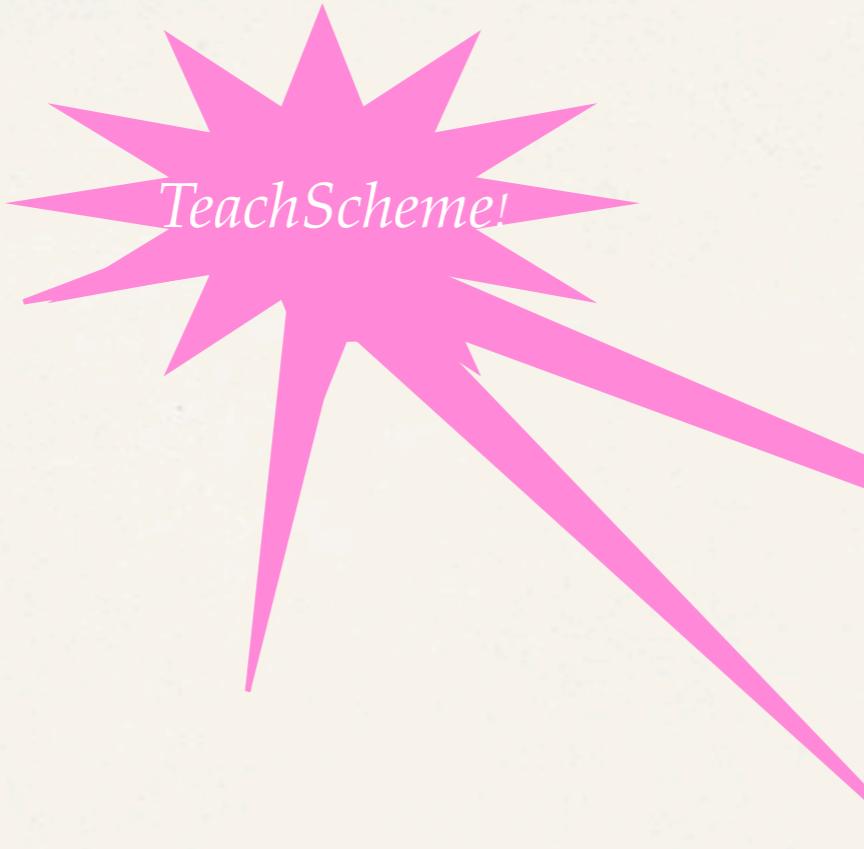


The *Teach Scheme!* Project

Matthias Felleisen (PLT)
Northeastern University
Boston, MA

The *Teach Scheme, NOT* Project



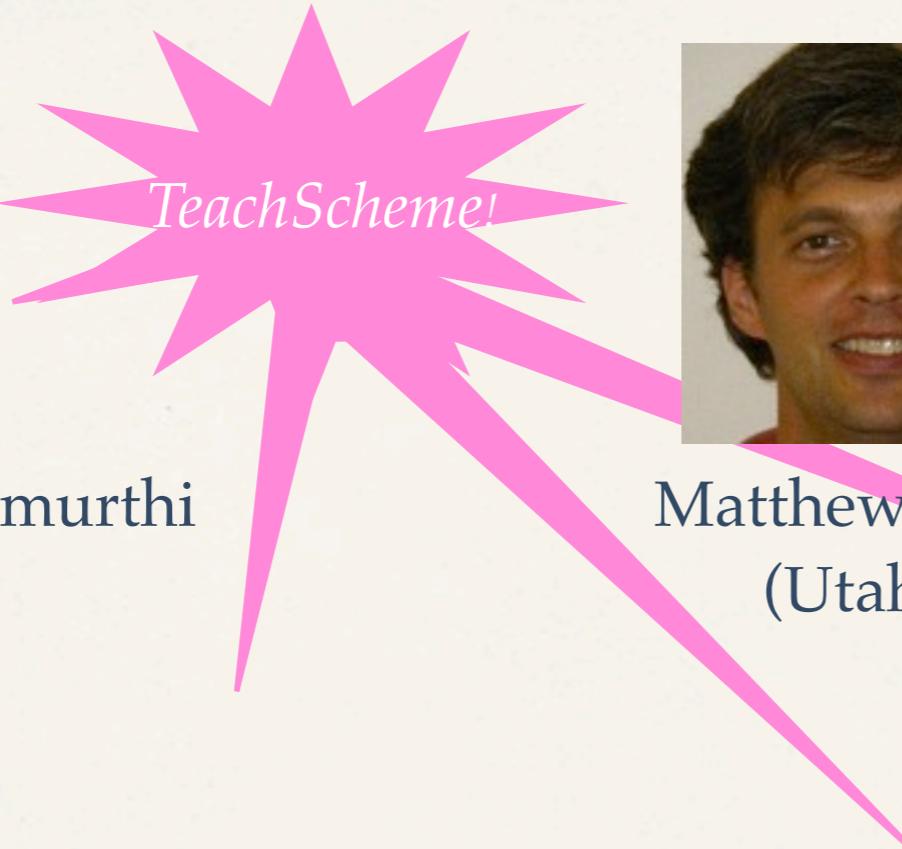
- ~ 15 years of outreach
- ~ 10 sites
- ~ 100ish research papers
- ~ 1,000ish teachers trained
- ~ 10,000s students in reach
- ~ 1,000,000 downloads (unique IP)

The *Teach Scheme, NOT* Project

January 26, 1995



Shriram Krishnamurthi
(Brown)



Matthew Flatt
(Utah)

- ~ 15 years of outreach
- ~ 10 sites
- ~ 100ish research papers
- ~ 1,000ish teachers trained
- ~ 10,000s students in reach
- ~ 1,000,000 downloads (unique IP)

The *Teach Scheme, NOT* Project

January 26, 1995



Shriram Krishnamurthi
(Brown)



Matthew Flatt
(Utah)



Robby Findler
(Northwestern)

- ~ 15 years of outreach
- ~ 10 sites
- ~ 100ish research papers
- ~ 1,000ish teachers trained
- ~ 10,000s students in reach
- ~ 1,000,000 downloads (unique IP)

The *Teach Scheme, NOT* Project

January 26, 1995



Shriram Krishnamurthi
(Brown)



Matthew Flatt
(Utah)



Robby Findler
(Northwestern)



Kathi Fisler
(WPI)

~ 15 years of outreach
~ 10 sites
~ 100ish research papers
~ 1,000ish teachers trained
~ 10,000s students in reach
~ 1,000,000 downloads (unique IP)

The *TeachScheme!* Idea

typical CS courses in high school:
20, 40, perhaps 80 students

typical high school:
1,000 or 2,000 or 3,000
and more students



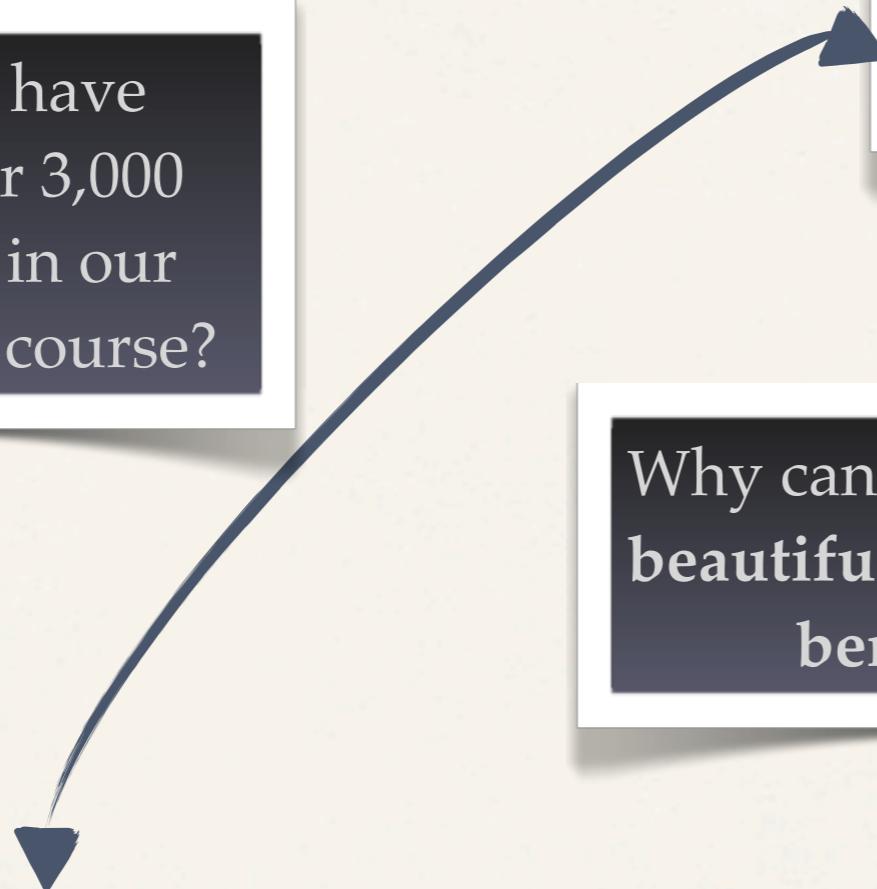
The *TeachScheme!* Idea

Why can't we have
1,000 or 2,000 or 3,000
or **all students** in our
introductory CS course?

typical high school:
1,000 or 2,000 or 3,000
and more students

Why can't we show them how
beautiful CS is and have them
benefit from it all?

typical CS courses in high school:
20, 40, perhaps 80 students



The *TeachScheme!* Idea

Observation: Teaching novices how to program in a professional language does not do computer science justice. Indeed, what we have seen in the last few decades is that it drives students away.

The *TeachScheme!* Idea

Observation: Teaching novices how to program in a professional language does not do computer science justice. Indeed, what we have seen in the last few decades is that it drives students away.

Idea: Teaching the ideas of programming with mathematics demonstrates the depth, breadth & beauty of computing. It also aligns our field with one of the three important R's in the education world.

The *TeachScheme!* Idea

Idea: Teaching the ideas of programming with mathematics demonstrates the depth, breadth & beauty of computing. It also aligns our field with one of the three important R's in the education world.

Synthesis: We can start with one and smoothly reach the other.

Observation: Teaching novices how to program in a professional language does not do computer science justice. Indeed, what we have seen in the last few decades is that it drives students away.

We can!

(1) program with mathematics

We can!

(1) program with mathematics

(2) provide a smooth path all the way to “real” programming

We can!

(1) program with mathematics

(2) provide a smooth path all the way to “real” programming

(3) introduce systematic design (comp prob solving) along the way

We can!

(1) program with mathematics

(2) provide a smooth path all the way to “real” programming

lessons learned

(3) introduce systematic design (comp prob solving) along the way

We can!

(1) program with mathematics

(2) provide a smooth path all the way to “real” programming

lessons learned

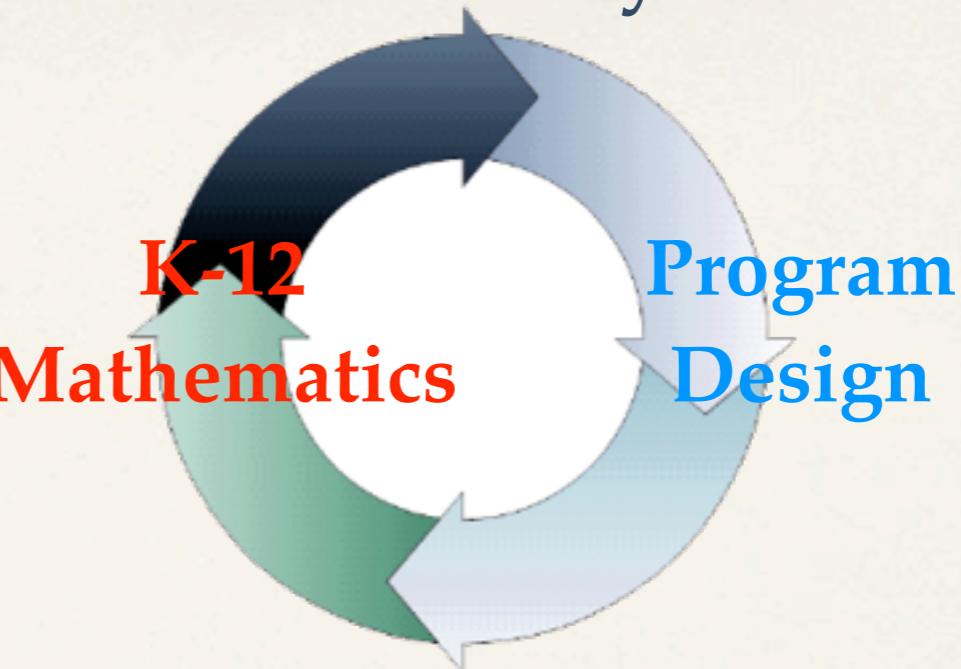
(3) introduce systematic design (comp prob solving) along the way



failures &
challenges

We can!

