Lessons Learned as a Student & as a Teacher

J. Eric Ivancich

ivancich@umich.edu

https://github.com/ivancich/Ivancich-Lessons

This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License. See: http://creativecommons.org/licenses/by-sa/3.0/deed.en_US

Overview

- My research & teaching credentials
- A brief (5-10 minute) lecture on a topic suitable for an undergraduate computer science class
- A “brief” summary of my teaching philosophy

Research

Hebbian Networks for Averting the Problem of Catastrophic Interference

Prof. Stephen Kaplan, chair
Prof. John Holland
Prof. John Laird
Prof. Ed Durfee
Prof. Rachel Kaplan, cognate member

Competing Neural Models

Parallel Distributed Processing (PDP)
James L. McClelland and David E. Rumelhart

Hebbian Cell Assemblies and Hebbian Learning
Donald O. Hebb

Model Comparison

	PDP	Hebbian
Architecture	Feed-Forward	Recurrent (cycles)
Training	Supervised	Unsupervised
Learning	Back-Propagation	Hebbian Learning Rule
Representations	Distributed	Distributed

What is Catastrophic Interference

What is Catastrophic Interference

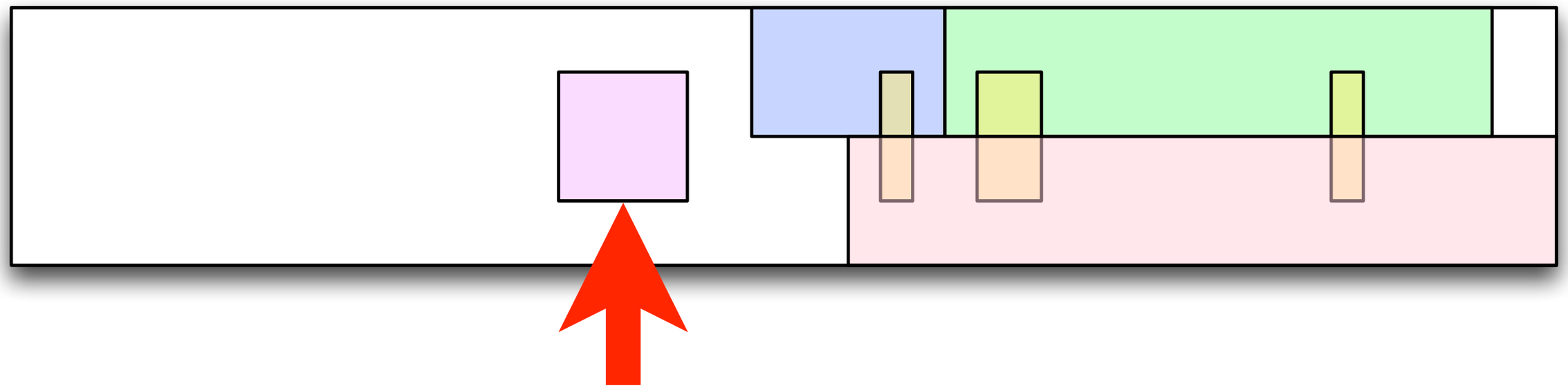
Sensitivity Stability

What I Found

Can Avert Catastrophic Interference
Add Noise to the Inputs
Increase the Scale of the Simulation

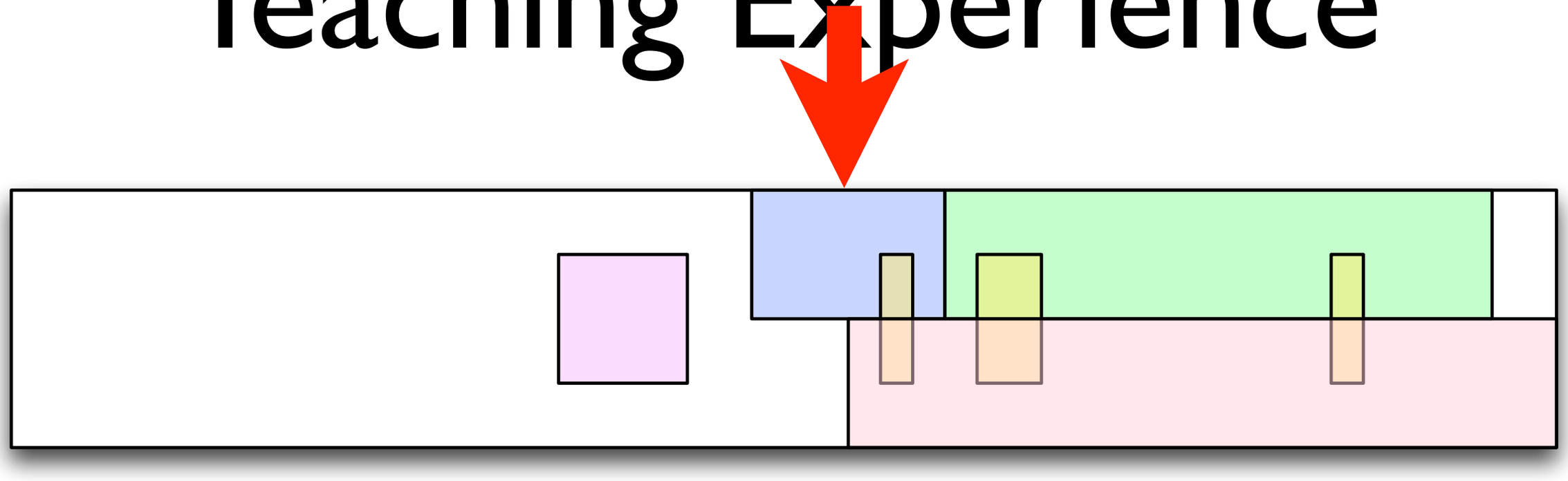
Teaching

Teaching Experience



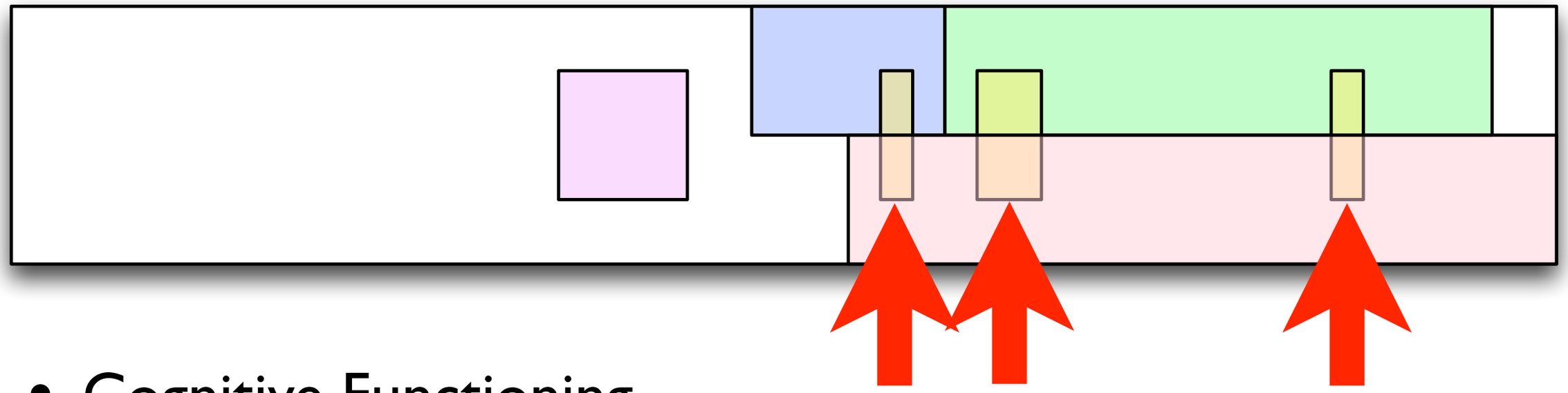
- Primarily teaching computer programming classes to children.
- Occasionally classes focused on applications (e.g., spreadsheets and word processors).
- Some classes were for adults, and some mixed children and adults.

Teaching Experience



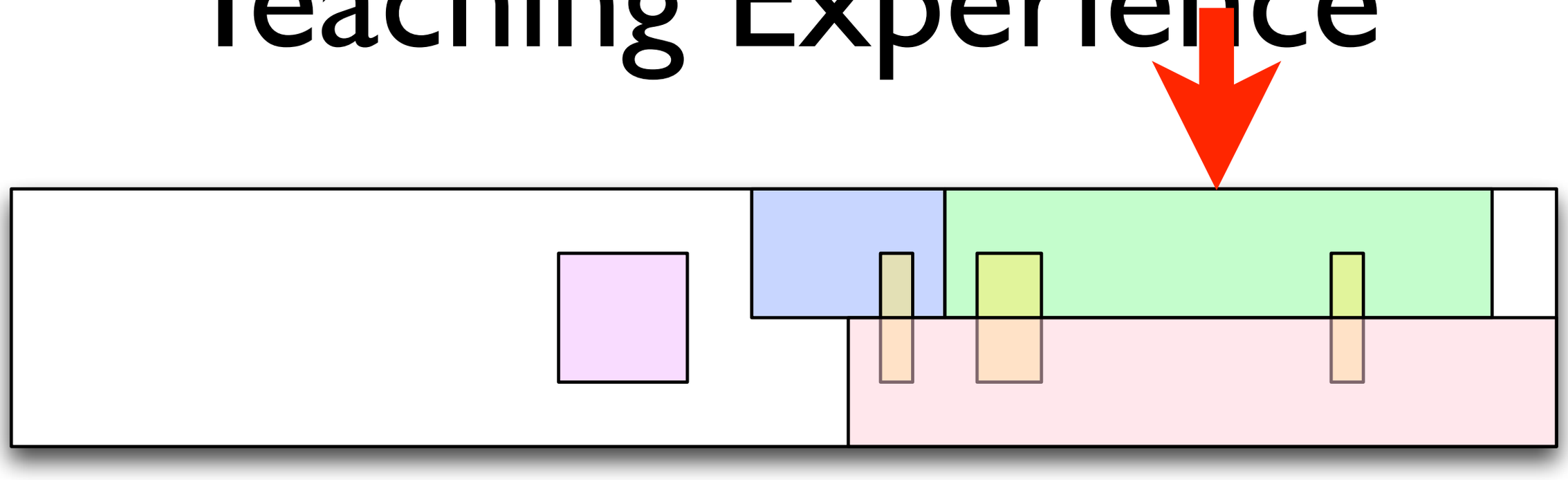
- EECS 183: Elementary Programming Concepts
- EECS 280: Programming & Introductory Data Structures
- EECS 283: Programming for Science & Engineering
- EECS 370: Introduction to Computer Organization
- EECS 380: Data Structures & Algorithms
- EECS 381: Systems Programming

Teaching Experience



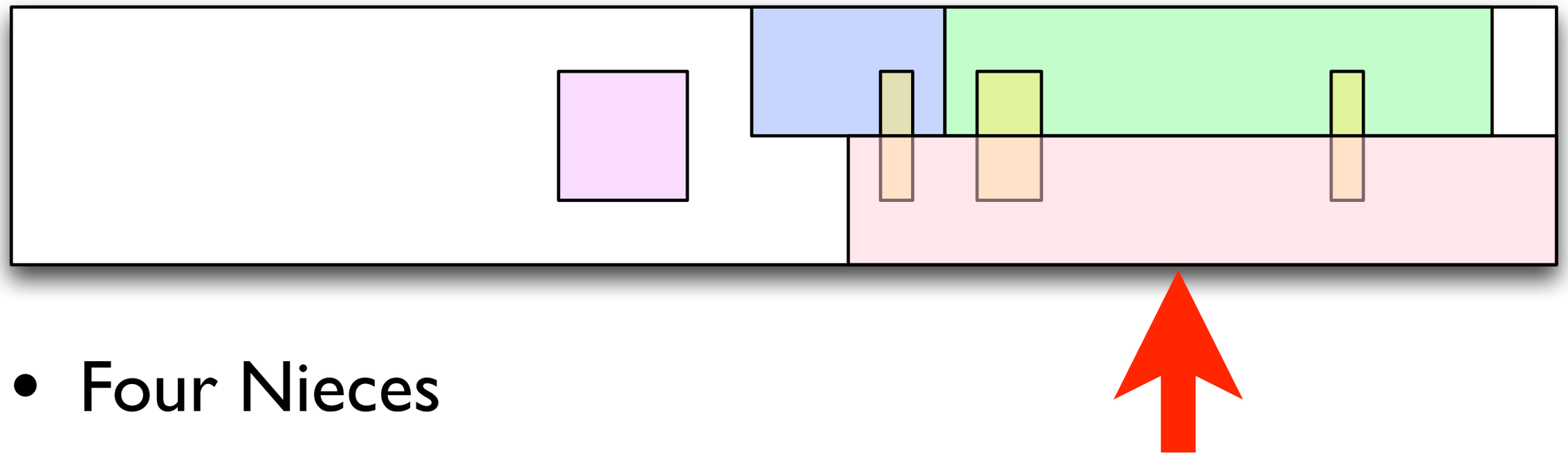
- Cognitive Functioning
  - Psych 649; Fall 1992, Fall 2006
- Cognition & Environment
  - Psych 703; Winter 1996
- Mechanisms of Mind: A Connectionist Approach
  - EECS 695/Psych 640; Winter 1995

Teaching Experience



- Primarily programming in a new language — Java, C, C++, Ruby
- Frameworks (e.g., web application, service-oriented)
- OS APIs (e.g., socket/network programming, inter-process communication)
- Operating systems principles and usage

Teaching Experience



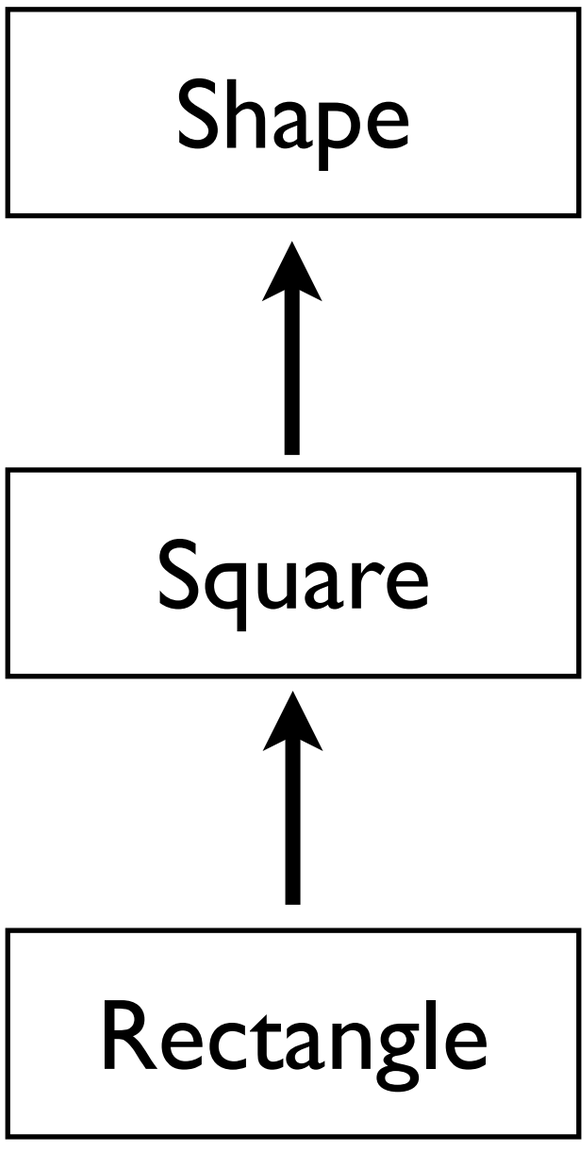
- Four Nieces
  - Lots of math homework — elementary & middle school
  - Programming in Scratch
  - A bit of everything else — science, reading, writing, spelling, vocabulary, ...

