support later abstraction.
- **Constructionism**: Little teaching, but let students play and learn from rapid feedback.
  - Inefficient. Requires high self-efficacy.
- **Functional**: Create analogies to learning mathematics.
  - Requires prior knowledge of mathematics and high self-efficacy for mathematics

- **Plan-based**: Teach students patterns of code (plans) that achieve something they want to do, and when they learn enough plans, they will learn the abstractions of the syntax and semantics. Structure for function, abstracting to behavior.
  - Developing experiments to test the theory (a) for instructional design and (b) as an explanatory tool for programs “in the wild.”

Extra Slides

Success Rates in CS1 from Fall 1999 to Spring 2002 (Overall: 78%)	
Architecture	46.7%
Biology	64.4%
Economics	53.5%
History	46.5%
Management	48.5%
Public Policy	47.9%

Total Fall01 Females Fall01 Males Fall01 Total Sp02 Females Sp02 Males Sp02 Total Fall02 Females Fall02 Males Fall02

Experiment with App Inventor

- Used subgoal labeling to teaching Android App Inventor (a blocks-based programming environment) to new Computer Science Students.
- Two groups of undergraduate students:
  - One group was shown a video for how to use the software to build an App and given text listing the steps in the instruction.
  - Another group was given the video and the steps with subgoal labels.

Lauren Margelieux, Mark Guzdial, and Richard Catrambone, ICER 2012

Steps in Experiment

Week 1:

- Watch the video.
- Take a test to demonstrate *understanding*.

Week 2:

- Take a test to demonstrate *retention.*
- Watch a new video.
- Take a test to demonstrate understanding of second video.
- Take a test where students must build a new app, *transferring* knowledge.

Results: Define Variable Step in Transfer Task