A virtuous cycle



(1) program with mathematics

(2) provide a smooth path all the way to “real” programming

lessons learned

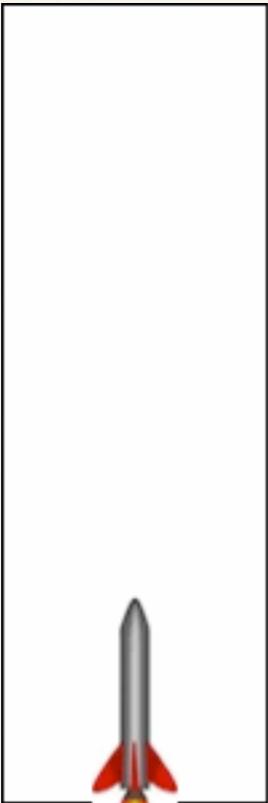
(3) introduce systematic design (comp prob solving) along the way

↓
failures &
challenges

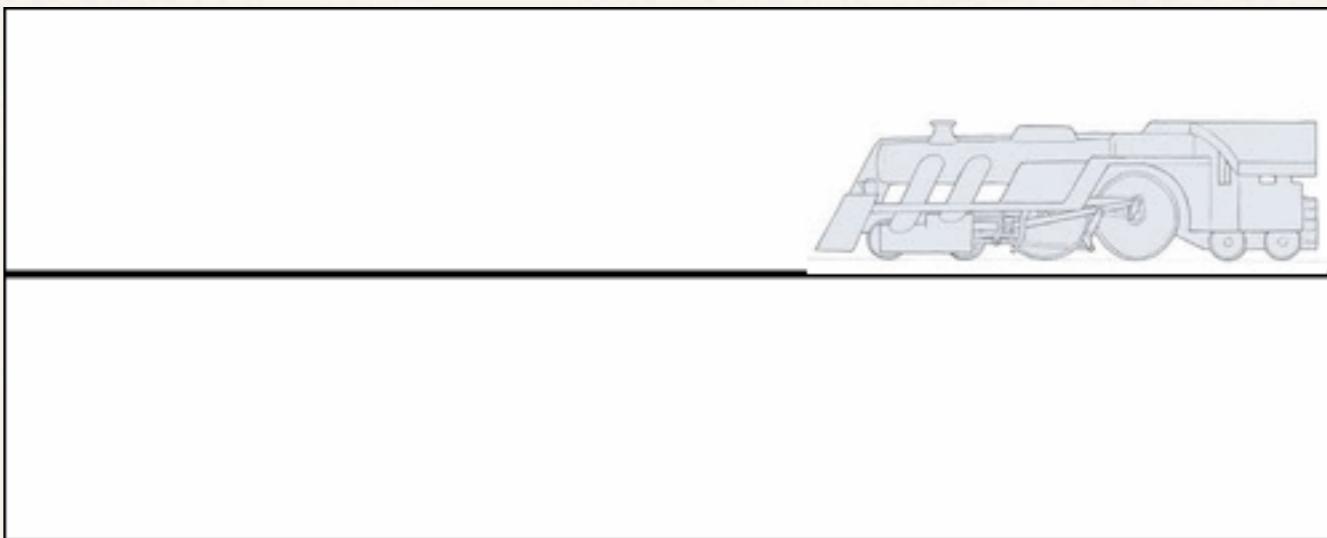
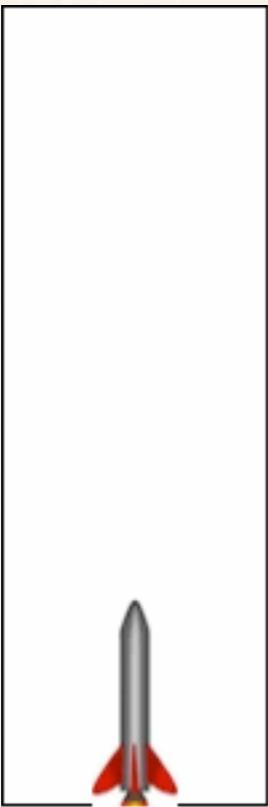
Programming with Mathematics for Fun

“Hello World” -- with mathematics

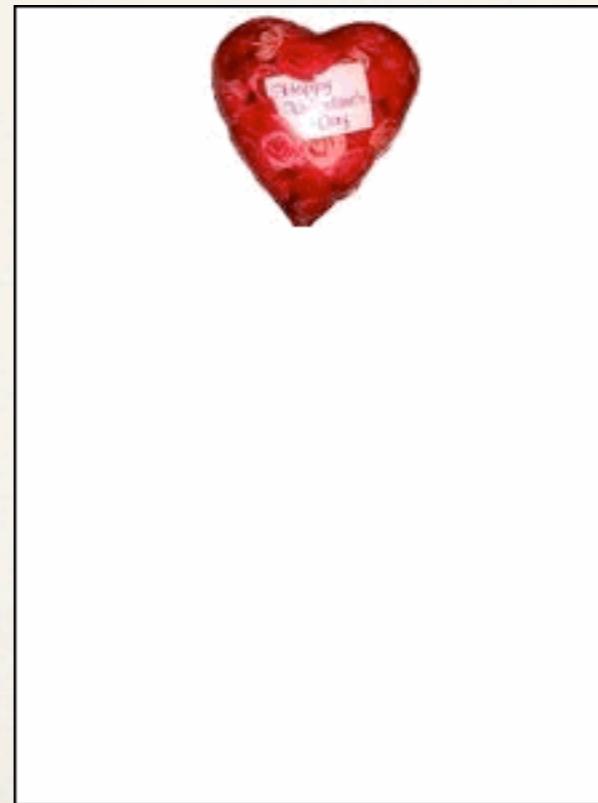
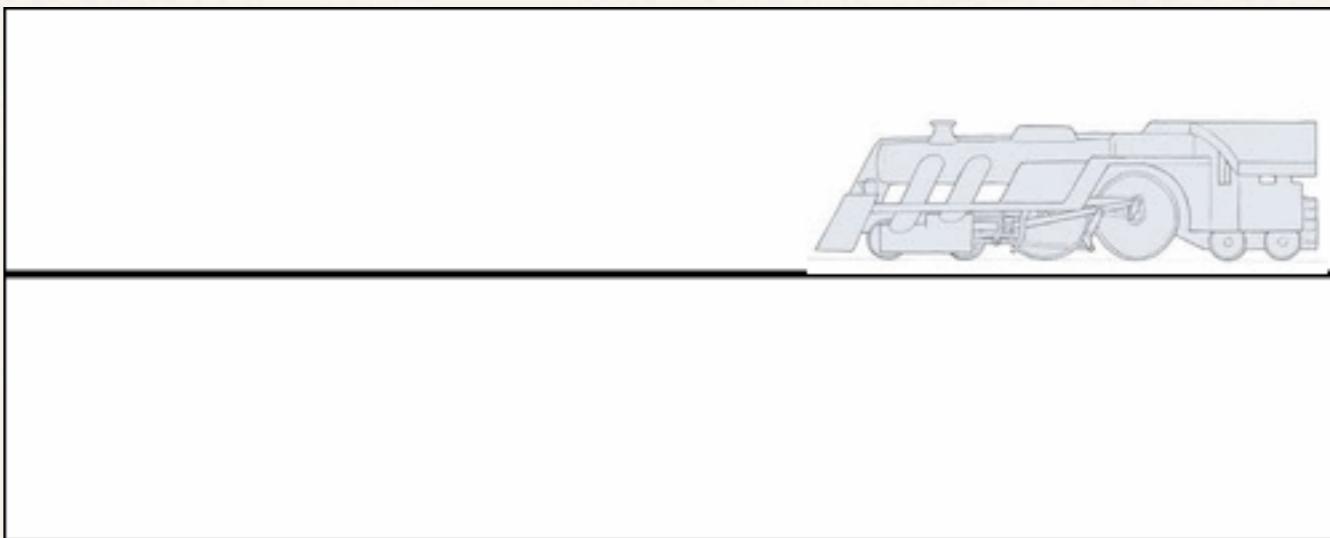
“Hello World” -- with mathematics



“Hello World” -- with mathematics



“Hello World” -- with mathematics



What do you need to get there?

Standard math problems,
which students hate

Problem 1: A rocket lifts off from its launchpad at the constant speed of 7.5 ft per second. What height does it reach after 0 seconds, 1 second, 2 seconds, 3 seconds, t seconds?

Problem 2: A train leaves Dallas for Chicago and travels at the constant speed of 65 miles per hour. How far does it get in 0 hours, 1 hour, 1.5 hours, 2 hours, h hours?

Problem 3: A balloon drops from the ceiling of a convention center at the constant speed of 15 feet per second. How far does it drop in 0 seconds, 1 second, 2 seconds, h seconds?

What do you need to get there?

Tables, which students know

Problem 1: A rocket lifts off from its launchpad at the constant speed of 7.5 ft per second. What height does it reach after 0 seconds, 1 second, 2 seconds, 3 seconds, t seconds?

time	0	1	2	3	4	... t ...
height	0	7.5	15.0	22.5	30.0	$y = 7.5 * t$

What do you need to get there?

Images, which aren't a problem

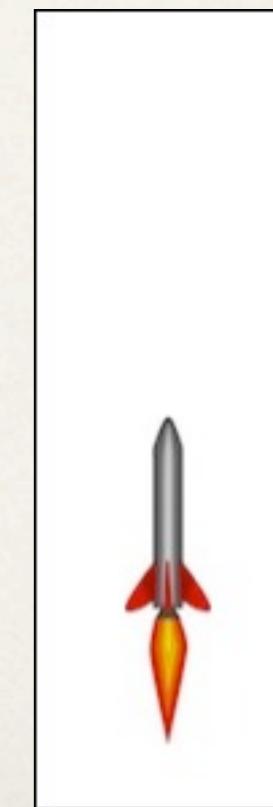
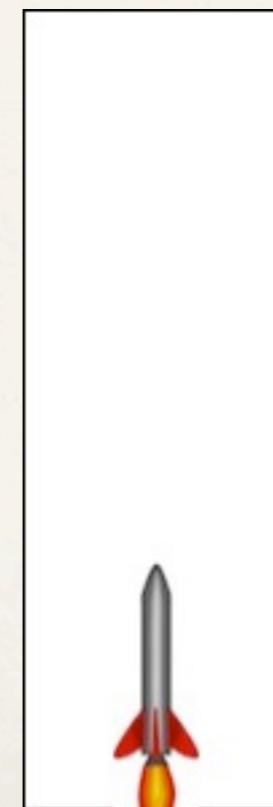
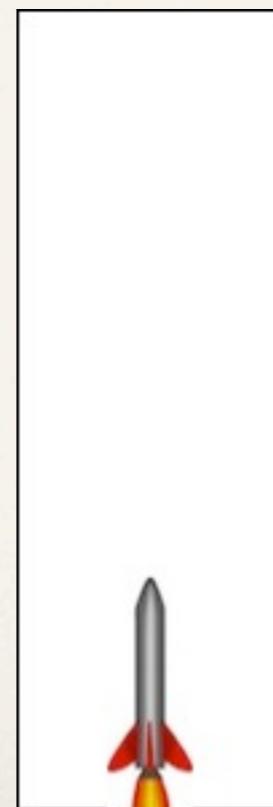
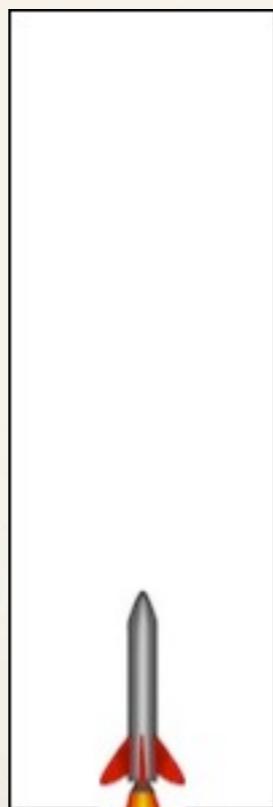
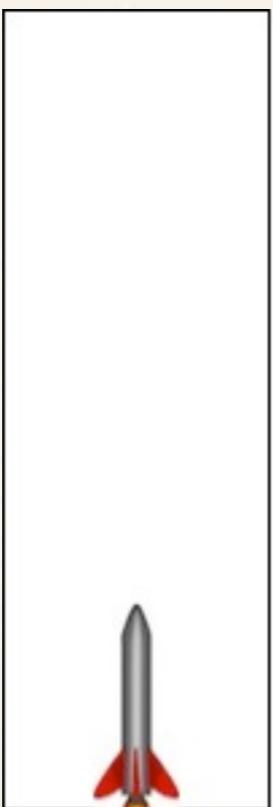
Problem 1: A rocket lifts off from its launchpad at the constant speed of 7.5 ft per second. What height does it reach after 0 seconds, 1 second, 2 seconds, 3 seconds, t seconds?

time	0	1	2	3	4	... t ...
height	0	7.5	15.0	22.5	30.0	$y = 7.5 * t$
sketch it						

The table shows the relationship between time and height for the rocket's ascent. The height increases linearly over time, starting at 0 ft and reaching 7.5 ft at 1 second, 15.0 ft at 2 seconds, 22.5 ft at 3 seconds, and 30.0 ft at 4 seconds. The general formula for height is given as $y = 7.5 * t$. Below the table, six boxes provide space for students to draw the rocket's position at each time interval, with the final box being a placeholder for time t .