Class Inheritance in Object-Oriented Programming Languages

Stage-Setting

- Classes in an Object Oriented language
- Inheritance
  - Subclasses represents a more specific type of the superclass
- That in subclasses you can:
  - Add new variables
  - Add new methods
  - Override existing methods

Shape, Square, and Rectangle



Shape is abstract; has notion of the shape's center, but not much else

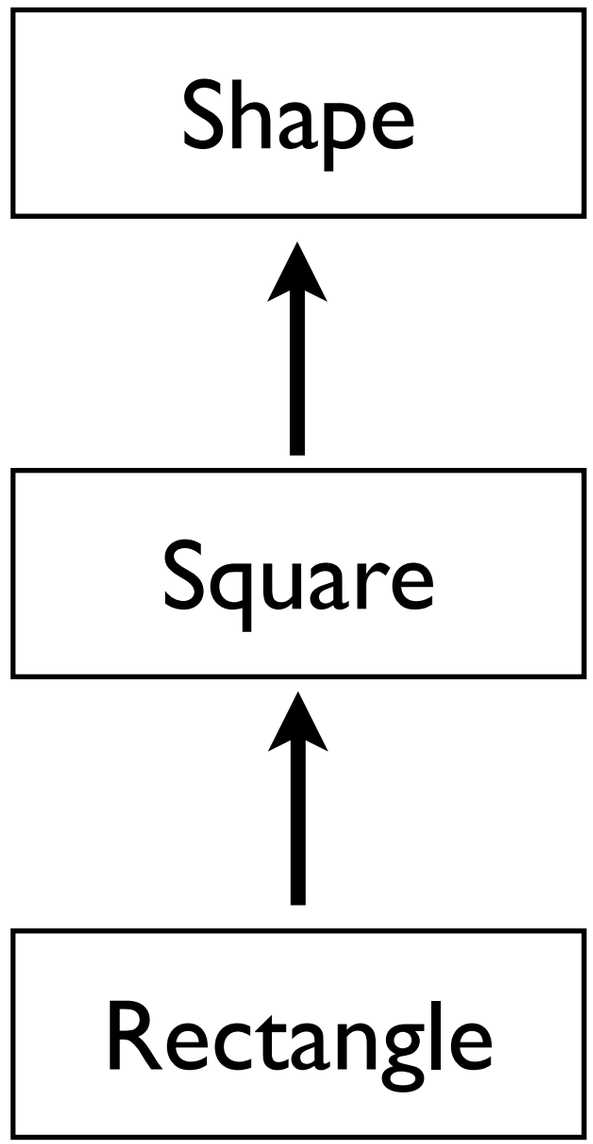
Square has one dimension that is used for both width and height.

Rectangle has two independent dimensions for width and height.

Add variable for 2nd dimension!

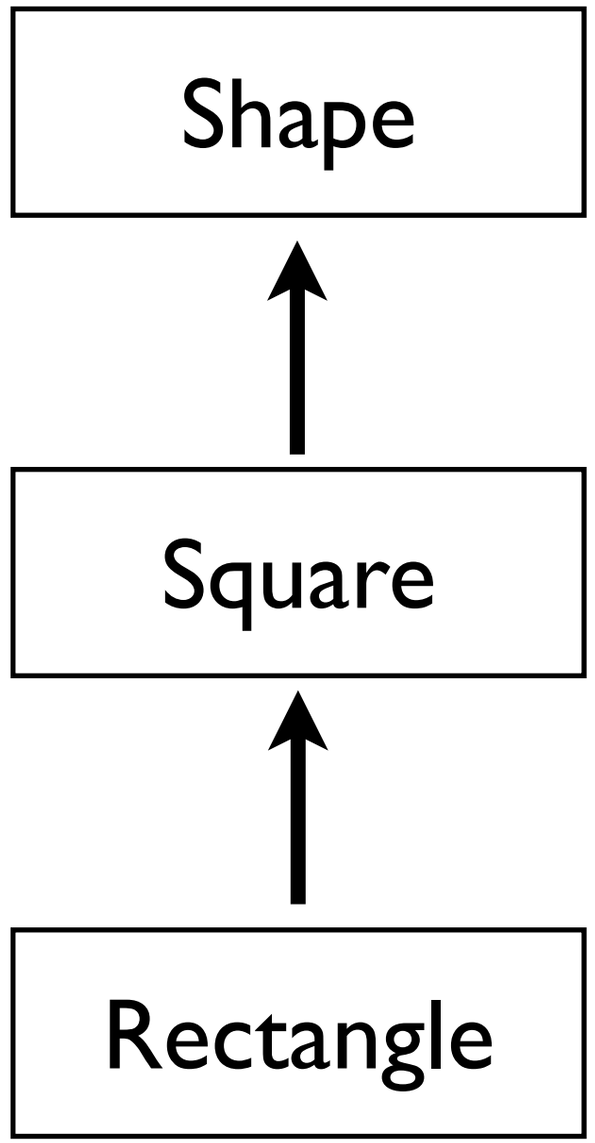
A Problem

```
class Shape {
    // ...
    boolean isBoundedBy(Rectangle r);
}
```



A Problem

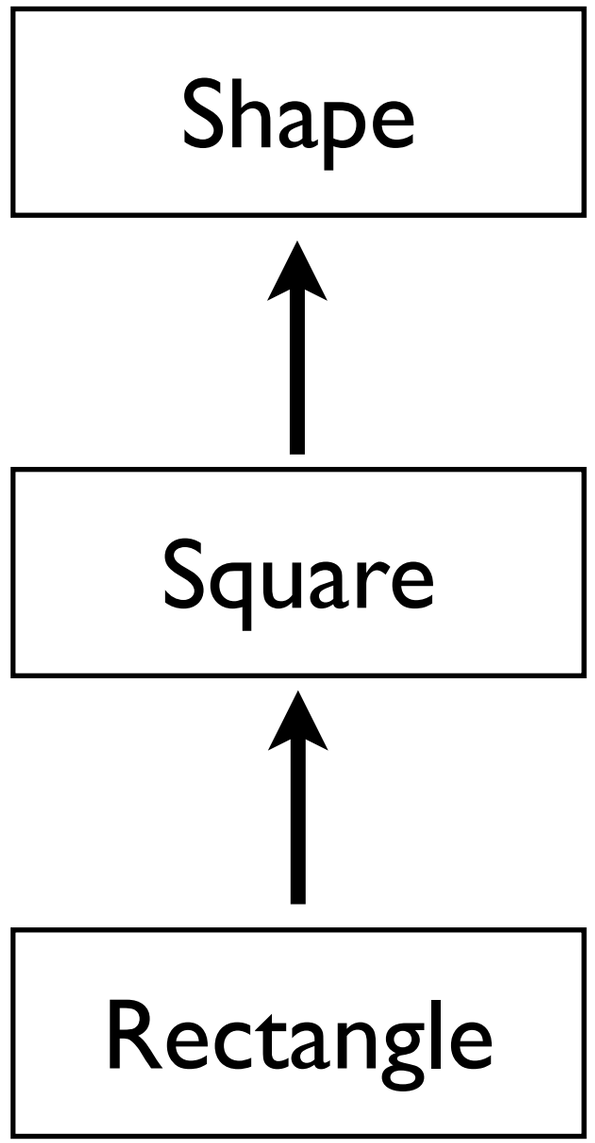
```
class Shape {
    // ...
    boolean isBoundedBy(Rectangle r);
}
```




```
Shape shape = // ...
Square tile = // ...
if (shape.isBoundedBy(tile)) {
    // ...
}
```

A Problem

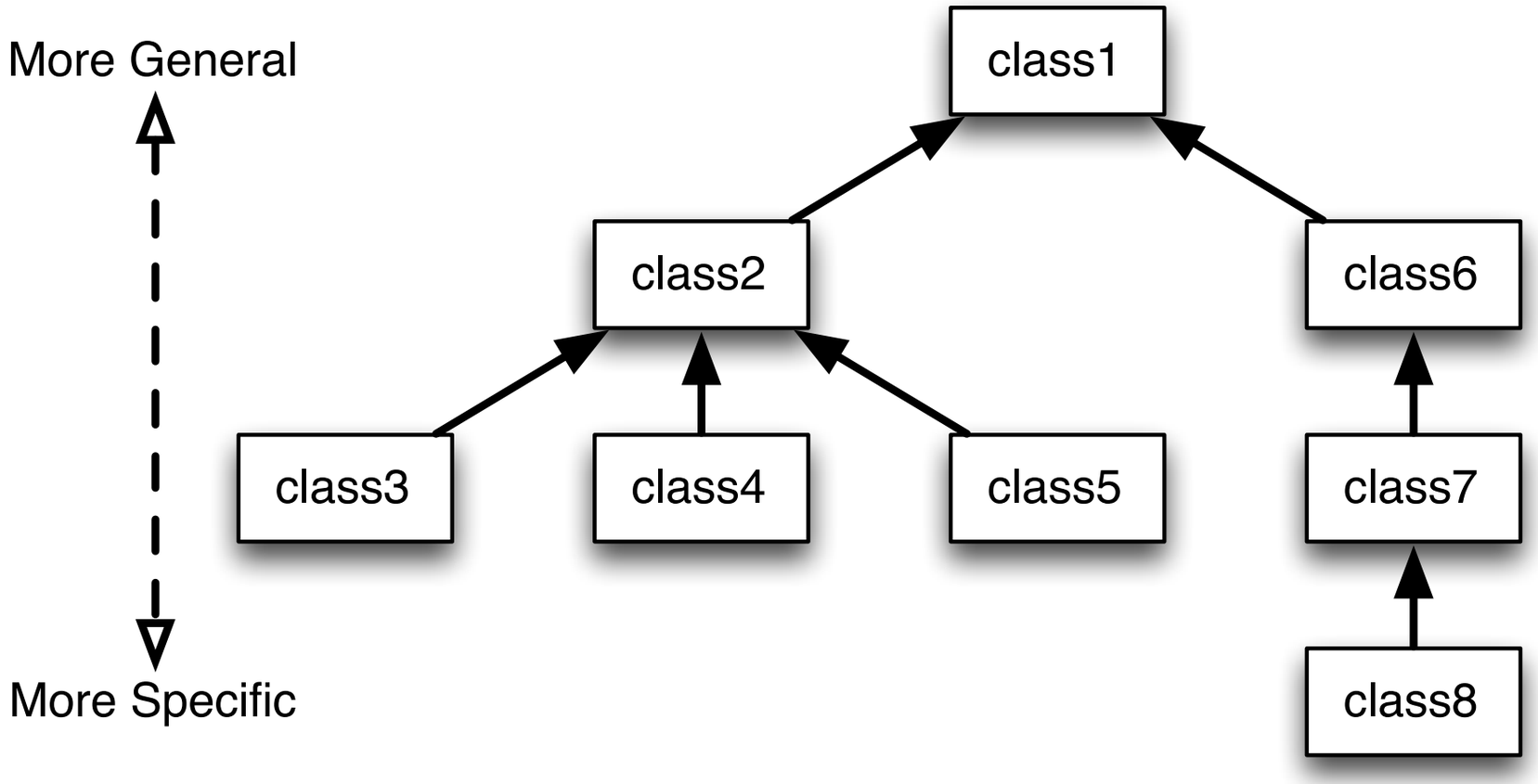
```
class Shape {
    // ...
    boolean isBoundedBy(Rectangle r);
}
```




```
Shape shape = // ...
Square tile = // ...
if (shape.isBoundedBy(tile)) {
    // ...
}
```

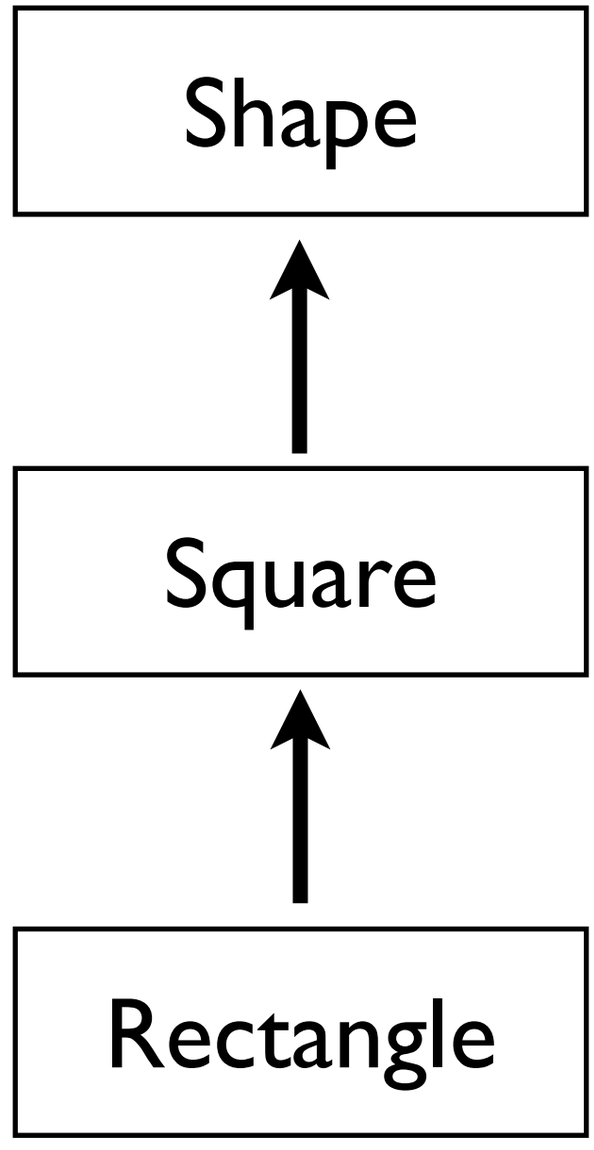
Compiler Error: Square cannot serve as a Rectangle.