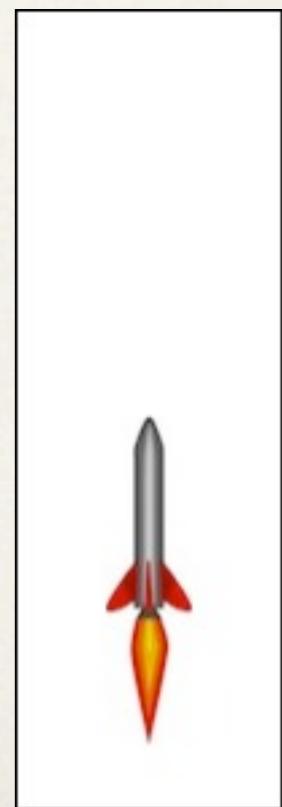
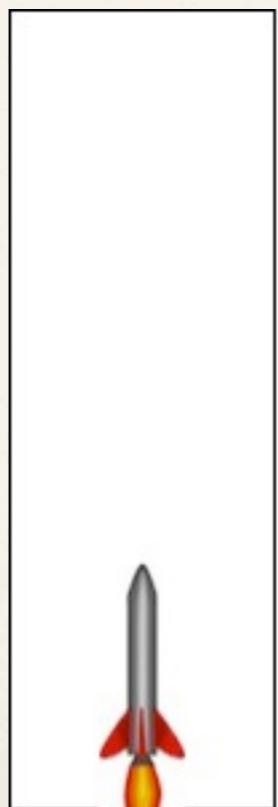
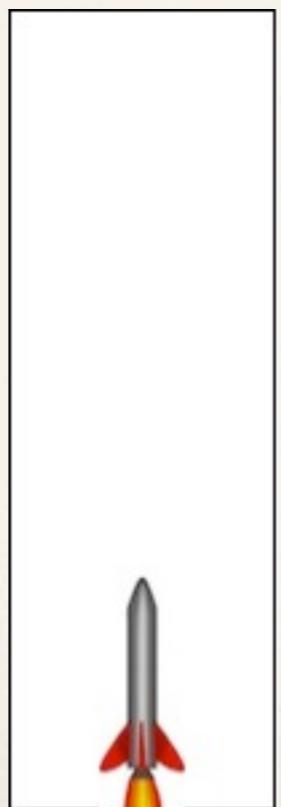
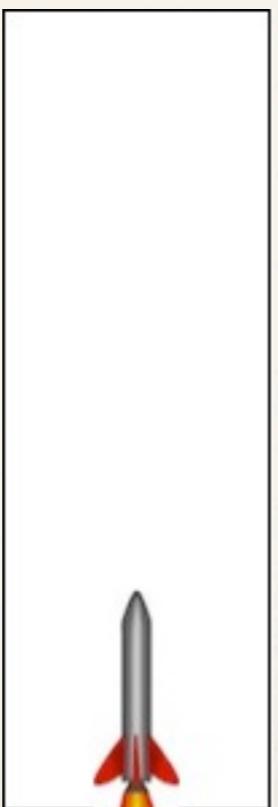
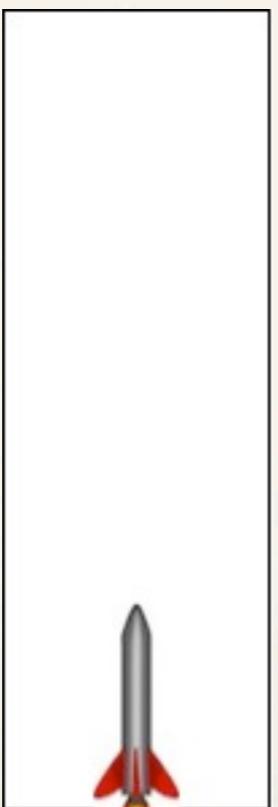
What do you need to get there?

Imagination: what happens if you display
~30 images per second of this series?



What do you need to get there?

Imagination: what happens if you display
~30 images per second of this series?



How do you get images into math?

The arithmetic of images:
images are like numbers

$$1 + 1 =$$

How do you get images into math?

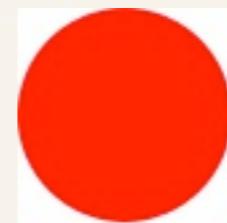
The arithmetic of images:
images are like numbers

$$1 + 1 = 2$$

How do you get images into math?

The arithmetic of images:
images are like numbers

$$1 + 1 = 2$$



overlay

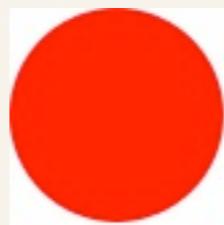


=

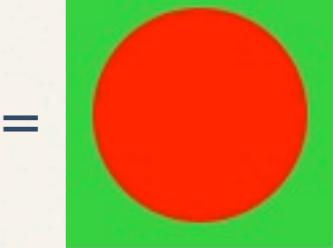
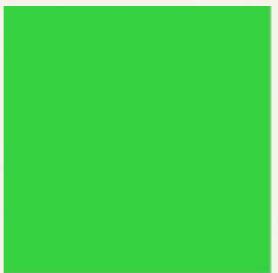
How do you get images into math?

The arithmetic of images:
images are like numbers

$$1 + 1 = 2$$



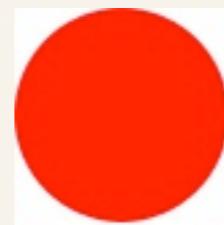
overlay



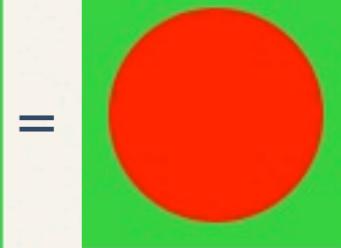
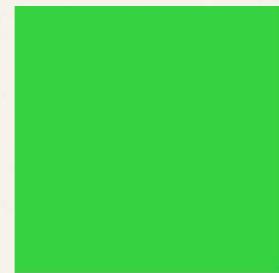
How do you get images into math?

The arithmetic of images:
images are like numbers

$$1 + 1 = 2$$



overlay

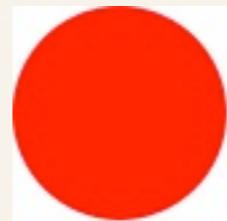


$$2 * (1 + 1) = 4$$

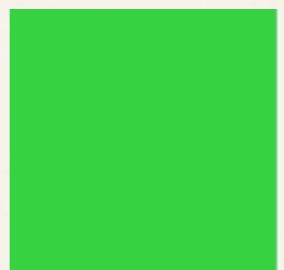
How do you get images into math?

The arithmetic of images:
images are like numbers

$$1 + 1 = 2$$



overlay



$$2 * (1 + 1) = 4$$



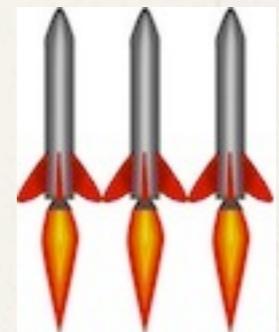
besides (



besides



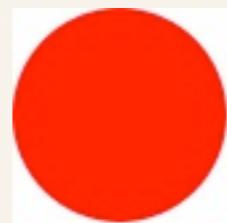
) =



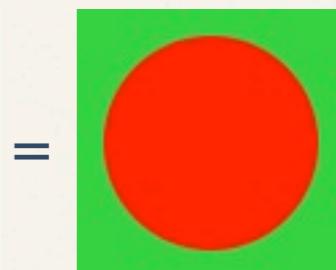
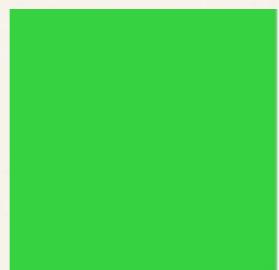
How do you get images into math?

The arithmetic of images:
images are like numbers

$$1 + 1 = 2$$



overlay



$$2 * (1 + 1) = 4$$



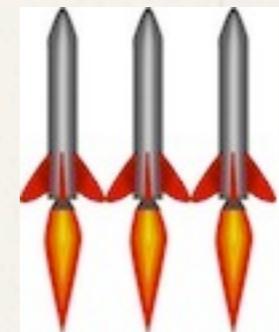
besides (



besides



) =

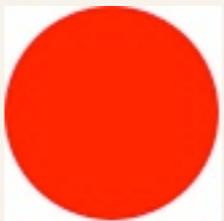


$$\sin 90 = 1$$

How do you get images into math?

The arithmetic of images:
images are like numbers

$$1 + 1 = 2$$



overlay



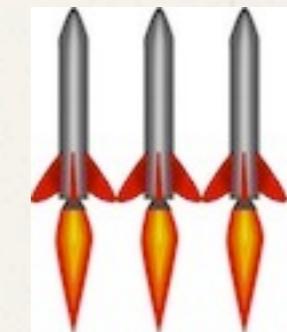
$$2 * (1 + 1) = 4$$



besides (



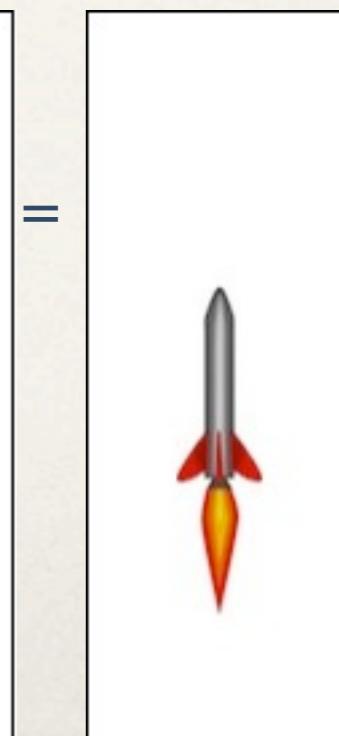
besides)



$$\sin 90 = 1$$



place at (50, 80) in



How do you get images into math: with DrRacket

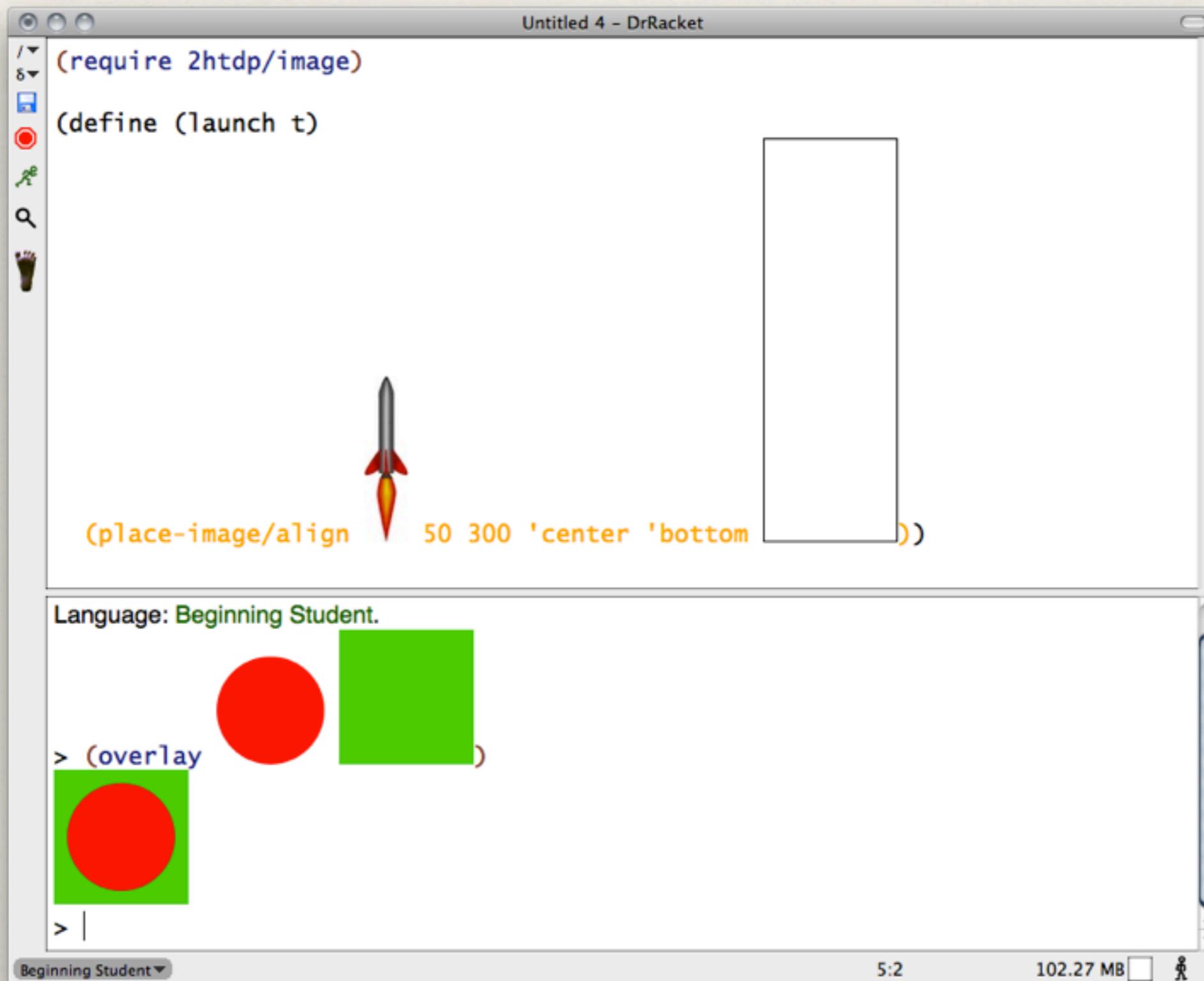
Untitled 4 – DrRacket

```
(require 2htdp/image)
(define (launch t)
  (place-image/align
    (image/rocket)
    50 300 'center 'bottom
    (empty-scene 400 300)))
```

Language: Beginning Student.

```
> (overlay
  (circle 100 "solid" "red")
  (square 100 "solid" "green"))
```

Beginning Student ▾ 5:2 102.27 MB

A screenshot of the DrRacket IDE. The top part shows racket code: '(require 2htdp/image)', '(define (launch t)', and '(place-image/align (image/rocket) 50 300 'center 'bottom (empty-scene 400 300))).'. The bottom part shows the output of the code: a red circle overlaid on a green square. The interface includes toolbars, a status bar at the bottom, and a vertical scroll bar on the right.

How do you get images into math: with DrRacket

Untitled 4 – DrRacket

```
(require 2htdp/image)
(define (launch t)
  (place-image/align 50 300 'center 'bottom))
Lesson 1: Your PL/IDE must support
an arithmetic of images.
```

Language: Beginning Student.

```
> (overlay
  (circle 100 "red")
  (square 100 "green"))
```

Beginning Student ▾ 5:2 102.27 MB ⌂ ⌂

A screenshot of the DrRacket IDE. The top part shows a code editor with Racket code: '(require 2htdp/image)' and '(define (launch t) (place-image/align 50 300 'center 'bottom))'. A large callout box covers the middle of the screen, containing the text 'Lesson 1: Your PL/IDE must support an arithmetic of images.' Below the code editor, the language is set to 'Beginning Student'. In the interaction area, the command '> (overlay (circle 100 "red") (square 100 "green"))' is entered, and its result is displayed as a red circle on top of a green square. The bottom status bar shows 'Beginning Student ▾ 5:2 102.27 MB ⌂ ⌂'.

How do you get images into math?

Now add the usual bit of algebra,
a plain function definition

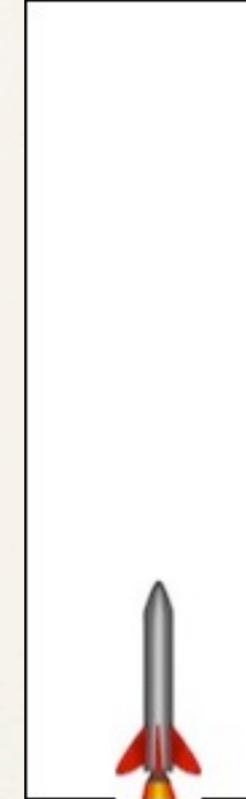
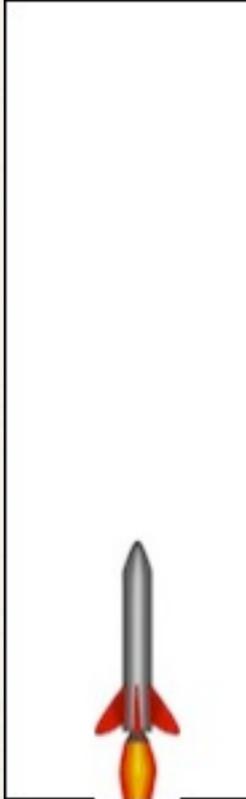
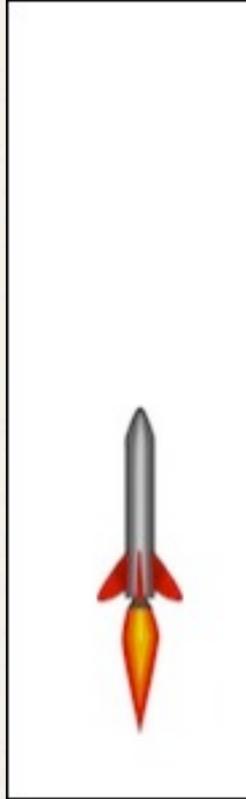
launch(t) = place



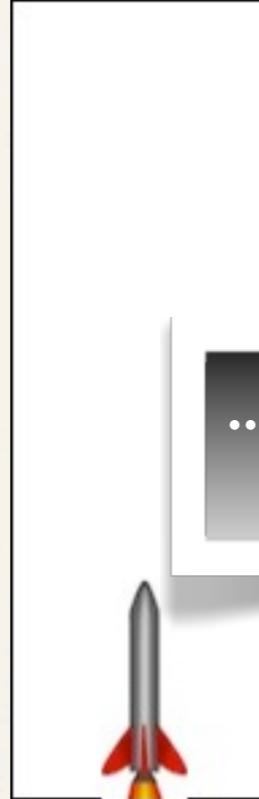
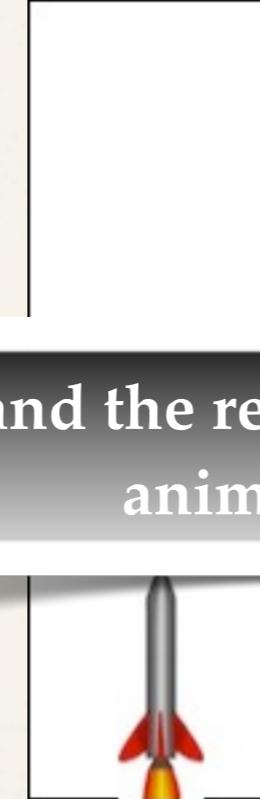
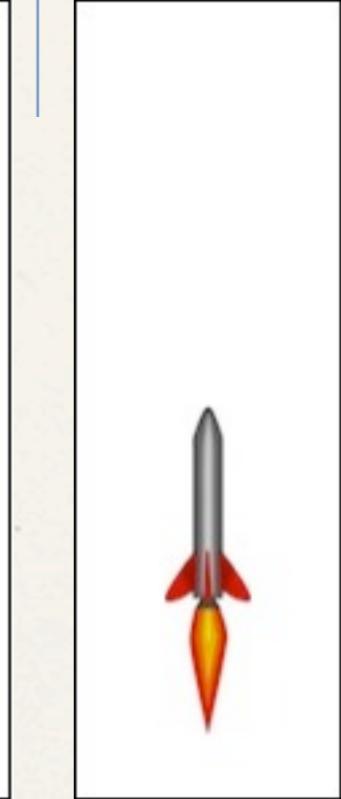
*at (50, 7.5 * t) in*



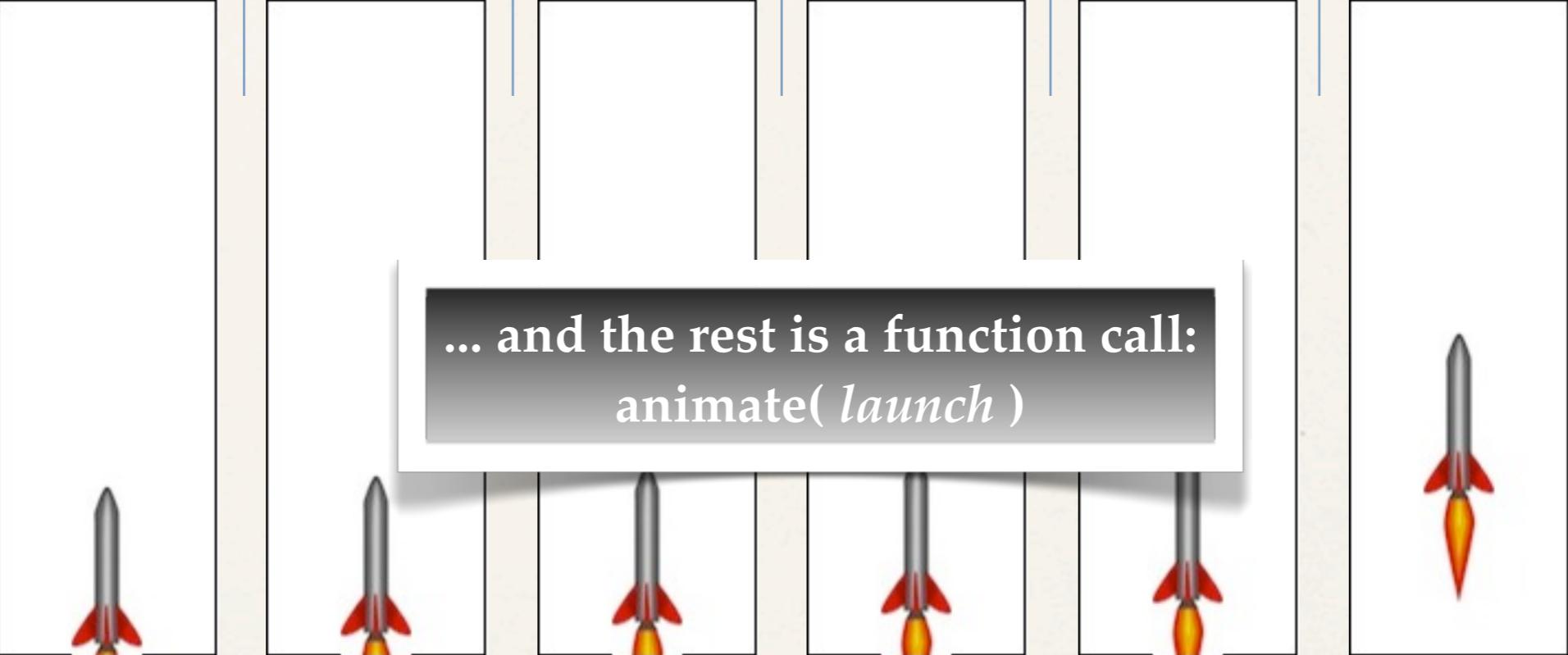
How do you get images into math?

time	0	1	2	3	4	... t ...
launch	$launch(0)$	$launch(1)$	$launch(2)$	$launch(3)$	$launch(4)$	
						

How do you get images into math?

time	0	1	2	3	4	... t ...
launch	<code>launch(0)</code>	<code>launch(1)</code>	<code>launch(2)</code>	<code>launch(3)</code>	<code>launch(4)</code>	
						

**... and the rest is a function call:
`animate(launch)`**



The diagram illustrates a sequence of six rocket frames, each enclosed in a white rectangular frame. The rockets are shown in various stages of ascent: at frame 0, the rocket is small and grey; by frame 1, it has a red base and a yellow flame; by frame 2, the body is elongated; by frame 3, the fins are visible; and by frame 4, the rocket is fully elongated with a large red base and a prominent yellow flame. A central callout box contains the text "... and the rest is a function call: animate(launch)".

How do you get images into math?

some more detail:
describe the world and its actions

```
(define (animate stateToImage)
  (big-bang 0
    [on-tick add1]
    [to-draw stateToImage] ))
```

How do you get images into math?

some more detail:
describe the world and its actions

```
(define (animate stateToImage)
  (big-bang 0
    [on-tick add1]
    [to-draw stateToImage] ))
```

animate (*stateToImage*) is:

(1) initial state of the world is 0

$\text{state}[0] = 0$

(2) every time the clock ticks, add 1 to the state of the world

$\text{state}[t+1] = \text{state}[t] + 1$

(3) after every event, render the state of the world

$\text{theImage} = \text{stateToImage}(\text{state}[t])$

How do you deal with *keyboard* and *mouse events* in math?

```
(define (game state0)
  (big-bang state0
    [on-tick time-handler]
    [on-key key-handler]
    [to-draw image-renderer] ))

;; World KeyEvent -> World
(define (key-handler current key)
  ...)
```

a video game is:

- (1) initial state of the world is *something*
- (2) every time the clock ticks,
compute the next state from the current state
- (3) every time a key event happens,
compute the next state from the current state & key event
- (4) every time a mouse event takes place,
compute the next state from the current state & mouse event
- (5) after every event: *compute an image from the current state*

Do kids find this *exciting*? Do they *benefit* from it?

Yes! In a short time, kids write
80s-style video games *with middle
school mathematics as PL.*

Do kids find this *exciting*? Do they *benefit* from it?

Yes! In a short time, kids write
80s-style video games *with middle
school mathematics as PL.*

$$\text{sign}(x) = \begin{cases} +1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

*Kids ask for more mathematics:
conditional functions, geometry,
trigonometry, pre-calculus
concepts, and many more.*

Do kids find this *exciting*? Do they *benefit* from it?

Yes! In a short time, kids write
80s-style video games *with middle
school mathematics as PL.*

$$\text{sign}(x) = \begin{cases} +1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

*Kids ask for more mathematics:
conditional functions, geometry,
trigonometry, pre-calculus
concepts, and many more.*

Their *performance* on all kinds of
mathematics exams improves.

And many want more programming!

And many want more programming!

- structures
- vectors
- unions
- lists
- graphs
- trees

And many want more programming!

and yes,
function-consuming and
function-producing functions

- structures
- vectors
- unions
- lists
- graphs
- trees

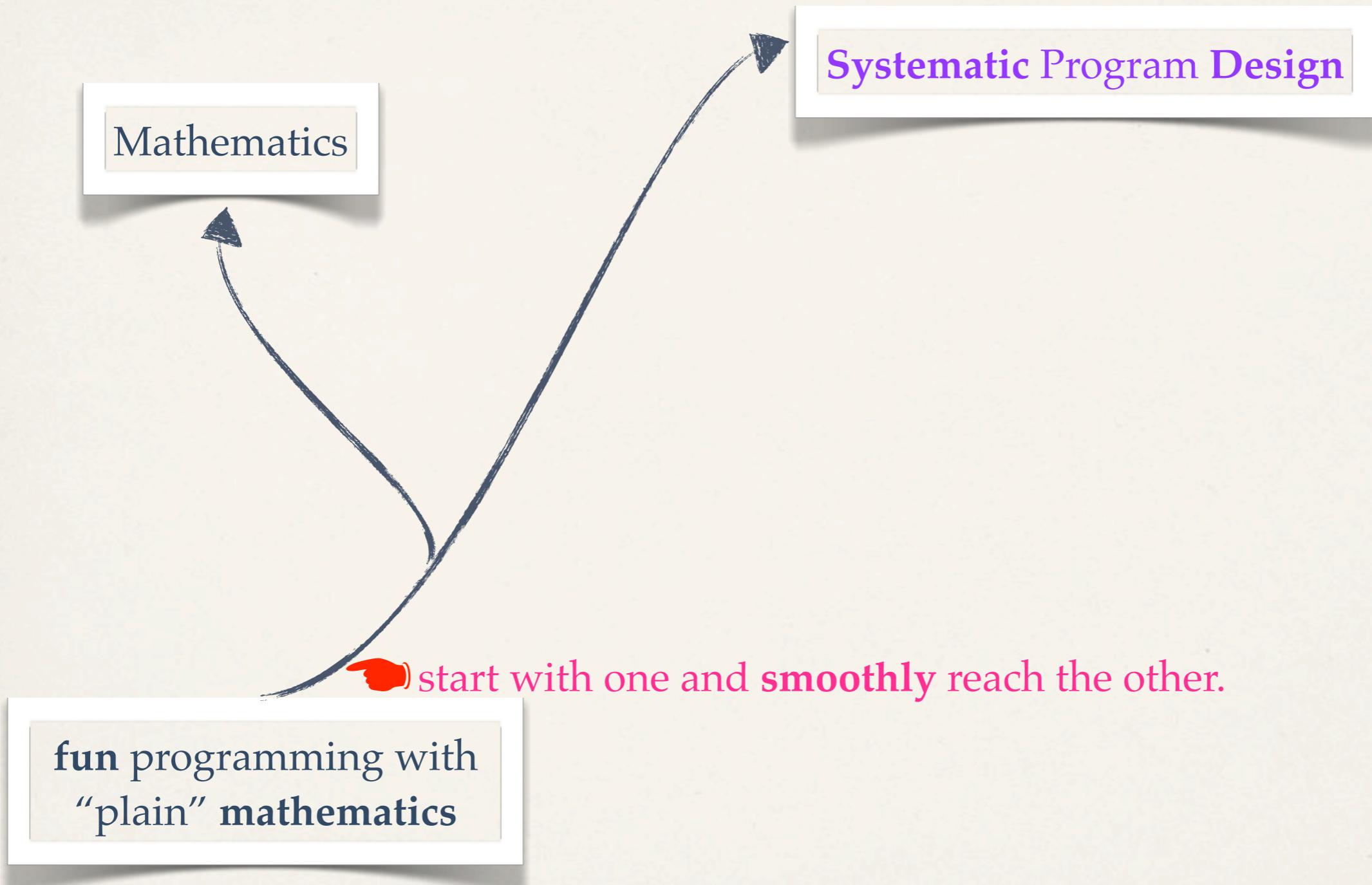
And many want more programming!