Pattern of Abstraction in Class Hierarchies



The IS-A Test

- Subclassing represents specialization
- Square IS-A Shape?
  - **YES**
- Rectangle IS-A Square?
  - **NO**, in fact most rectangles are not squares!
- Square IS-A Rectangle?
  - **YES**, it's a Rectangle with an added constraint



Rectangle

```
class Rectangle extends Shape {
    double width;
    double height;

    void setWidth(double newWidth) {
        width = newWidth;
    }

    // ...
}
```

Square

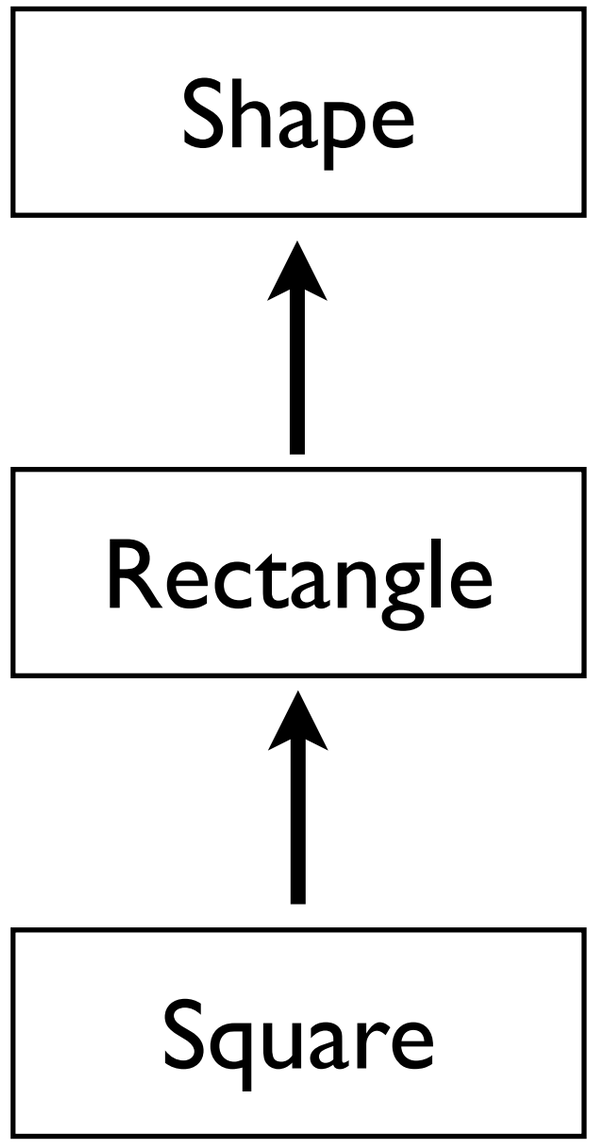
```
class Square extends Rectangle {
    // inherits width and height

    // override setWidth to maintain "squareness"
    void setWidth(double newWidth) {
        width = newWidth;
        height = newWidth;
    }

    // ...
}
```

It's Fixed!

```
class Shape {
    // ...
    boolean isBoundedBy(Rectangle r);
}
```




```
Shape shape = // ...
Square tile = // ...
if (shape.isBoundedBy(tile)) {
    // ...
}
```

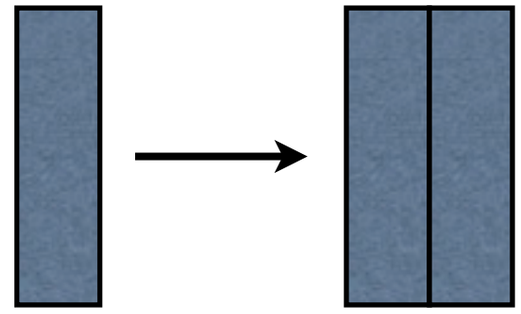
Later...

```
void doubleArea(Rectangle r) {
    // ...

    // double r's area
    double width = r.getWidth();
    r.setWidth(2 * width);

    // ...
}
```

Later...

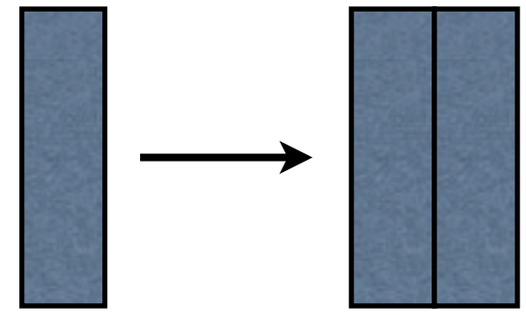


```
void doubleArea(Rectangle r) {
    // ...

    // double r's area
    double width = r.getWidth();
    r.setWidth(2 * width);

    // ...
}
```

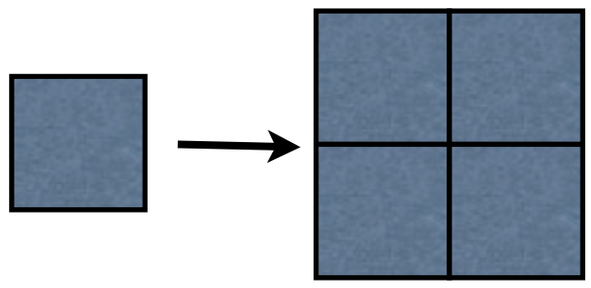
Later...



```
void doubleArea(Rectangle r) {
    // ...

    // double r's area
    double width = r.getWidth();
    r.setWidth(2 * width);

    // ...
}
```



Liskov Substitution Principle

"Let *q(x)* be a property provable about objects *x* of type *T*.
Then *q(y)* should be provable for objects *y* of type *S*
where *S* is a subtype of *T*."

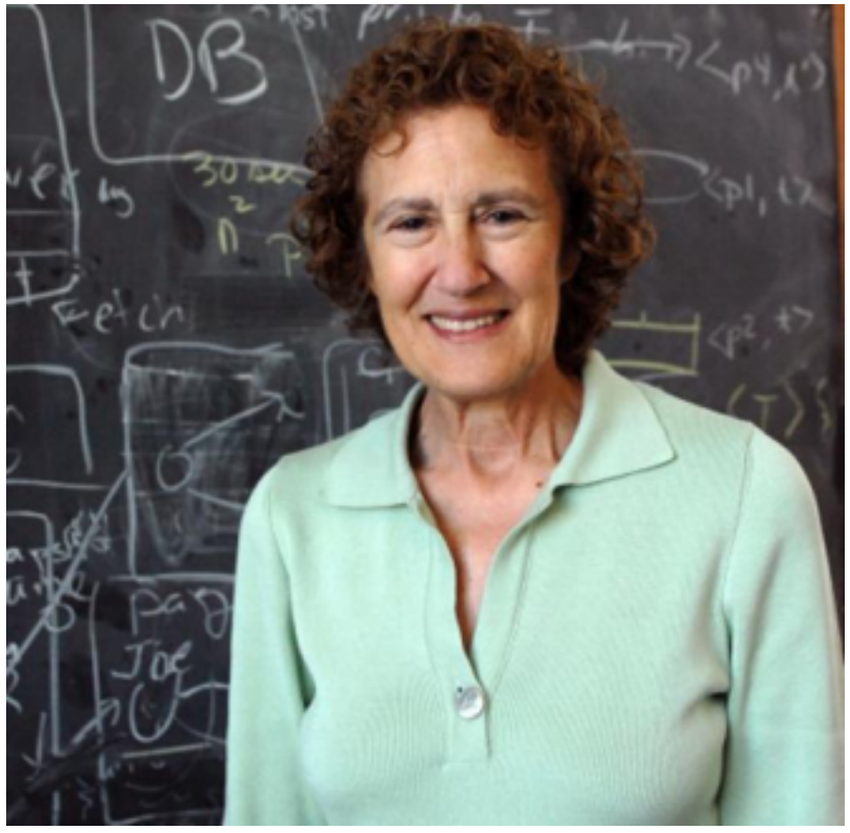
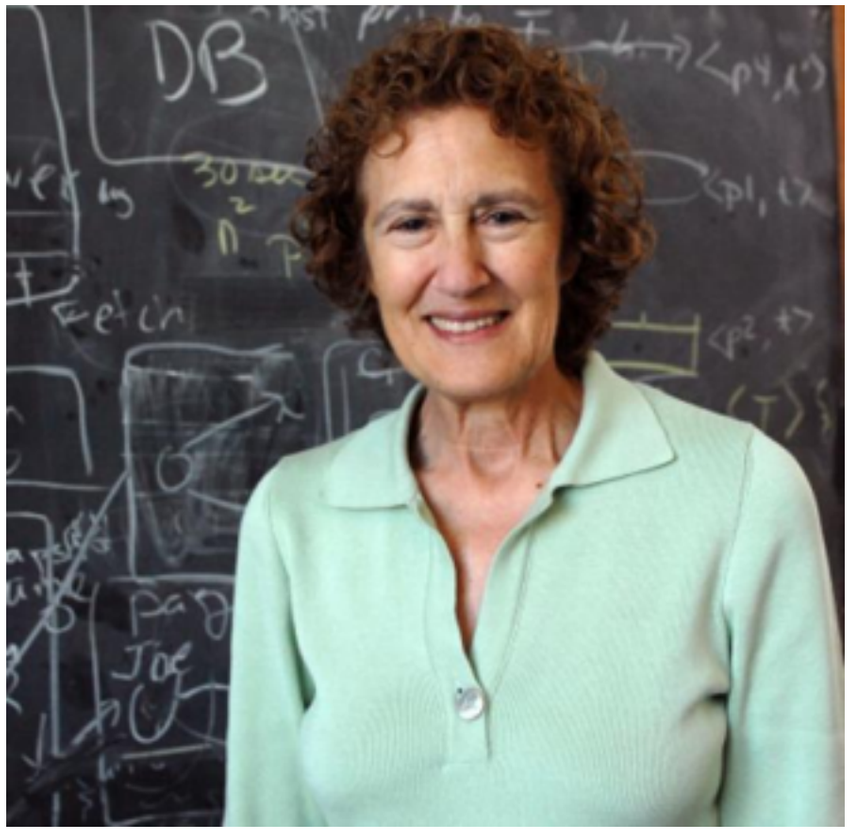


Image from Boston.com; credited to Donna Coveney/ MIT.
Quote from Liskov and Wing 1994.
Principle originally described in Liskov 1987.

Liskov Substitution Principle

"Let *q(x)* be a property provable about objects *x* of type *T*.
Then *q(y)* should be provable for objects *y* of type *S*
where *S* is a subtype of *T*."



T *x* *q(x)*

↑

S *y* *q(y)*

Image from Boston.com; credited to Donna Coveney/ MIT.
Quote from Liskov and Wing 1994.
Principle originally described in Liskov 1987.

Liskov Substitution Principle

For subclassing to work well...

We should be able to substitute an instance of a subclass

for an instance of a superclass

without violating any of the assumptions inherent in the superclass.