and yes,
function-consuming and
function-producing functions

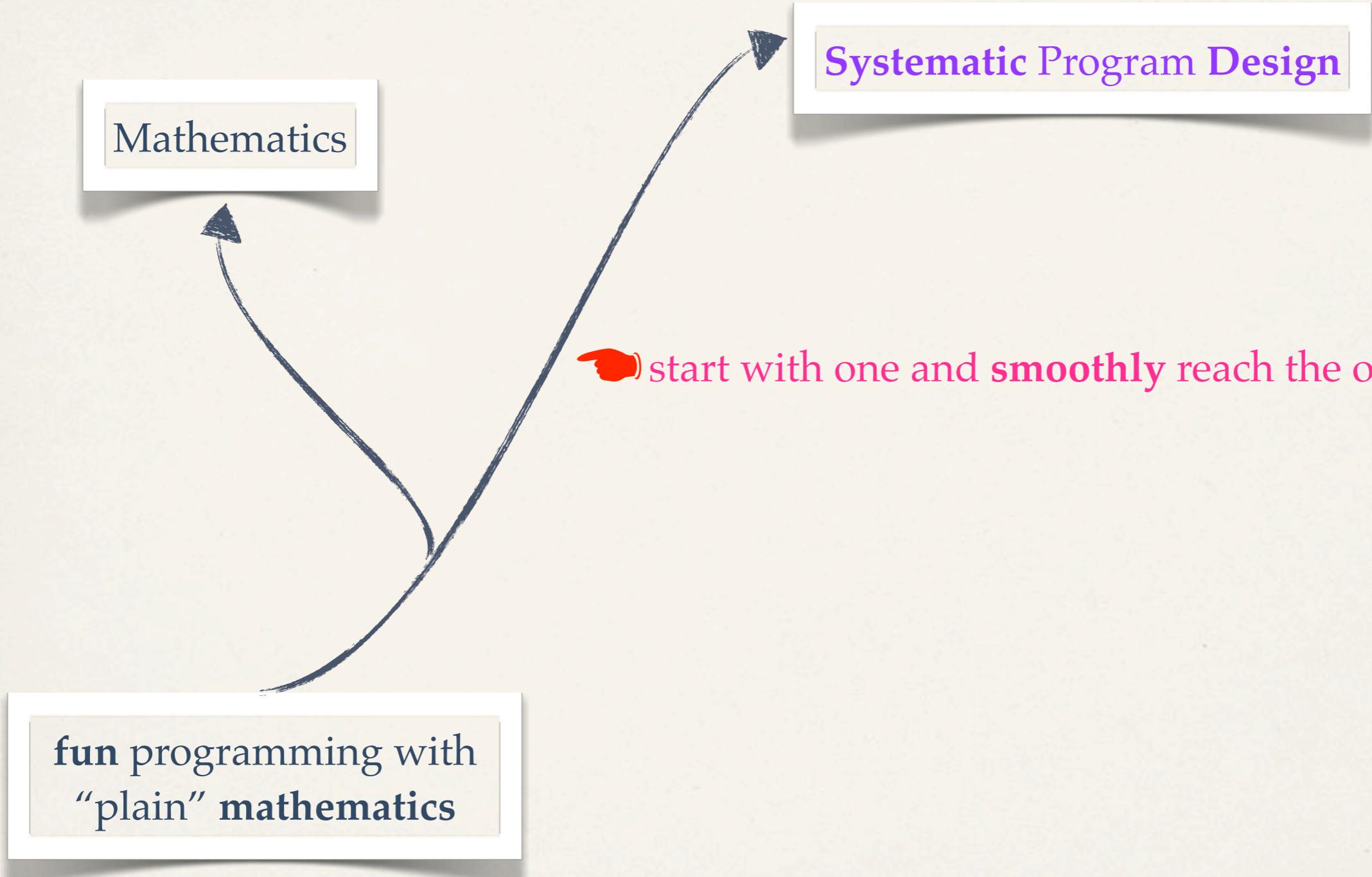
-- structures
-- vectors
-- unions
-- lists
-- graphs
-- trees

plus
something like modules,
classes, objects, or other
organizational mechanisms

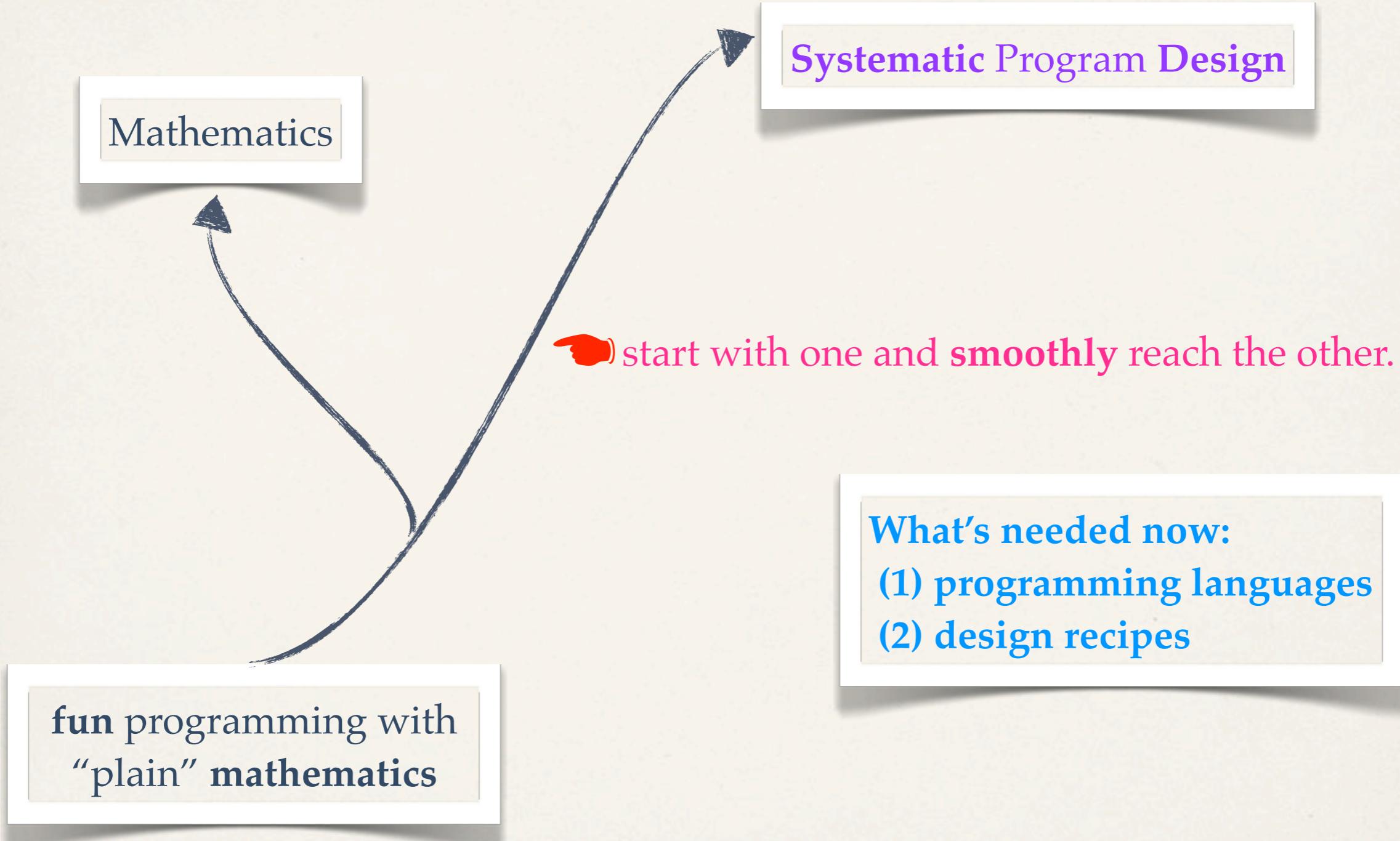
We got them hooked!



We got them hooked!



We got them hooked!



What's needed now:
(1) programming languages
(2) design recipes

Programming Languages Matter

Programming in ‘off-the-shelf’ languages

The year is 1995, Houston.

Michael Hunt & Karen North’s classrooms

C++ is the language *du jour*.

Programming in ‘off-the-shelf’ languages

The year is 1995, Houston.

Michael Hunt & Karen North’s classrooms

C++ is the language *du jour*.

```
void main() {  
    double price_per_slice = 21.95 / 12;  
    double total_due      = 0.0;  
    int number_of_slices  = 0;  
  
    cout << "how many slices did you eat?" << endl;  
    cin >> number_of_slices;  
    price_per_slice * number_of_slices = total_due;  
    cout << "your share is " << total_due << endl;  
}
```

Programming in ‘off-the-shelf’ languages

The year is 1995, Alief @ Houston.
Karen North is the host.
C++ is the language *du jour*.

```
int main() {  
    double price_per_slice = 21.95 / 12;  
    double total_due      = 0.0;  
    int number_of_slices  = 0;  
  
    cout << "how many slices did you eat?" << endl;  
    cin >> number_of_slices;  
    price_per_slice * number_of_slices = total_due;  
    cout << "your share is " << total_due << endl;  
}
```

Programming in ‘off-the-shelf’ languages

The year is 1995, Alief @ Houston.
Karen North is the host.
C++ is the language *du jour*.

```
int main() {  
    double price_per_slice = 21.95 / 12;  
    double total_due      = 0.0;  
    int number_of_slices  = 0;  
  
    cout << "how many slices did you eat?" << endl;  
    cin >> number_of_slices;  
    price_per_slice * number_of_slices = total_due;  
    cout << "your share is " << total_due << endl;  
}
```

ivalue expected here

Programming in ‘off-the-shelf’ languages

The year is 1995, Rice @ Houston.
Scheme is my language *of hope*.

```
(define (how-many a-list)
  (cond
    [ (empty? a-list) 0]
    [else add1 (how-many (rest a-list))]))
```

```
> (how-many empty)  
0
```

Good!

Programming in ‘off-the-shelf’ languages

The year is 1995, Rice @ Houston.
Scheme is my language *of hope*.

```
(define (how-many a-list)
  (cond
    [ (empty? a-list) 0]
    [else add1 (how-many (rest a-list))]))
```

```
> (how-many empty)
```

```
0
```

Good!

```
> (how-many `(sigcse))
```

```
0
```

Programming in ‘off-the-shelf’ languages

The year is 1995, Rice @ Houston.
Scheme is my language *of hope*.

```
(define (how-many a-list)
  (cond
    [ (empty? a-list) 0]
    [else add1 (how-many (rest a-list))]))
```

```
> (how-many empty)  
0
```

Good!

```
> (how-many `(sigcse))  
0
```

Huh?

Programming in ‘off-the-shelf’ languages

The year is 1995, Rice @ Houston.
Scheme is my language *of hope*.

```
(define (how-many a-list)
  (cond
    [ (empty? a-list) 0]
    [else add1 (how-many (rest a-list))]))
```

```
> (how-many empty)  
0
```

Good!

```
> (how-many `(sigcse))  
0
```

Huh?

```
> (how-many `(sigcse is in dallas))  
0
```

Programming in ‘off-the-shelf’ languages

The year is 1995, Rice @ Houston.
Scheme is my language *of hope*.

```
(define (how-many a-list)
  (cond
    [ (empty? a-list) 0]
    [else add1 (how-many (rest a-list))]))
```

```
> (how-many empty)  
0
```

Good!

```
> (how-many `(sigcse))  
0
```

Huh?

```
> (how-many `(sigcse is in dallas))  
0
```

Wrong!

Programming in ‘off-the-shelf’ languages

The year is 1995, Rice @ Houston.
Scheme is my language *of hope*.

```
(define (how-many a-list)
  (cond
    [ (empty? a-list) 0]
    [else add1 (how-many (rest a-list))]))
```

Omitting parentheses
produces a *syntactically correct* program with a
totally different meaning
than intended (begin).

Programming in 'off-the-shelf' languages

The year is Anytime, Anyplace @ Anywhere.
Your choice of the day is some professional
language X for X in

- Basic
- Java
- JavaScript
- Python
- Ruby on Rails

Programming in 'off-the-shelf' languages

The year is Anytime, Anyplace @ Anywhere.
Your choice of the day is some professional
language X for X in
Basic
Java
JavaScript
Python
Ruby on Rails

Problem: You will inevitably run into similar
problems with syntax / error reporting.

Programming in ‘off-the-shelf’ languages

The year is Anytime, Anyplace @ Anywhere.
Your choice of the day is some professional
language X for X in
Basic
Java
JavaScript
Python
Ruby on Rails

Problem: You will inevitably run into similar problems with syntax / error reporting.

Solution: You and your colleagues specify and use only a small portion of the language to keep your students safe.

Programming in a teaching language

The *Real* Problem: Your students don't stick to the chosen subset of the language.

Programming in a teaching language

The *Real* Problem: Your students don't stick to the chosen subset of the language.

Lesson 2a: If you specify and define a sublanguage, implement it and exploit the restrictions in the parser.

Programming in a teaching language

The parser of a professional language
reports errors with messages that assume
a *knowledgable programmer*, someone
who knows *the whole language*.

Programming in a teaching language

The parser of a professional language reports errors with messages that assume a *knowledgable programmer*, someone who knows *the whole language*.

Novices make mistakes,
and by definition they don't
know the whole language.

Programming in a teaching language

The parser of a professional language reports errors with messages that assume a *knowledgable programmer*, someone who knows *the whole language*.

Novices make mistakes, and by definition they don't know the whole language.

The parser of a *teaching language* reports errors with messages that assume a *knowledgable novice*, someone who knows *a small part of the language*.

Programming in a teaching language

```
int main() {  
    double price_per_slice = 21.95 / 12;  
    double total_due      = 0.0;  
    int number_of_slices  = 0;  
  
    cout << "how many slices did you eat?" << endl;  
    cin >> number_of_slices;  
    price_per_slice * number_of_slices = total_due;  
    cout << "your share is " << total_due << endl;  
}
```

An assignment statement
must have a variable on
the left-hand side.

Programming in a teaching language

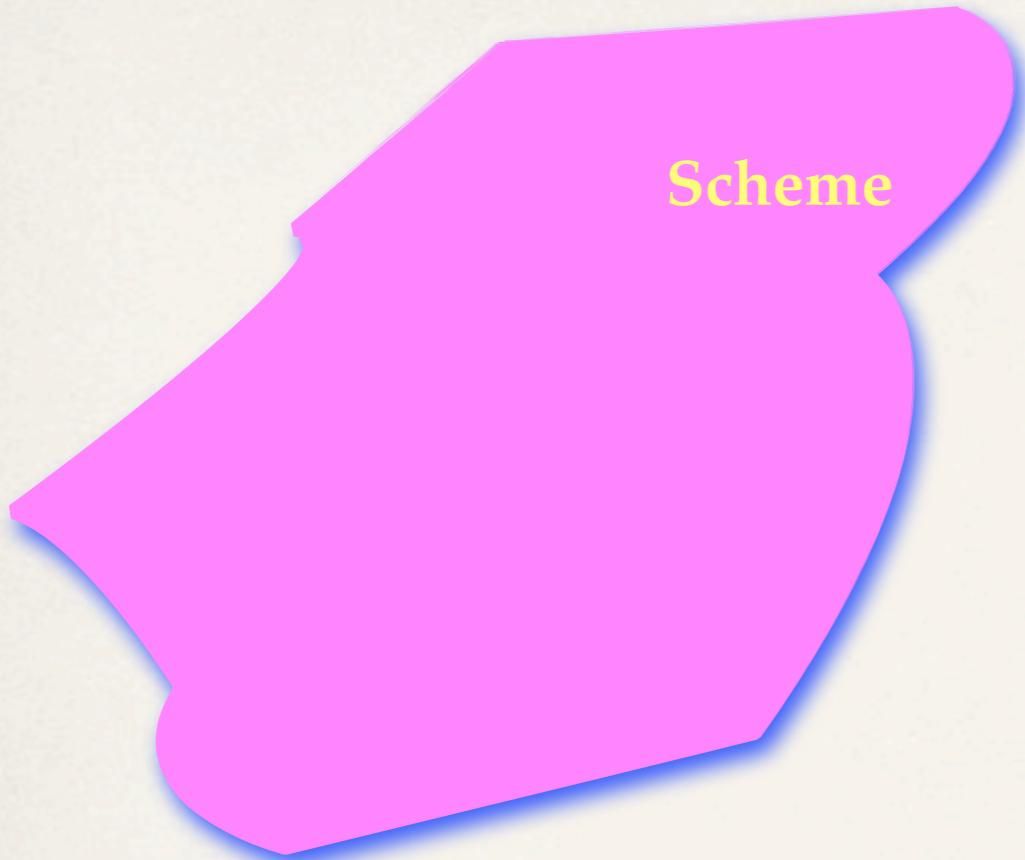
```
int main() {  
    double price_per_slice = 21.95 / 12;  
    double total_due      = 0.0;  
    int number_of_slices = 0;  
  
    cout << "how many slices did you eat?" << endl;  
    cin >> number_of_slices;  
    price_per_slice * number_of_slices = total_due;  
    cout << "your share is " << total_due << endl;  
}
```

An assignment statement
must have a variable on
the left-hand side.

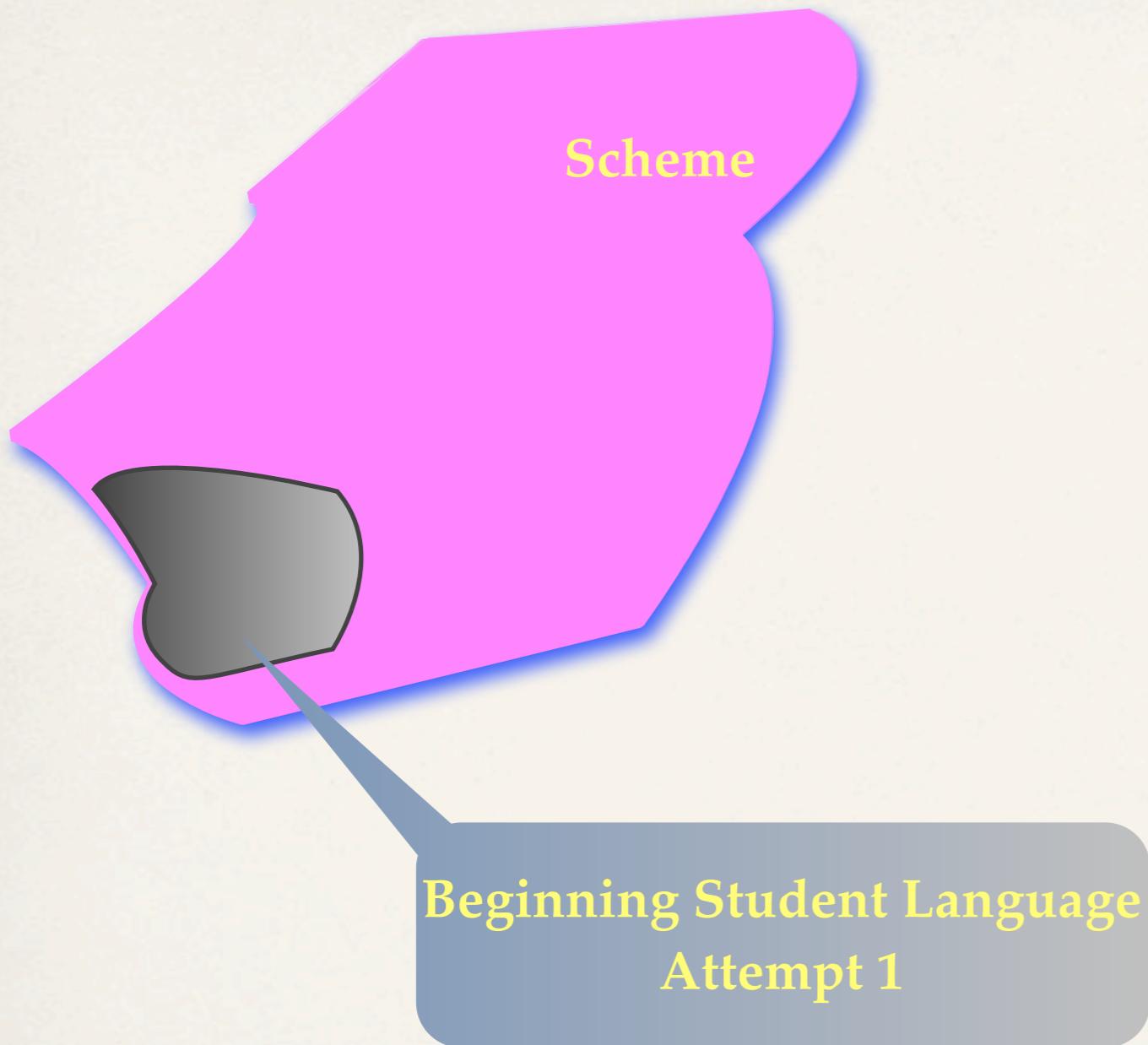
The branch of a conditional
must consist of one expression.

```
(define (how-many a-list)  
  (cond  
    [ (empty? a-list) 0]  
    [else add1 (how-many (rest a-list)) ])))
```

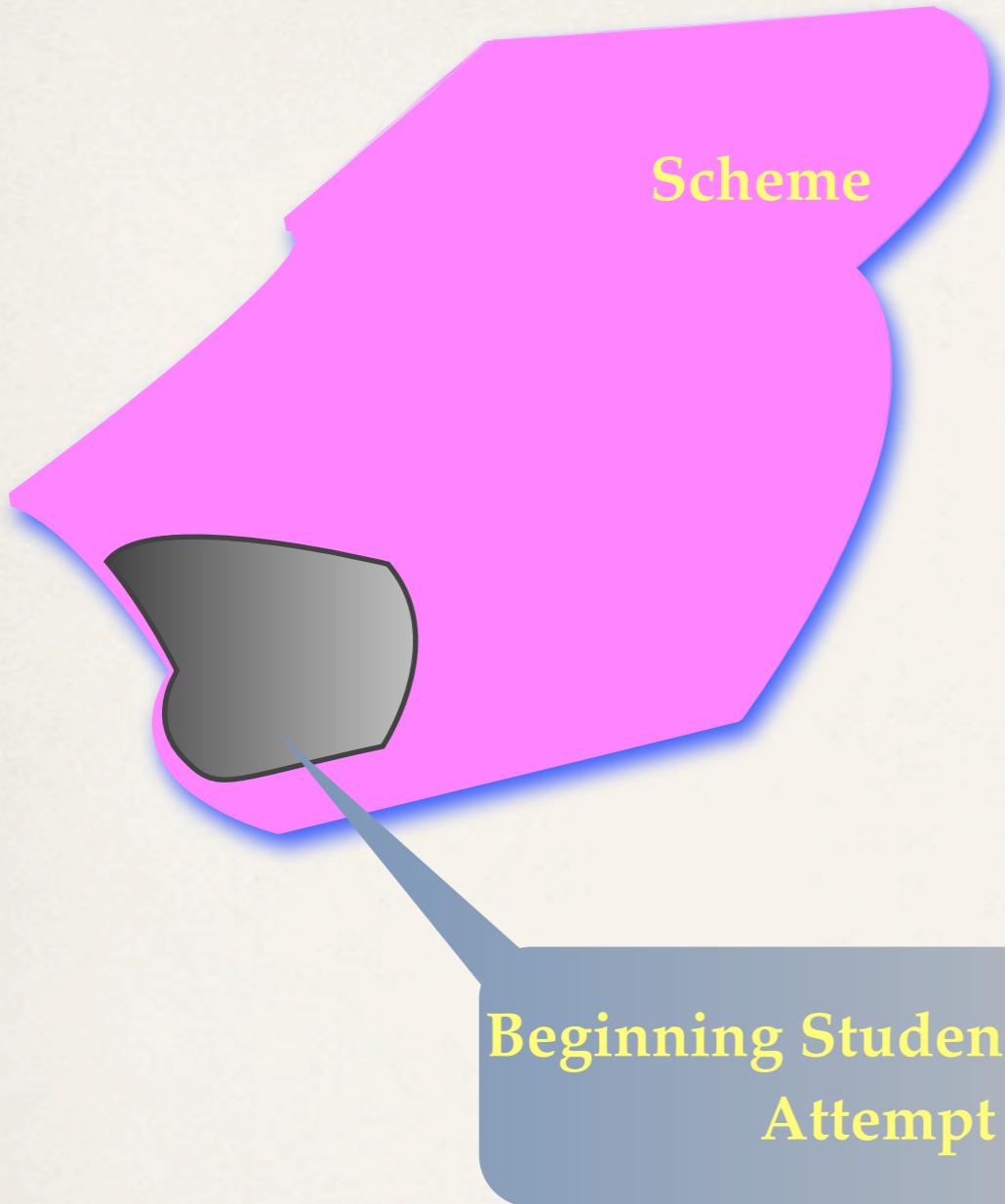
Programming in a teaching language



Programming in a teaching language

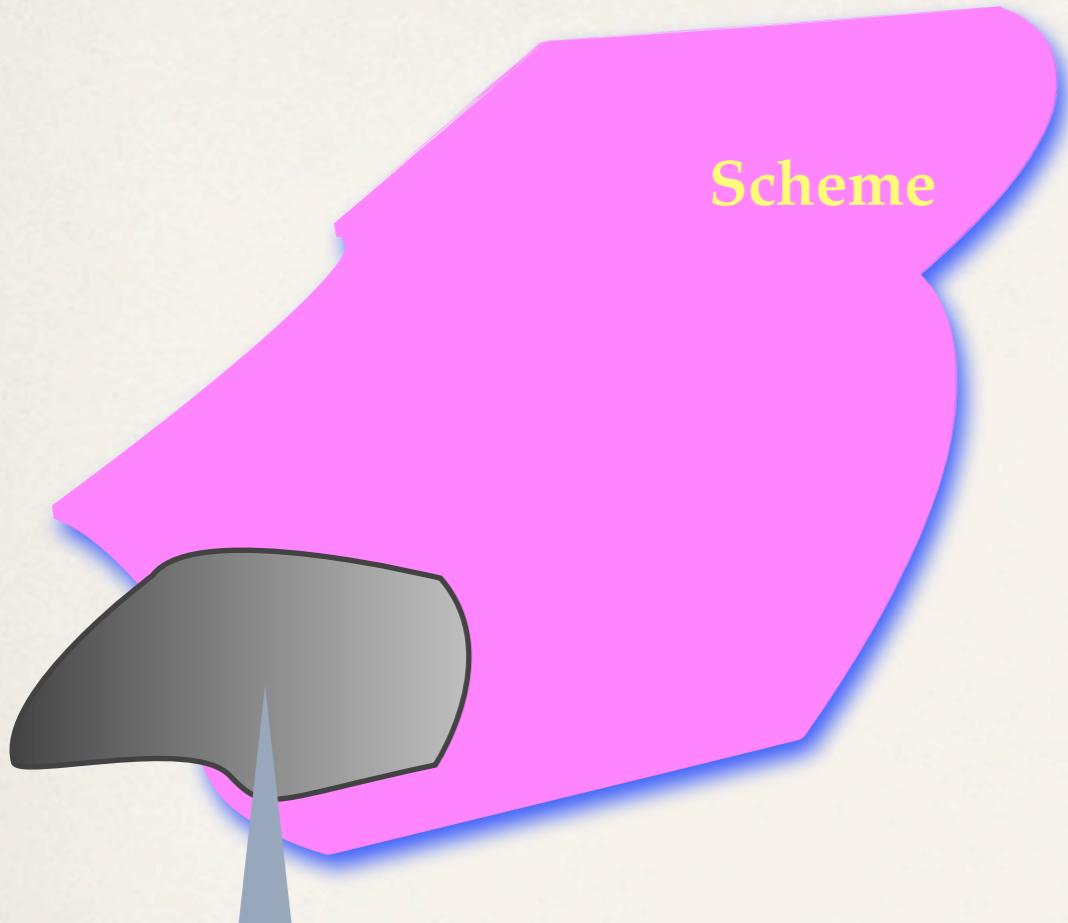


Programming in a teaching language



Lesson 2b: To teach *systematic design*, you may need a language that doesn't overlap with your chosen professional language.