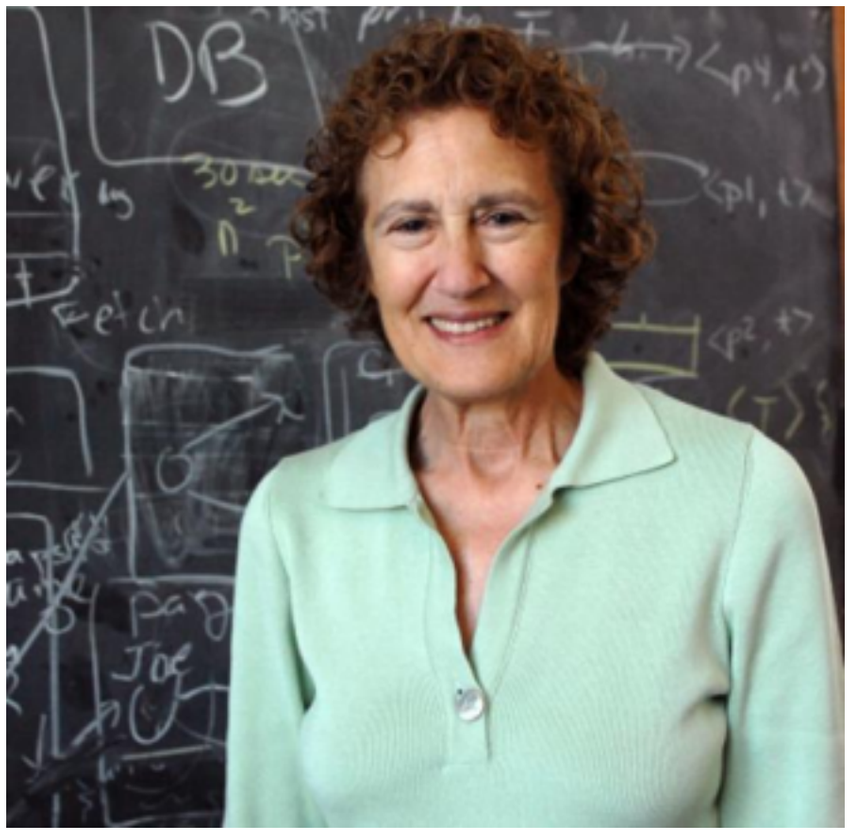
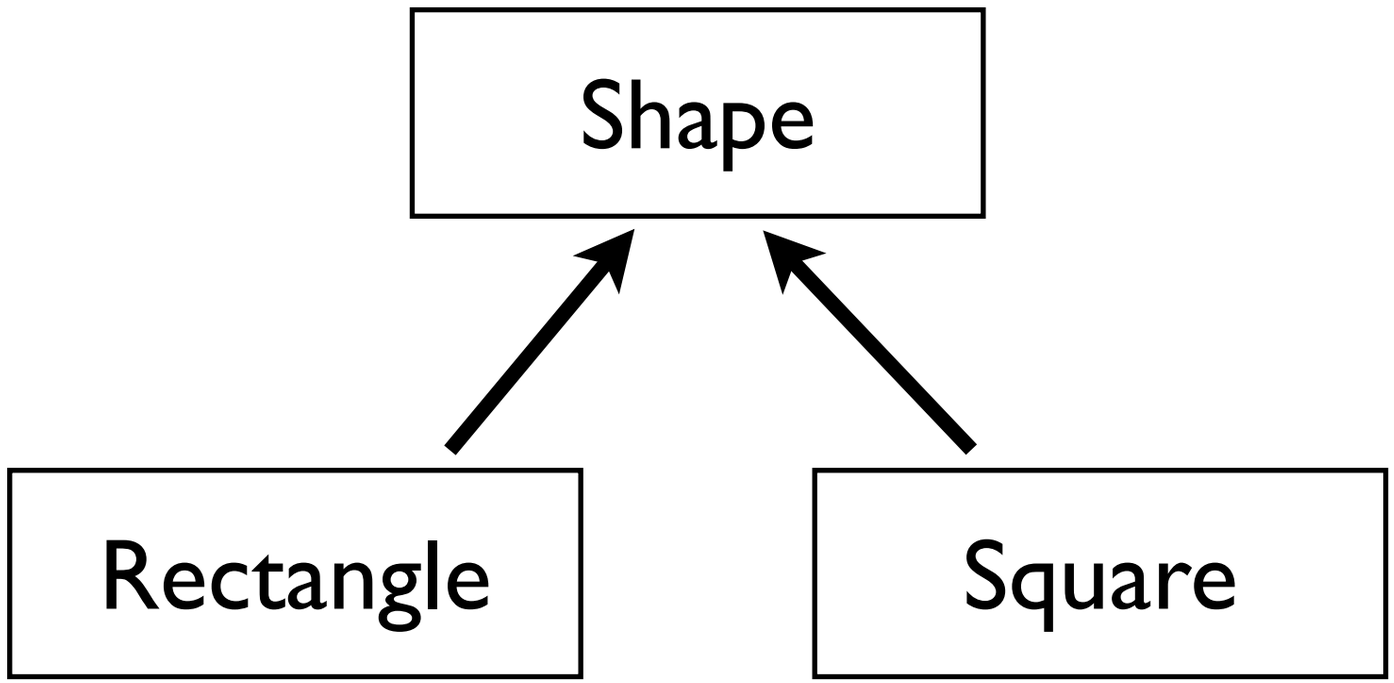


Image from Boston.com; credited to Donna Coveney/ MIT.
Principle originally described in Liskov 1987.

Version 3



Converting Square to Rectangle

```
class Square extends Shape {
    // ...

    Rectangle getEquivalentRectangle() {
        return new Rectangle(...);
    }
}
```

What We'd Cover Next...

- We've only set things up for a deeper discussion of the Liskov Substitution Principle
- Next we'd explore related issues of:
  - covariance and contravariance
  - pre-conditions, post-conditions, invariants
  - exceptions

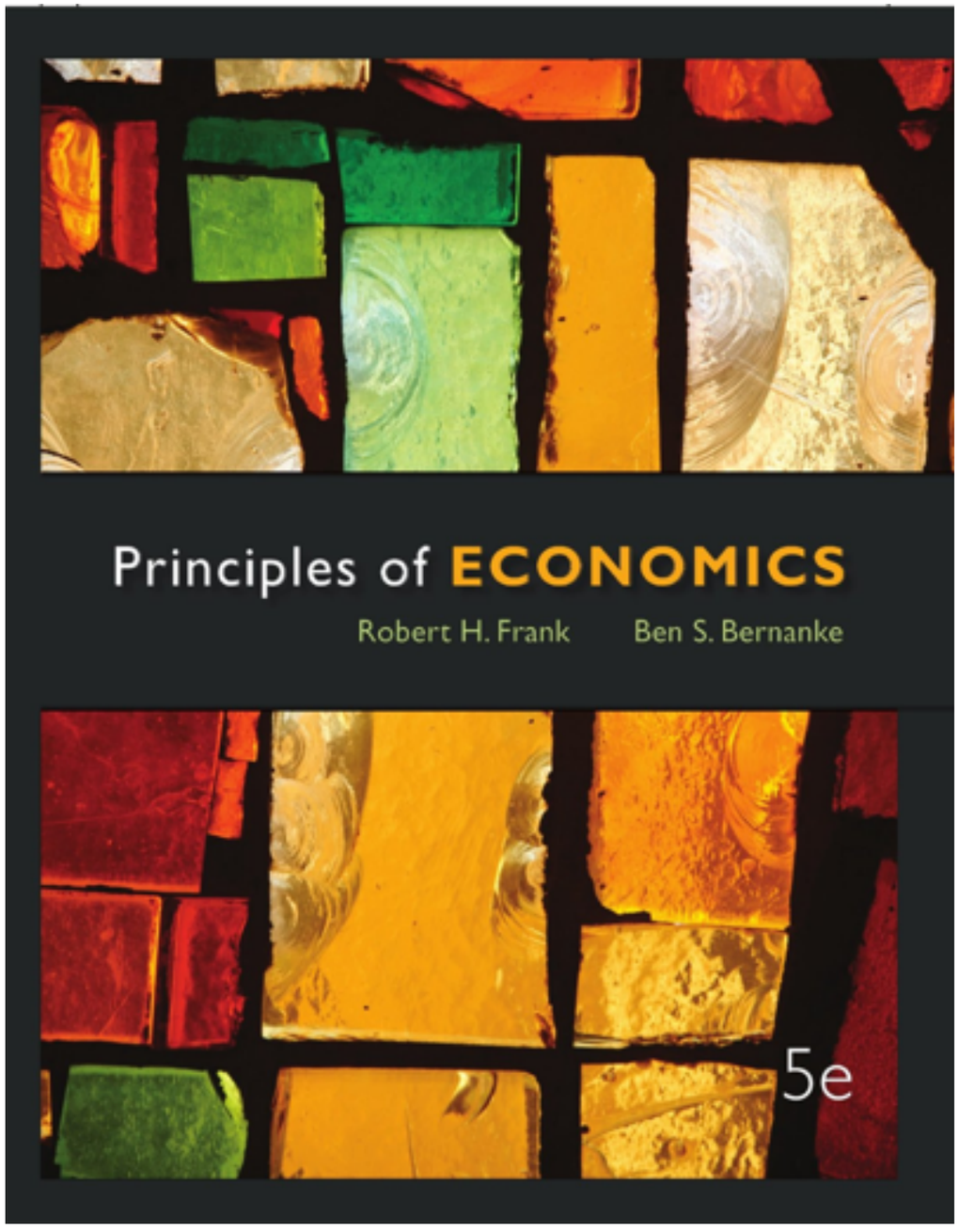
Philosophy / Lessons Learned

Philosophy / Lessons Learned

1. Learning that is Integrated and Durable
2. Abstraction Hierarchies and Learning
3. Leveraging the Clarity Mechanism

Learning that is Integrated and Durable

Economics 101



Robert H. Frank

Ben S. Bernanke

Frank Interview: http://www.insidehighered.com/news/2007/06/01/frank
Frank Image: http://www.johnson.cornell.edu/Faculty-And-Research/Profile.aspx?id=rhf3
Bernanke Image: http://en.wikipedia.org/wiki/File:Ben_Bernanke_official_portrait.jpg

The Problem

- Test scores 6 months after taking Econ 101
- Paul Ferraro & Laura Taylor (2005) study
  - 25% / 17.2% / 7.4%
- "The intermediate course teaches everything that's in the traditional principles course for the second time. It's repetitive in that sense — they need to do that, really, because the students didn't learn it the first time."

Guess, A. (2007) "Economics Education 101: An Interview with Robert Frank," Insider Higher Ed. http://www.insidehighered.com/news/2007/06/01/frank
Ferraro and Taylor (2005): http://www2.gsu.edu/~wwwcec/docs/ferrarotaylorbep.pdf

Diagnosis

- "... big encyclopedic texts with thousands of topics on the syllabus thrown at students ..."

- "... much of [the information] in the form of equations and graphs ..." / "... extremely convoluted and complex graphs ..."

- Failure to connect the concepts to the real world.

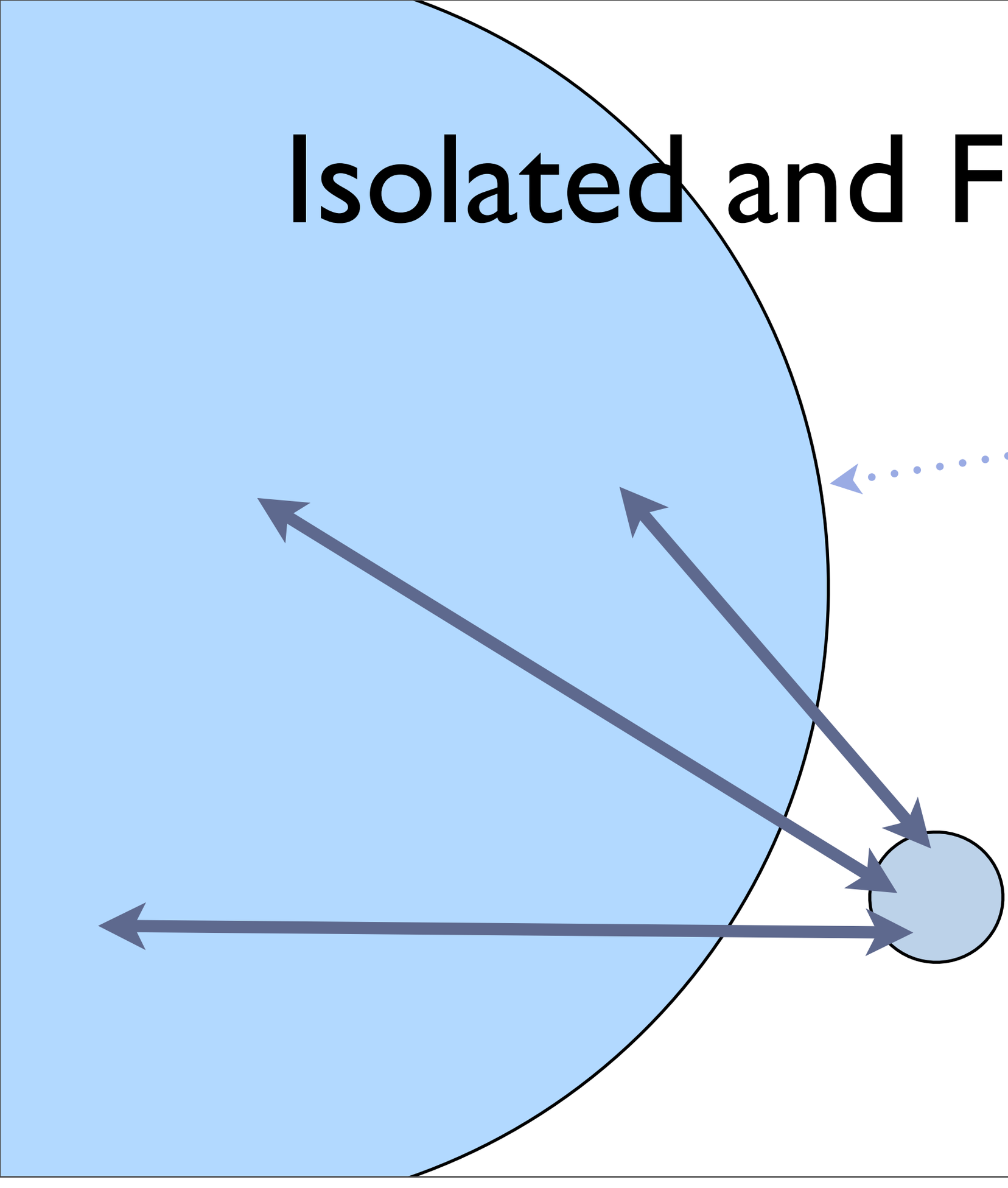
New Approach

- Focus on 5 - 6 key ideas that "do most of the heavy lifting in economics"
- Use stories to demonstrate the ideas
  - (Graphs are fewer and more simple)
- The "economic naturalist" writing assignment

Findings

- "it's not easy for [students] to come up with an interesting question the first time...."
- "By the time the second assignment ... 'Oh, Professor Frank, can I do a medley? I've got these three great questions!' "
- "So, it's quite visible in cases of those students that there's a lot of rewiring going on in the brain over the course of the semester. They're learning to look around them and see economic principles at work in ordinary settings...."

Isolated and Fragile



Learn by Using

- Cover a few new concepts
  - Provide some concrete examples
  - Show them how the concepts translate to code
- Give participants an exercise in which they build new code with those concepts
  - Helps them consolidate their understanding
  - Creates a situation in which they can
    - Discover and fill in gaps
    - Confront possible wrong assumptions

Designing Programming Exercises

Dan Meyer TED Talk: http://www.ted.com/talks/dan_meyer_math_curriculum_makeover.html

Designing Programming Exercises

Dan Meyer TED Talk: http://www.ted.com/talks/dan_meyer_math_curriculum_makeover.html

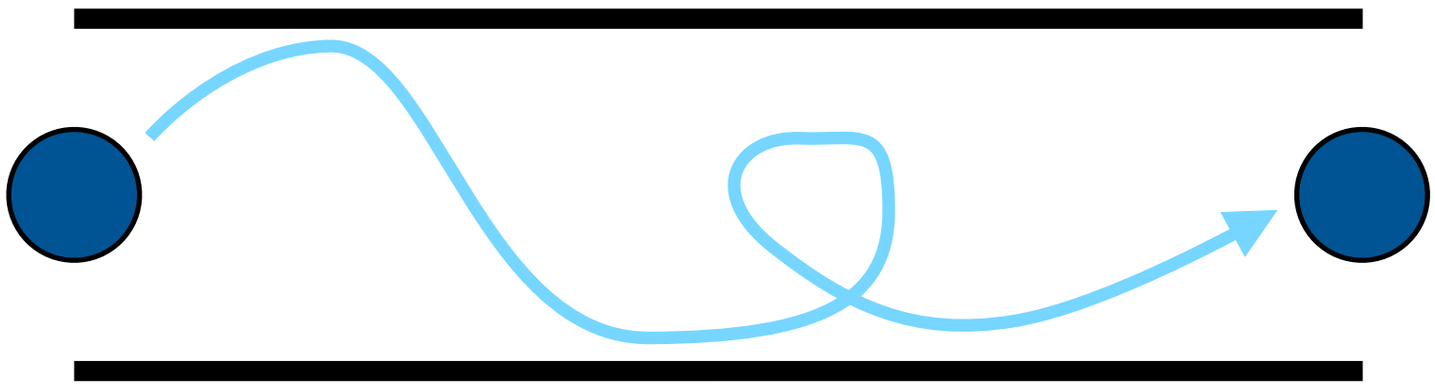
Designing Programming Exercises

"... we're paving a smooth, straight path from one to the other and congratulating our students for how well they can step over the small cracks in the way." — Dan Meyer

Dan Meyer TED Talk: http://www.ted.com/talks/dan_meyer_math_curriculum_makeover.html

Designing Programming Exercises

"... we're paving a smooth, straight path from one to the other and congratulating our students for how well they can step over the small cracks in the way." — Dan Meyer



Dan Meyer TED Talk: http://www.ted.com/talks/dan_meyer_math_curriculum_makeover.html

Abstraction Hierarchies and Learning

Which programming language should someone learn first?

Bottom-Up View

Quoting one participant in ruby-talk May 2008 (emphasis added):

> Ruby is an awesome language, highly abstracted from the hardware, incredibly flexible and fluid, and probably not a good first language just for those reasons alone.
>
> **Learn C first.**
>
> **To really get programming, to understand what's going on, you need to go deep. All the way down to C (though some say assembler).**
>
> C is the lingua franca of computers. **It doesn't make things easy for you, it doesn't make things pretty, or necessarily intuitive, but it does bring you right down to the metal in the end.**
>
> **You grab your own memory**, and are responsible for putting it back. **You make and move pointers to access the memory**: if you point to the wrong place, you'll get the wrong data, corrupt your own program, and probably crash it. You've got to link your own binaries, and link them to the right libraries.
>
> **And doing all that makes you a better programmer, makes you understand what really is going on behind the scenes.** Not to mention makes you appreciate languages like ruby, perl, python and java so much more when you get to them **:)**

Bottom-Up View

Quoting a second participant in ruby-talk May 2008 (emphasis added):

> **This is exactly why we chose C as the first language for electronic engineering and information systems engineering students.** Previously the course had been given in Algol 68 (!), then Pascal, but it was decided that C would be a much better foundation, and useful in the real world too.
>
> **C is very hard to learn. Almost everybody comes unstuck on pointers and memory allocation. But this trains your mind, and once you've learnt C, you realise what other languages are protecting you from, and how they work internally (since most are written in C, like Ruby).**
>
> Recently some ex-students of mine contacted me via a social networking site to thank me, saying that C had been very valuable to them in their careers.
>
> **If the OP wants a good foundation in programming, C will provide it. But Ruby would be gentler. :-)**

Bottom-Up vs. Top-Down

Quoting a third participant in ruby-talk May 2008 (emphasis added):

> It's a question of whether you learn top-down, or bottom-up. Different people have different learning preferences.
>
> **Traditionally technical subjects have been taught bottom-up in academia ("building on a firm foundation"), but there are very good arguments for the top-down approach too.**

Degrees of Abstraction

High-Level

Low-Level

Anonymous Functions, First-Class Functions, Closures, Continuations
Classes, Type Hierarchies, Data Structures & Algorithms, Recursion
Expressions, Statements, Variables, Control Structures, Subroutines, Simple Types, Variable Scope
Addresses, Pointers / Pointer Arithmetic, Memory Management, Interrupts, Mutexes

Degrees of Abstraction

More Abstract
High-Level

Low-Level
Nearer Hardware/OS

Anonymous Functions, First-Class Functions, Closures, Continuations
Classes, Type Hierarchies, Data Structures & Algorithms, Recursion
Expressions, Statements, Variables, Control Structures, Subroutines, Simple Types, Variable Scope
Addresses, Pointers / Pointer Arithmetic, Memory Management, Interrupts, Mutexes

Degrees of Abstraction in Other Categories

Animal	Furniture
Mammal, Reptile, Insect, Bird	
Dog, Cat, Tiger, Snake	Chair, Table, Shelves
Beagle, Poodle, Golden Retriever	Rocking Chair, Recliner, Swivel Chair

Degrees of Abstraction in Other Categories

Animal	Furniture
Mammal, Reptile, Insect, Bird	
Dog, Cat, Tiger, Snake	Chair, Table, Shelves
Beagle, Poodle, Golden Retriever	Rocking Chair, Recliner, Swivel Chair

Basic-Level Categories

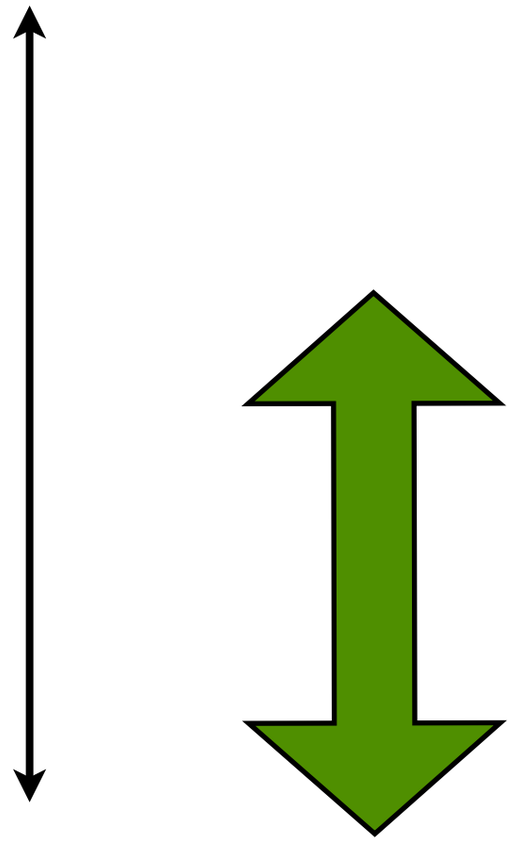
- Most easily grouped by overall shape
- Highest level at which we interact with objects similarly
- People most likely to use basic level names ("what are you sitting on?")
- Basic level names often have fewer syllables
- Children learn the basic level first

Eleanor Rosch

Rosch, E.H.; Mervis, C.B.; Gray, W.D.; Johnson, D.M.; Boyes-Braem, P. (1976). "Basic objects in natural categories". Cognitive Psychology 8 (3): 382–439. doi:10.1016/0010-0285(76)90013-X.
Rosch, E., "Principles of Categorization", pp. 27–48 in Rosch, E. & Lloyd, B.B. (eds), Cognition and Categorization, Lawrence Erlbaum Associates, Publishers, (Hillsdale), 1978.

Learning How to Program

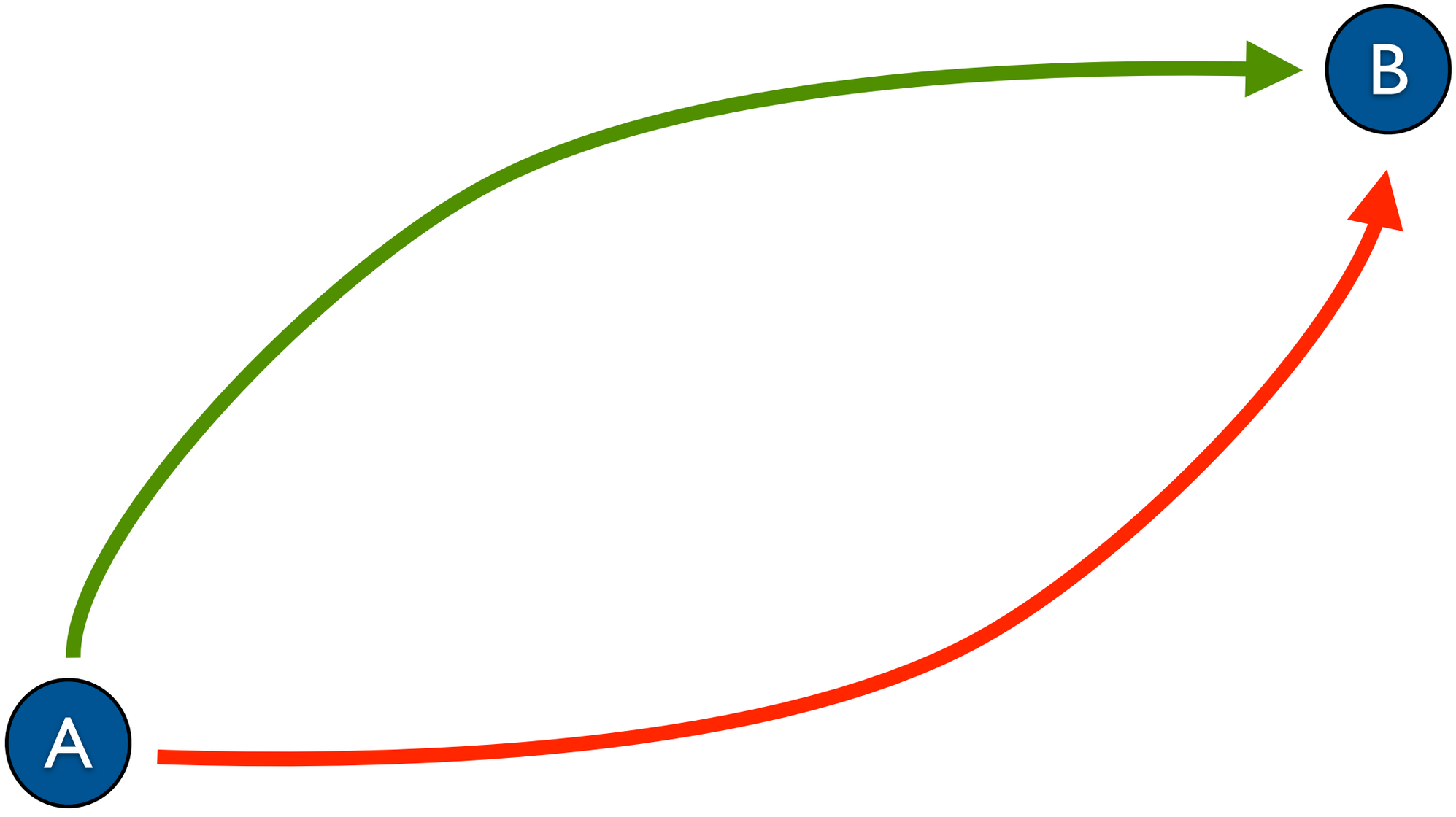
More Abstract
High-Level



Low-Level
Nearer Hardware/OS

Anonymous Functions, First-Class Functions, Closures, Continuations
Classes, Type Hierarchies, Data Structures & Algorithms, Recursion
Expressions, Statements, Variables, Control Structures, Subroutines, Simple Types, Variable Scope
Addresses, Pointers / Pointer Arithmetic, Memory Management, Interrupts, Mutexes

Multiple Paths



Leveraging the Clarity Mechanism

I did not have time to go through this section of my talk.