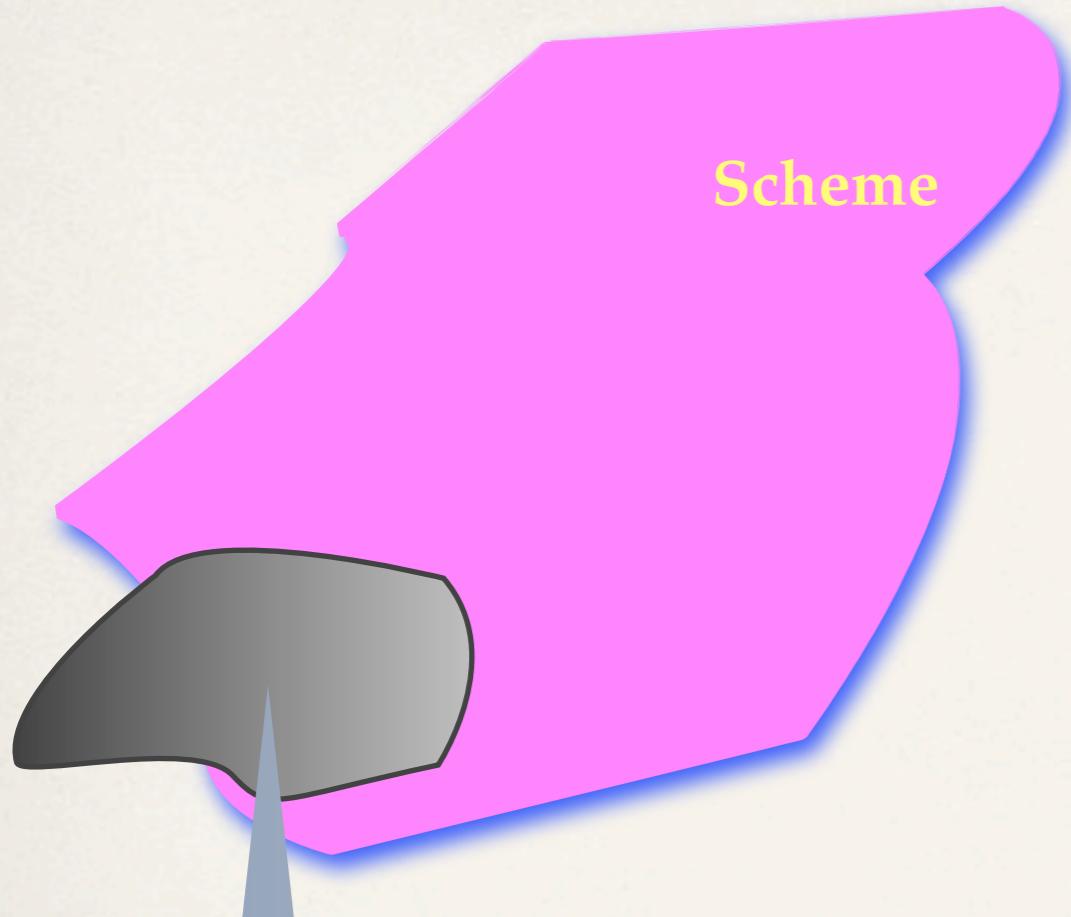
Programming in a teaching language



Beginning Student Language
Attempt 2

Lesson 2b: To teach *systematic design*, you may need a language that doesn't overlap with your chosen professional language.

Programming in a teaching language



Beginning Student Language
Attempt 2

fun programming with
“plain” mathematics

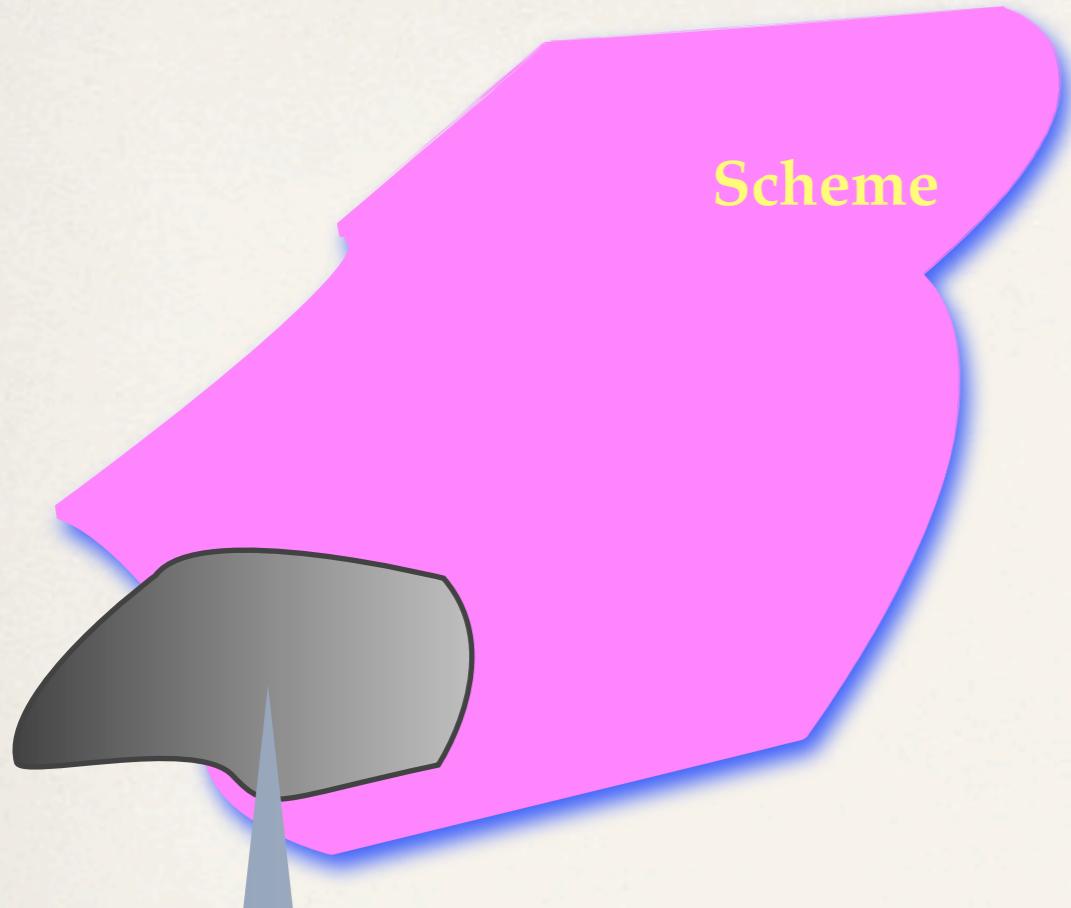
Lesson 2b: To teach *systematic design*, you may need a language that doesn't overlap with your chosen professional language.

Systematic Program Design



start with one and smoothly reach the other.

Programming in a teaching language



Scheme

Lesson 2b: To teach *systematic design*, you may need a language that doesn't overlap with your chosen professional language.

Systematic Program Design

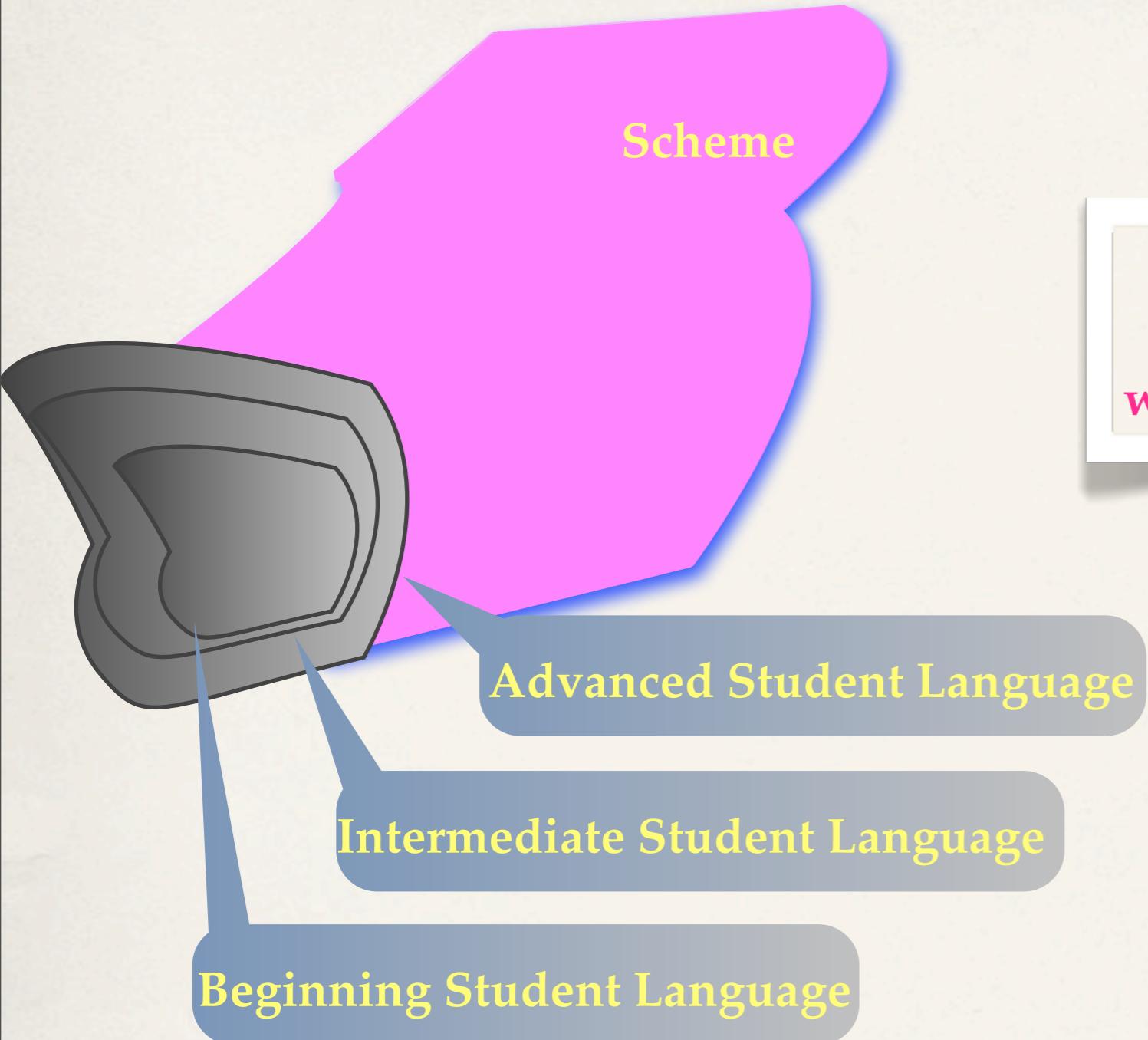
Beginning Student Language
Attempt 2

fun programming with
“plain” mathematics



start with one and smoothly reach the other.

Programming in teaching languages



Lesson 2c: To accommodate smooth transitions, you need a whole *series of teaching languages*.

Programming in teaching languages: More Information

Felleisen, Findler, Flatt, Krishnamurthi.

The TeachScheme! project: computing and programming for every student.

Journal of Computer Science Education. Vol. 14(1), 2004, 55--77.

Findler, Flatt, Krishnamurthi, Steckler, Felleisen

The DrScheme Programming Environment.

Journal of Functional Programming. Vol. 12(2), 2002, 159--182.

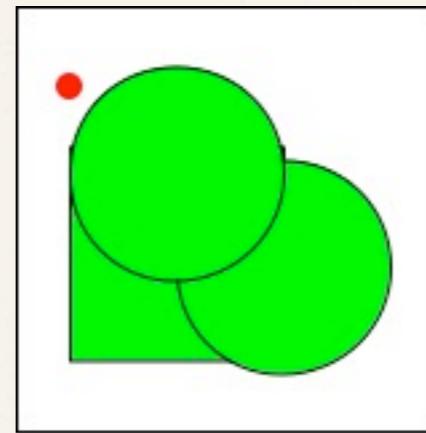
Systematic Program Design

Programming by example

Problem: Design a program that determines whether a mouse click is inside some given shape. A shape is one of the following:

- a square of some size at some location;
- a circle of some radius at some location;
- or the composition of two shapes,
one shape on top of another.