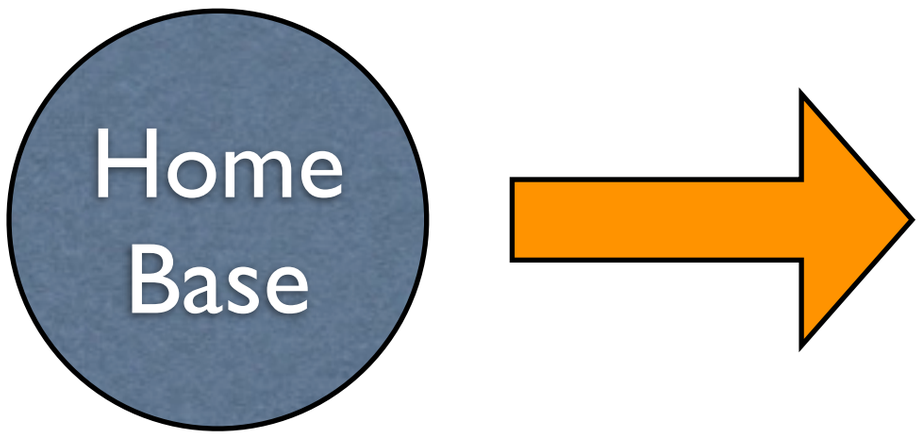
I will try to give a sense of what I intended to say at various points in this section by including some additional text in this shade of blue.

Humans and Knowledge

- We seek out new experiences
- We learn; we build internal models of our world, others, ourself
- We employ our knowledge
  - Our models provide us with expectations of what will likely happen next
  - When our models fail we pay attention

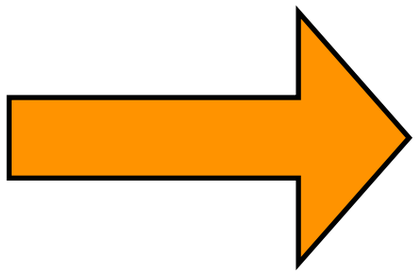
Knowledge-Seeking

Knowledge-Seeking



Bored

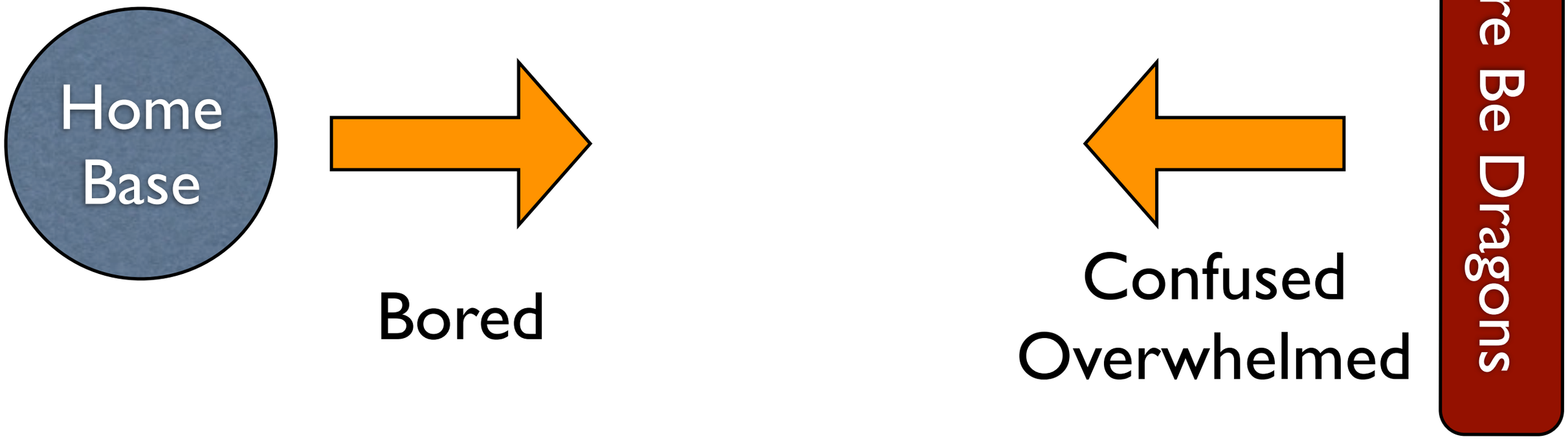
Knowledge-Seeking



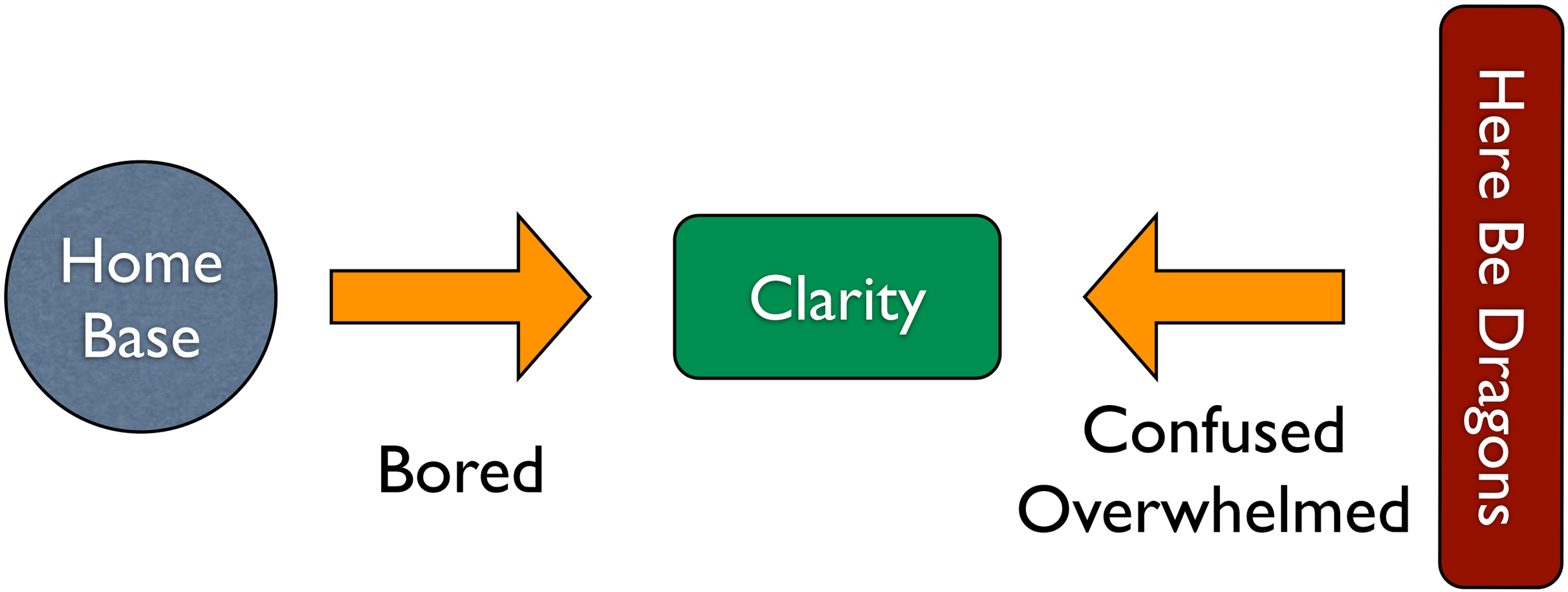
Bored

Here Be Dragons

Knowledge-Seeking



Knowledge-Seeking



The clarity mechanism is a theorized neural process that nudges us into situations where we are likely to learn.

It looks at the status of cognition recognizing the conditions of boredom or confusion and generating a proportional degree of discomfort in response (i.e., more confusion → more pain).

But we would be a sorry species if we simply tried to minimize our pain by finding a space between boredom and confusion.

Clarity

Clear about Things

+

Clarity

Things are Coming Together ++

Clear about Things +

Clarity

EUREKA! / AHA! +++

Things are Coming Together ++

Clear about Things +

Clarity provides anywhere from mild to intense cognitive pleasure.

Mild when things are moving along.

Stronger when things are coming together.

Intense when things fall into place — AHA!

Clarity
by Other Names

- Fascination
- Mystery
- Wonder
- Intrigue
- The Thrill of Discovery

Tickling Clarity

- Novels, Plays, Movies
- Card Games, Board Games, Video Games, Gambling
- Athletic Games — as Player or Fan/Follower
- Puzzles — Crosswords, Logic, Mechanical
- How-To Books, Magic Shows, Tourism, Court Cases, Political Elections
- Education — Formal and Informal

Many human activities provide us pleasure through the clarity mechanism.

Whole industries are built around tickling people's clarity mechanism.

How can clarity be used to improve learning?

When I was an undergraduate I took a physics series. One topic was polarized light. The lecture and demonstration that covered this topic are vivid to me decades later.

I will describe how that lecture worked.

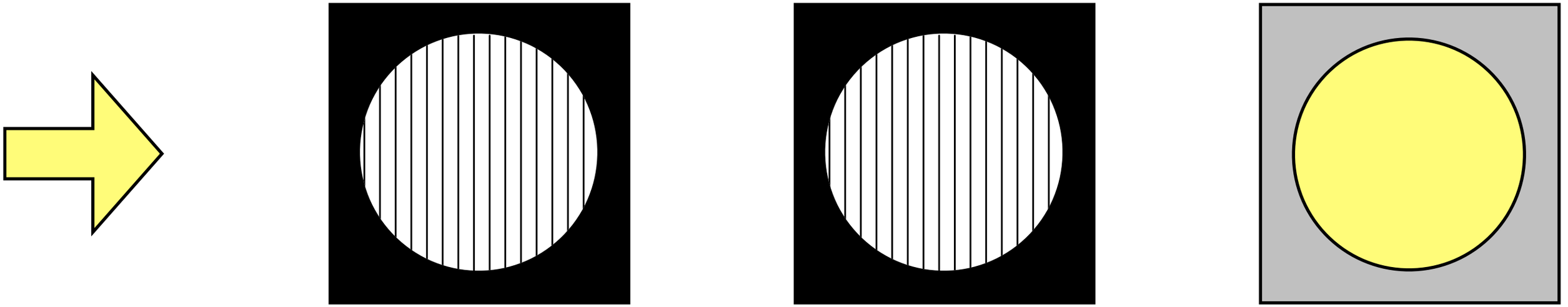
Prof. George Williams (Physics, Univ. of Utah) gave this lecture.

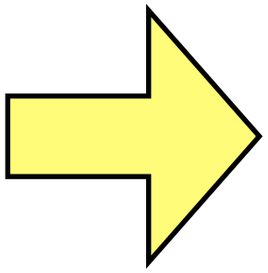
Light travels in a direction.

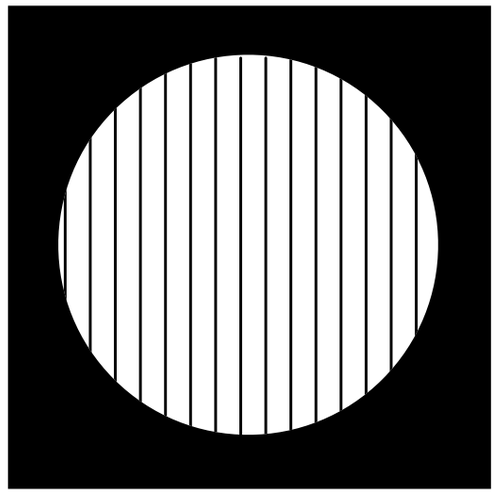
In addition, the light waves have an orientation perpendicular to the direction the light travels.

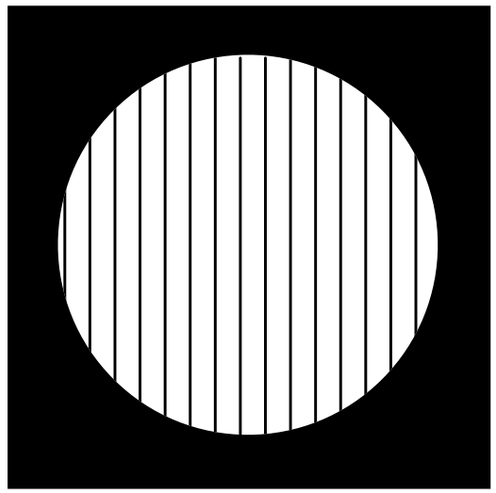
So if there were a beam of light traveling north, some waves would be in the up/down direction, others in the east/west direction, and others at all angles in between.

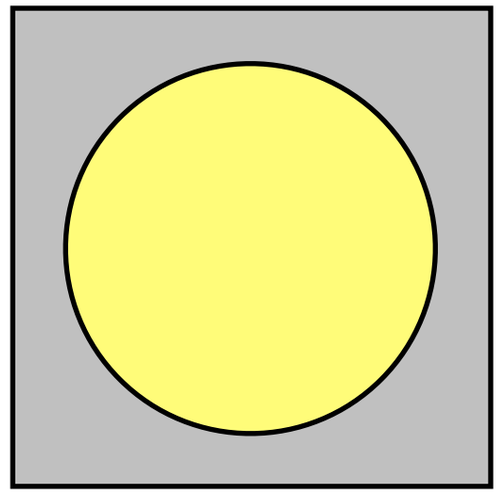
Polarized light is created when the orientation of the waves are all aligned, say through a polarizing filter.

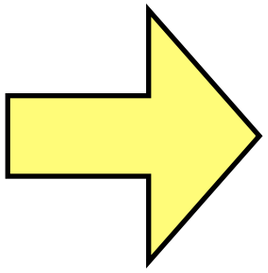


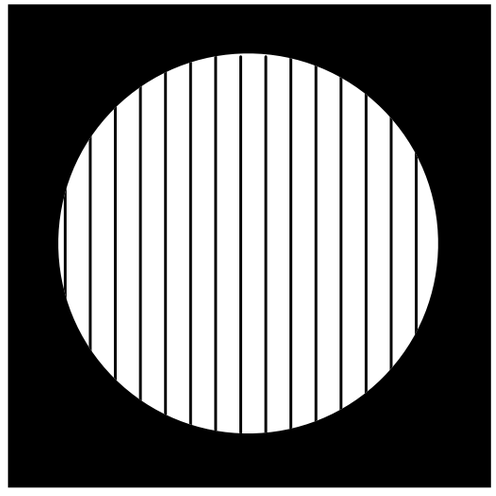


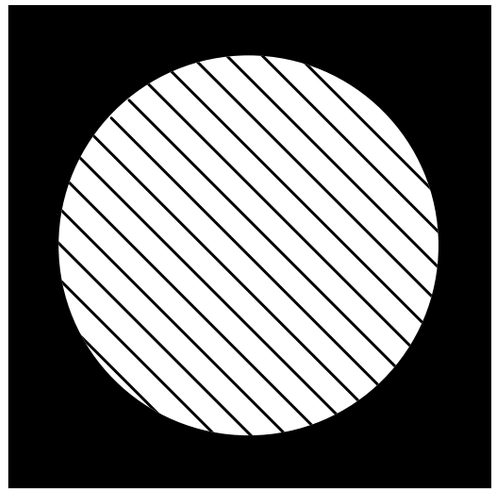


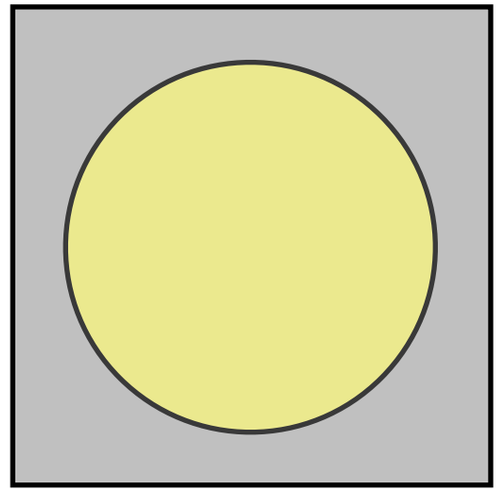


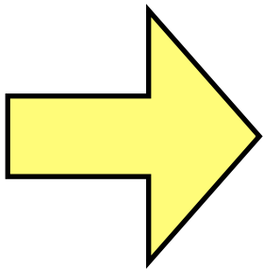


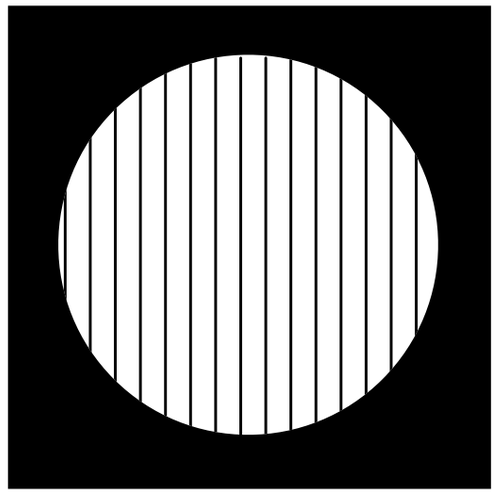


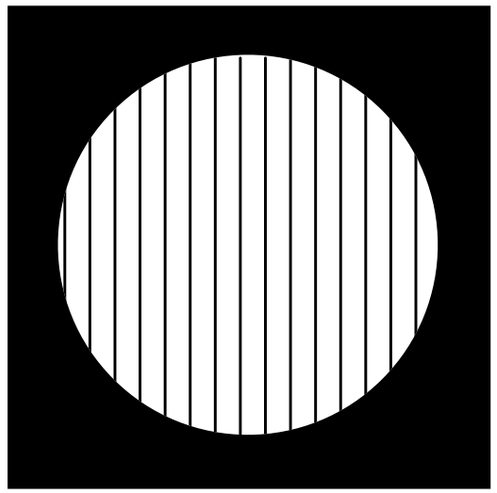


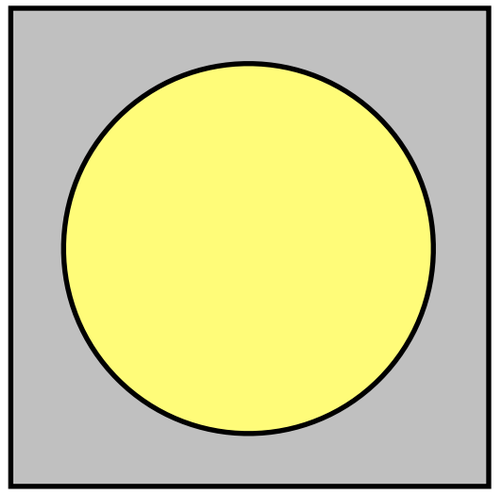


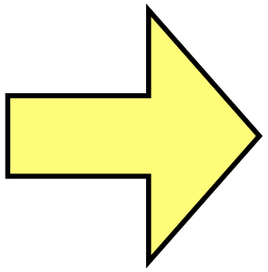


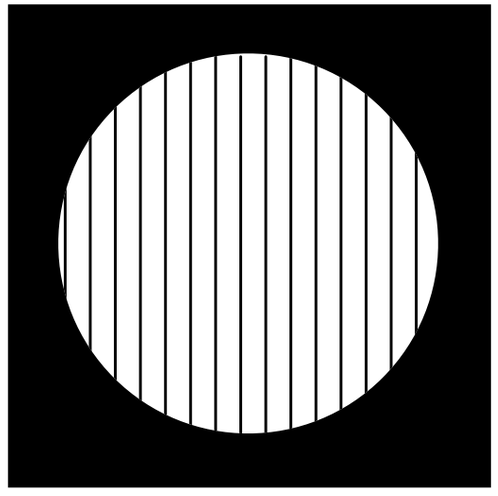


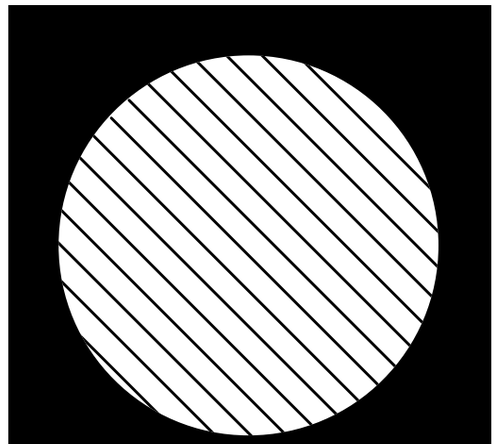


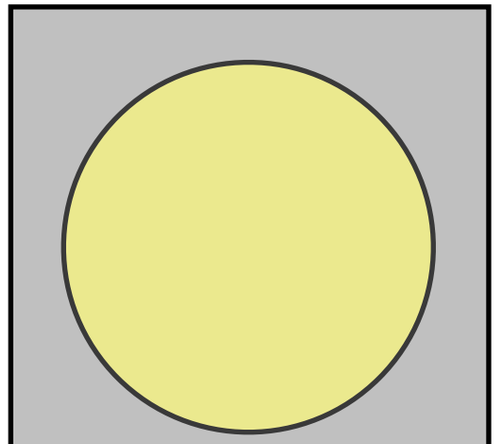


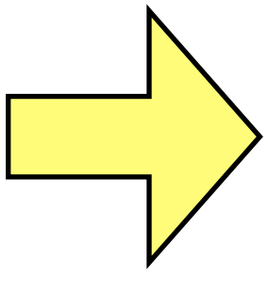


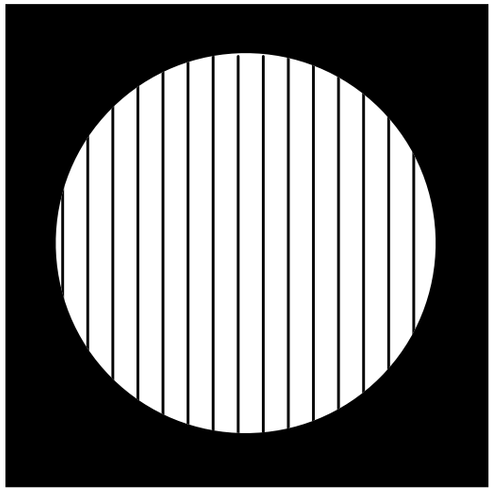


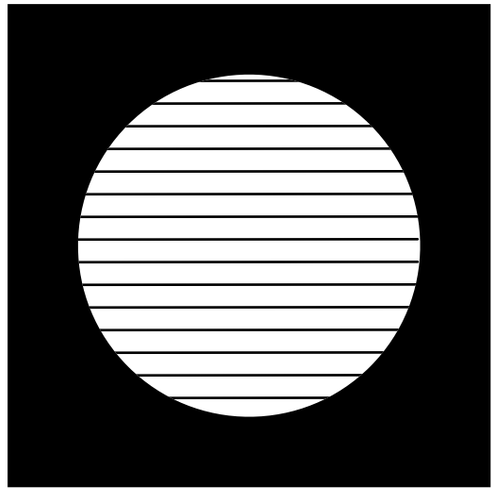


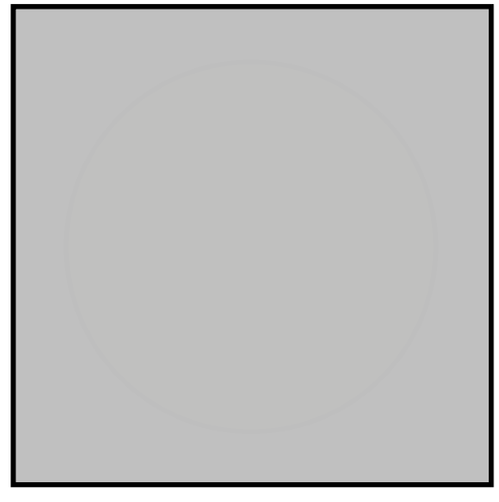












When light goes through a polarizing filter, all the light that emerges from the other side is polarized in the direction of the filter.

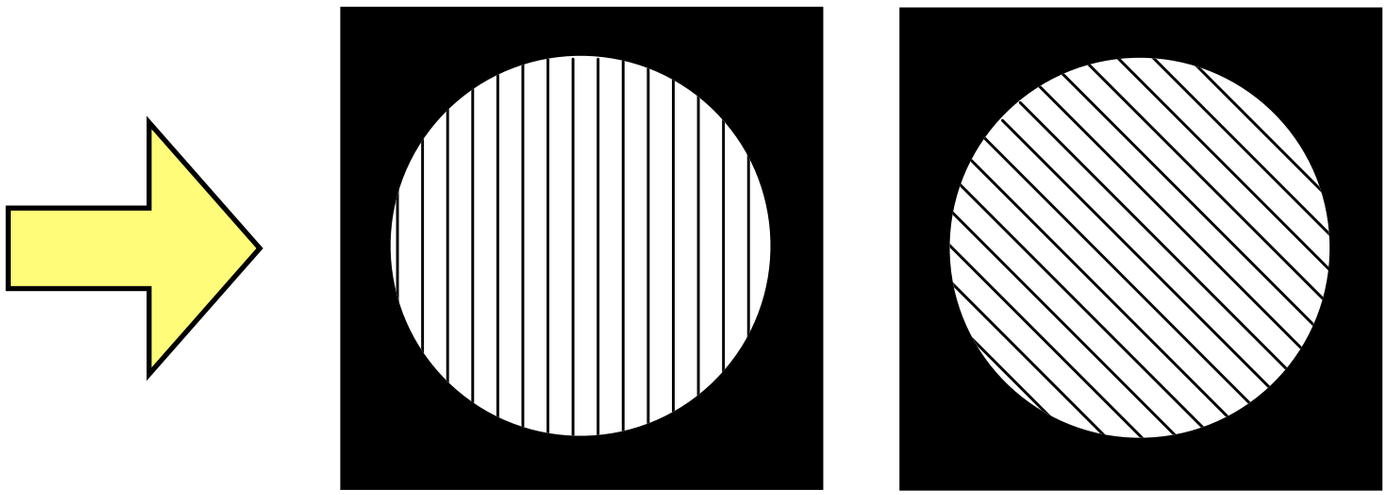
When polarized light (having gone through a filter) goes through a second filter there are two consequences:

1. the amount of light is proportional to the cosine of the difference of the angles of polarization of the two filters, and
2. any light that emerges from the second filter is polarized according to its orientation.

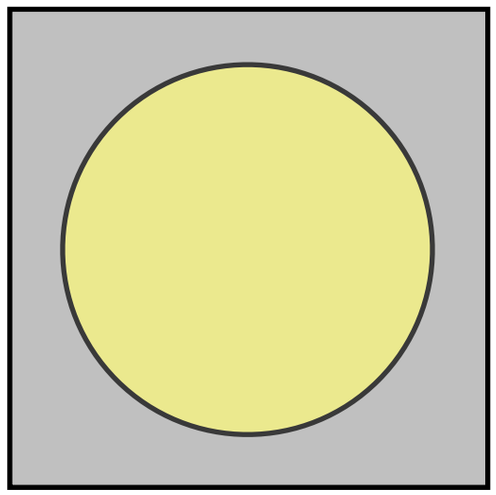
The cosine of 0° is 1.0, so the most light gets through the second filter when the two filters are aligned.

The cosine of 90° is 0.0, so no light gets through the second filter when the two filters are perpendicular.

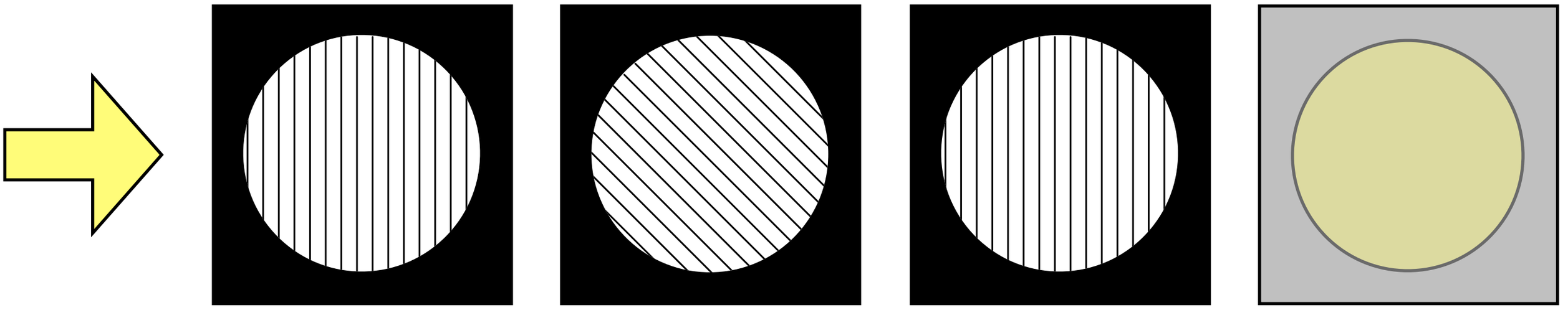
The cosine of 45° is about 0.7, so 30% of the light is lost when the two filters are at 45° to one another.



cos(45°)≈0.7



0.7



cos(45°)≈0.7 cos(45°)≈0.7

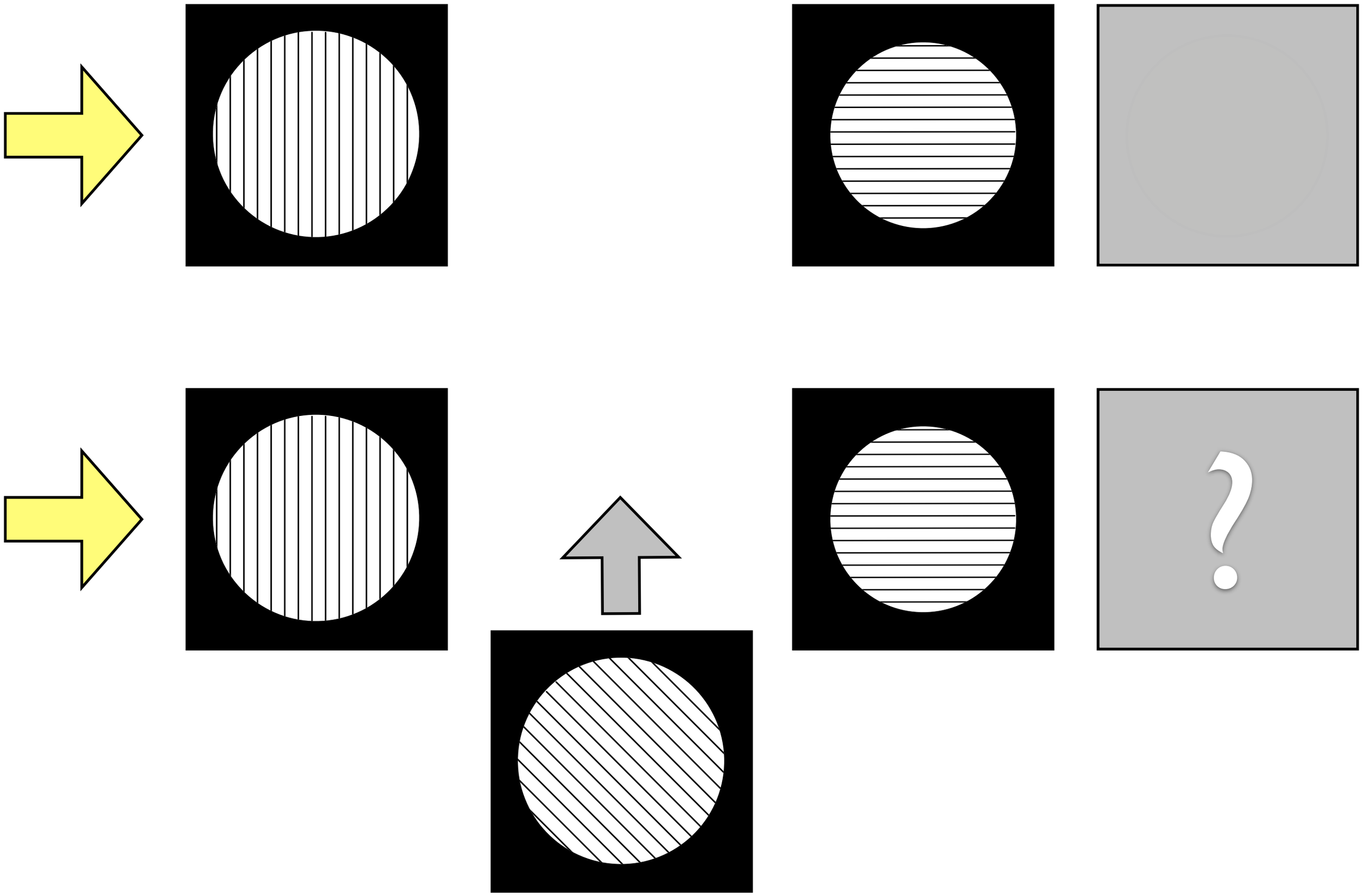
0.7x0.7 = 0.49

Since we have a model of understanding what happens to light when it travels through two filters, we can calculate what will happen when it goes through even more filters, treating each consecutive pair of filters at a time.

Now the Prof. Williams prepared the finale.

He set up the two "original" filters so they were 90° apart, and therefore no light passed through.

He then set up a third filter that was 45° from the two original filters and was about to insert it between the two original filters.

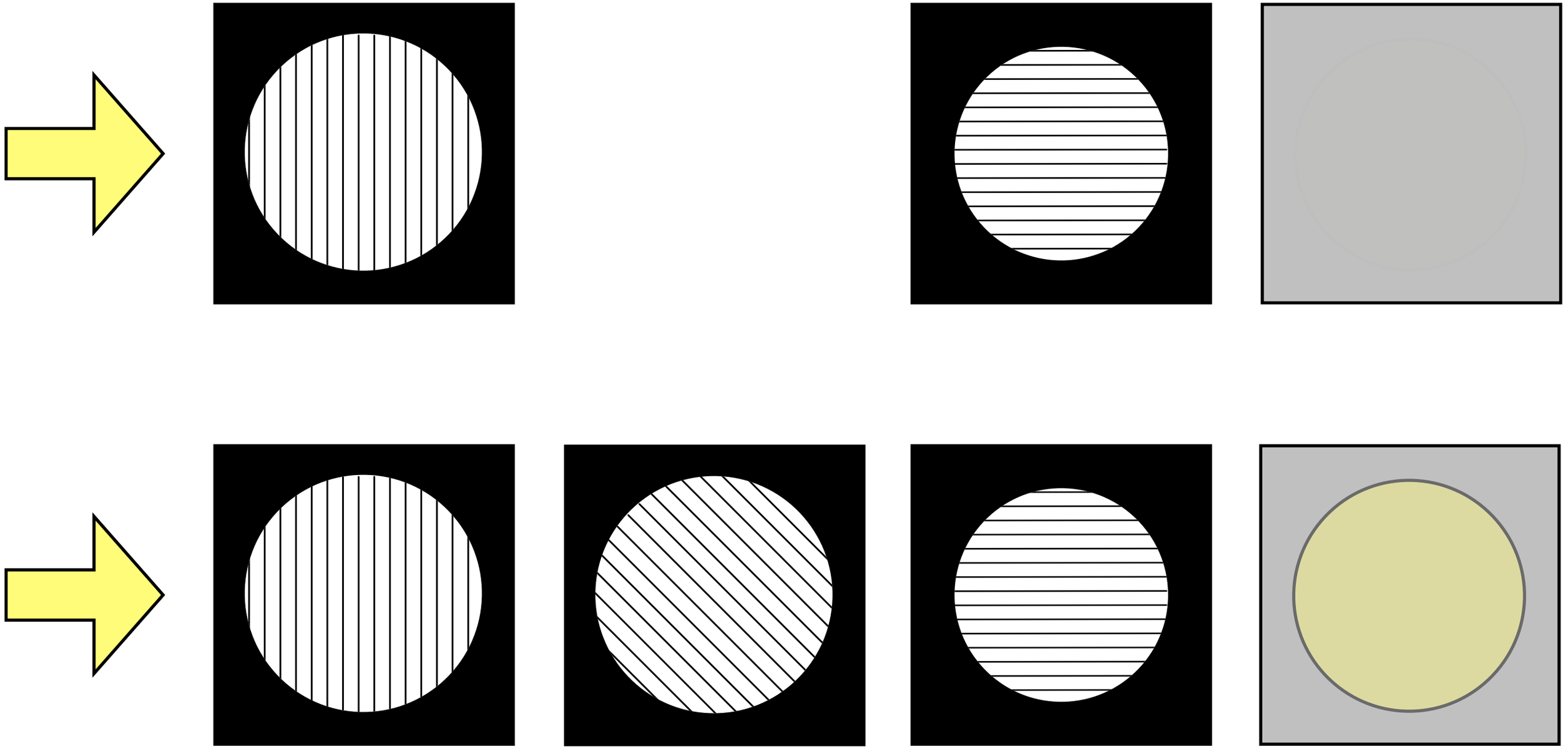


Prof. Williams asked us what would happen.

I was torn. Since there were two filters 90° apart it didn't seem like any light could get through. Besides, how could inserting ONE MORE FILTER between two existing filters INCREASE the light?

On the other hand, the model he gave us predicted that since the 1st and 2nd were 45° apart and the 2nd and 3rd were 45° apart, 70% of 70% should get through, or around 49%.

I was on the edge of my seat....



Sure enough the model was correct. Adding a third filter allowed light to get through where none had gotten through before!

Harnessing Clarity in Teaching

- Present issues as puzzles to be solved
- Get students to make a prediction in — and thereby become invested in — an outcome
- Lead students down the garden path and into a brick wall

Summary

- Create durable, integrated learning.
    - Change the way people perceive.
- Find the basic level — the sweet spot — and build outwards.
- Find ways to engage the clarity mechanism.

Thank You!

ivancich@umich.edu

https://github.com/ivancich/Ivancich-Lessons

Questions & Discussion Welcome!

Bibliography

Guess, A. (2007) "Economics Education 101: An Interview with Robert Frank," Insider Higher Ed. http://www.insidehighered.com/news/2007/06/01/frank

Hebb, D. O. (1972). A textbook of psychology (3rd ed., pp. 208–215). Philadelphia: W. B. Saunders Co.

Ivancich, J. E. (2005) Hebbian Networks for Averting the Problem of Catastrophic Interference. Ph.D. thesis, University of Michigan, Ann Arbor.

Kaplan, S. (1978). Attention and fascination: The search for cognitive clarity. In S. Kaplan & R. Kaplan (Eds.) Humanscape (pp. 84–90). Belmont, CA: Duxbury. Republished by Ann Arbor, MI: Ulrich's 1982.

Kaplan, S. and Kaplan, R. (1982). Cognitive chaos: Attention and stress (ch. 5). In Cognition and environment: Functioning in an uncertain world. New York: Praeger. Republished by Ann Arbor, MI: Ulrich's, 1989.

Kaplan, S. (1991). Beyond rationality: Clarity-based decision making. In T. Gärling and G. Evans (Eds.) Environment, cognition and action: An integrative multidisciplinary approach (pp. 171–190). New York: Oxford University Press.

Liskov, B. H. (1987). "Data abstraction and hierarchy". OOPSLA '87 Keynote, October 4. http://portal.acm.org/citation.cfm?id=62141

Liskov, B. H. and Wing, J. M. (1994) "A Behavioral Notion of Subtyping," ACM Trans. on Prog. Lang. and Systems, vol. 16, no. 6, November, pp. 1811-1841. http://www.cs.cmu.edu/~wing/publications/LiskovWing94.pdf

McClelland, J.L., D.E. Rumelhart and the PDP Research Group (1986). Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models, Cambridge, MA: MIT Press

Meyer, D. (2010) "Math class needs a makeover," TED Talk. http://www.ted.com/talks/dan_meyer_math_curriculum_makeover.html

Rosch, E.H.; Mervis, C.B.; Gray, W.D.; Johnson, D.M.; Boyes-Braem, P. (1976). "Basic objects in natural categories". Cognitive Psychology 8 (3): 382–439. doi:10.1016/0010-0285(76)90013-X.

Rosch, E., "Principles of Categorization", pp. 27–48 in Rosch, E. & Lloyd, B.B. (eds), Cognition and Categorization, Lawrence Erlbaum Associates, Publishers, (Hillsdale), 1978.

Rumelhart, D.E., J.L. McClelland and the PDP Research Group (1986). Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations, Cambridge, MA: MIT Press

Two Great Links

- If you're interested in learning and teaching, I recommend these two links:
- The Robert Frank interview:
  http://www.insidehighered.com/news/2007/06/01/frank
- The Dan Meyer Ted Talk:
  http://www.ted.com/talks/dan_meyer_math_curriculum_makeover.html

Supplementary Material

Ferraro & Taylor (2005) Opportunity Cost Question

Please Circle the Best Answer to the Following Question:

You won a free ticket to see an Eric Clapton concert (which has no resale value). Bob Dylan is performing on the same night and is your next-best alternative activity. Tickets to see Dylan cost $40. On any given day, you would be willing to pay up to $50 to see Dylan. Assume there are no other costs of seeing either performer. Based on this information, what is the opportunity cost of seeing Eric Clapton?

A. $0
B. $10
C. $40
D. $50

I never took an Economics 101 course, and I got the Ferraro & Taylor (2005) question wrong. But here's their explanation of which answer is correct and why.

A missed benefit is a loss; and a missed loss is a benefit. By not going to the Dylan concert, you missed paying for the ticket, so that's +$40 (a benefit). But you also miss a benefit, so that's -$50 (a loss). They combine to a net loss of -$10. So the opportunity cost is $10, or answer "B".