(We pose such a problem as early as
in week 4 and as late as in week 6 for
complete novice programmers.)



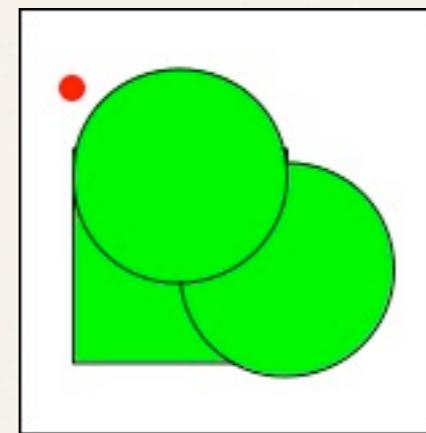
(DARPA Programming Contest, 1993)

Programming by example

Problem: Design a program that determines whether a mouse click is inside some given shape. A shape is one of the following:

- a square of some size at some location;
- a circle of some radius at some location;
- or the composition of two shapes,
one shape on top of another.

(We pose such a problem as early as in week 4 and as late as in week 6 for complete novice programmers.)



(DARPA Programming Contest, 1993)

Is a Cartesian point
inside some shape?

Programming by example

```
(struct square (location size))
(struct circle (location radius))
(struct composition (top bot))

;; Shape Point -> Boolean
;; is point pt inside of shape sh?
(define (shape-in? sh pt)
  (cond
    [(is-square? sh) (square-in? sh pt)]
    [(is-circle? sh) (circle-in? sh pt)]
    [(is-composition? sh)
     (or (shape-in? (shape-top sh) pt)
         (shape-in? (shape-bot sh) pt))]))
```

Programming by example

```
interface IShape {  
    // is point pt inside of this shape  
    boolean inside(Point pt);  
}  
  
class Square implements IShape {  
    private Point location;  
    private int size;  
    public boolean inside(Point pt) { ... }  
}  
  
class Circle implements IShape {  
    private Point location;  
    private int radius;  
    public boolean inside(Point pt) { ... }  
}  
  
class Composition implements IShape {  
    private IShape top;  
    private IShape bot;  
    public boolean inside(Point pt) {  
        return top.inside(pt) || bot.inside(pt);  
    }  
}
```

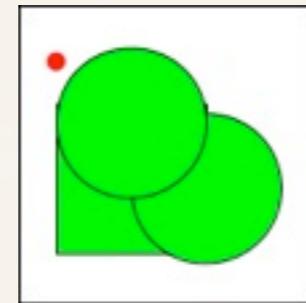
Programming by example

Problem Statement

Problem: Design a program that determines whether a mouse click is inside some given shape. A shape is one of the following:

- a square of some size at some location;
- a circle of some radius at some location;
- or the composition of two shapes,
one shape on top of another.

(We pose such a problem as early as
in week 4 and as late as in week 6 for
complete novice programmers.)



Natural Solution

```
(struct square (location size))
(struct circle (location radius))
(struct composition (top bot))

;; Shape Point -> Boolean
;; is point pt inside of shape sh?
(define (shape-in? sh pt)
  (cond
    [(is-square? sh) (square-in? sh pt)]
    [(is-circle? sh) (circle-in? sh pt)]
    [(is-composition? sh)
     (or (shape-in? (shape-top sh) pt)
         (shape-in? (shape-bot sh) pt))]))
```

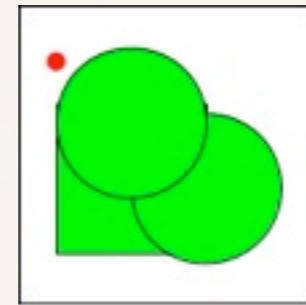
Programming by example

Problem Statement

Problem: Design a program that determines whether a mouse click is inside some given shape. A shape is one of the following:

- a square of some size at some location;
- a circle of some radius at some location;
- or the composition of two shapes,
one shape on top of another.

(We pose such a problem as early as in week 4 and as late as in week 6 for complete novice programmers.)



Natural Solution

```
(struct square (location size))  
(struct circle (location radius))  
(struct composition (top bot))  
  
;; Shape Point -> Boolean  
;; is point pt inside of shape sh?  
(define (shape-in? sh pt)  
  (cond  
    [(is-square? sh) (square-in? sh pt)]  
    [(is-circle? sh) (circle-in? sh pt)]  
    [(is-composition? sh)  
     (or (shape-in? (shape-top sh) pt)  
         (shape-in? (shape-bot sh) pt))]))
```

How did you go from the *problem statement* to the *solution*?

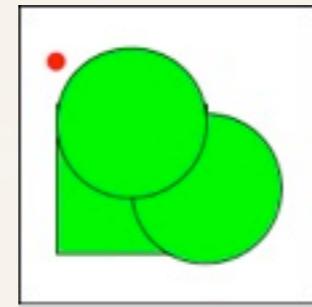
Programming by example

Problem Statement

Problem: Design a program that determines whether a mouse click is inside some given shape. A shape is one of the following:

- a square of some size at some location;
- a circle of some radius at some location;
- or the composition of two shapes,
one shape on top of another.

(We pose such a problem as early as in week 4 and as late as in week 6 for complete novice programmers.)



How do you teach your students to get from here to there?

Natural Solution

```
(struct square (location size))  
(struct circle (location radius))  
(struct composition (top bot))  
  
;; Shape Point -> Boolean  
;; is point pt inside of shape sh?  
(define (shape-in? sh pt)  
  (cond  
    [(is-square? sh) (square-in? sh pt)]  
    [(is-circle? sh) (circle-in? sh pt)]  
    [(is-composition? sh)  
     (or (shape-in? (shape-top sh) pt)  
         (shape-in? (shape-bot sh) pt))]))
```

How did you go from the *problem statement* to the *solution*?

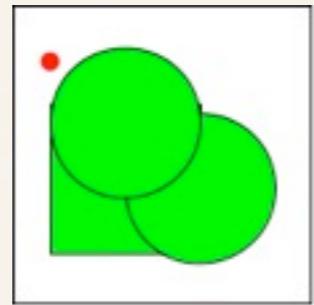
Programming by example

Problem Statement

Problem: Design a program that determines whether a mouse click is inside some given shape. A shape is one of the following:

- a square of some size at some location;
- a circle of some radius at some location;
- or the composition of two shapes,
one shape on top of another.

(We pose such a problem as early as in week 4 and as late as in week 6 for complete novice programmers.)



How do you teach your students to get from here to there?

Natural Solution

```
(struct square (location size))  
(struct circle (location radius))  
(struct composition (top bot))  
  
;; Shape Point -> Boolean  
;; is point pt inside of shape sh?  
(define (shape-in? sh pt)  
  (cond  
    [(is-square? sh) (square-in? sh pt)]  
    [(is-circle? sh) (circle-in? sh pt)]  
    [(is-composition? sh)  
     (or (shape-in? (shape-top sh) pt)  
         (shape-in? (shape-bot sh) pt))]))
```



How did you go from the *problem statement* to the *solution*?

What do you do when your students get stuck?

How to Design Programs

Lesson 3: To empower your students,
you need a systematic design process
from the *very* beginning.

How to Design Programs

Structural Design: Forms of Data

The Design Recipe

	atomic	enumer.	structs	hier.	union	recursive	mut. rec.
data description							
purpose statement							
functional examples							
template outline							
code!							
examples to testing							

Lesson 3: To empower your students,
you need a systematic design process
from the *very* beginning.

How to Design Programs

Structural Design: Forms of Data

The Design Recipe

Process Steps	atomic	enumer.	structs	hier.	union	recursive	mut. rec.
data description							
purpose statement							
functional examples							
template outline							
code!							
examples to testing							

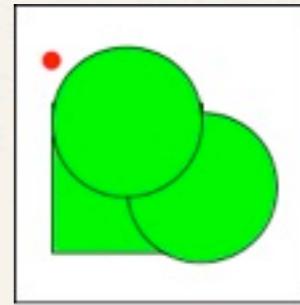
Lesson 3: To empower your students,
you need a systematic design process
from the *very* beginning.

How to Design Programs -- *data descriptions*

Problem Statement

Problem: Design a program that determines whether a mouse click is inside some given shape. A shape is one of the following:

- a square of some size at some location;
- a circle of some radius at some location;
- or the composition of two shapes,
one shape on top of another.



How to Design Programs -- *data descriptions*

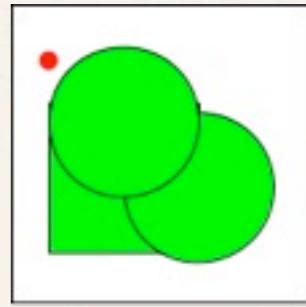
Q: What kind of information does the problem talk about?

**A: Points and Shapes, incl.
Squares, Circles, & Composites**

Problem Statement

Problem: Design a program that determines whether a mouse click is inside some given shape. A shape is one of the following:

- a square of some size at some location;
- a circle of some radius at some location;
- or the composition of two shapes,
one shape on top of another.



How to Design Programs -- *data descriptions*

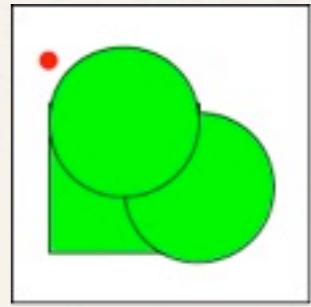
Q: What kind of information does the problem talk about?

**A: Points and Shapes, incl.
Squares, Circles, & Composites**

Problem Statement

Problem: Design a program that determines whether a mouse click is inside some given shape. A shape is one of the following:

- a square of some size at some location;
- a circle of some radius at some location;
- or the composition of two shapes,
one shape on top of another.



Q: What data do you use to represent this information?

A: Structures & unions.

```
(struct point (x y))  
  
(struct square (location size))  
(struct circle (location radius))  
(struct composition (top bot))  
;; A Shape is a square, a circle,  
;; or a composition.
```

How to Design Programs -- *data descriptions*

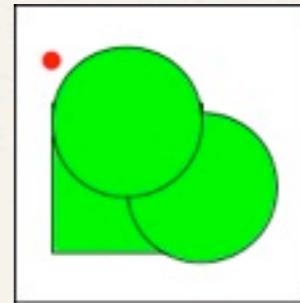
Q: What kind of information does the problem talk about?

A: Points and Shapes, incl. Squares, Circles, & Composites

Problem Statement

Problem: Design a program that determines whether a mouse click is inside some given shape. A shape is one of the following:

- a square of some size at some location;
- a circle of some radius at some location;
- or the composition of two shapes, one shape on top of another.



Q: What data do you use to represent this information?

A: Structures & unions.

```
(struct point (x y))  
  
(struct square (location size))  
(struct circle (location radius))  
(struct composition (top bot))  
;; A Shape is a square, a circle,  
;; or a composition.
```

Q: Can you create example instances from your data descriptions?

A: If not, they are not precise enough.

```
(define ex1 (square (point 10 20) 50))  
(define ex2 (circle (point 30 10) 40))  
(define ex3 (composition ex1 ex2))
```

How to Design Programs -- *signature & purpose statement*

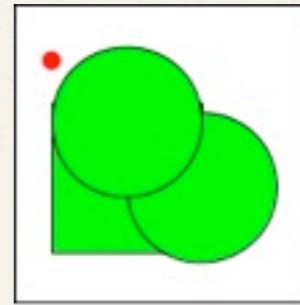
Q: What kind of data does the desired method/function consume? What does it produce?

A: It consumes *Points* and *Shapes* & returns *Boolean*.

Problem Statement

Problem: Design a program that determines whether a mouse click is inside some given shape. A shape is one of the following:

- a square of some size at some location;
- a circle of some radius at some location;
- or the composition of two shapes, one shape on top of another.



`;; Shape Point -> Boolean`

How to Design Programs -- *signature & purpose statement*

Q: What kind of data does the desired method/function consume? What does it produce?

A: It consumes *Points* and *Shapes* & returns *Boolean*.

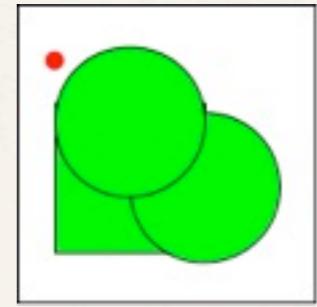
Q: Can you summarize its purpose as a one-liner?

A: It checks whether the given *Point* is inside the given *Shapes* & returns *Boolean*.

Problem Statement

Problem: Design a program that determines whether a mouse click is inside some given shape. A shape is one of the following:

- a square of some size at some location;
- a circle of some radius at some location;
- or the composition of two shapes, one shape on top of another.



`;; Shape Point -> Boolean`

`;; is point pt inside of shape sh?`

How to Design Programs -- *signature & purpose statement*

Q: What kind of data does the desired method/function consume? What does it produce?

A: It consumes *Points* and *Shapes* & returns *Boolean*.

Q: Can you summarize its purpose as a one-liner?

A: It checks whether the given *Point* is inside the given *Shapes* & returns *Boolean*.

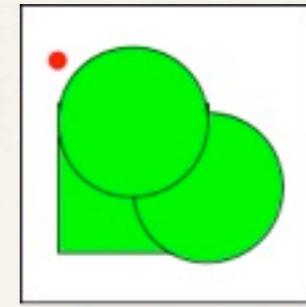
Q: Can you formulate a function stub?

A: The function takes two arguments and *false* is a good sample value.

Problem Statement

Problem: Design a program that determines whether a mouse click is inside some given shape. A shape is one of the following:

- a square of some size at some location;
- a circle of some radius at some location;
- or the composition of two shapes, one shape on top of another.



`;; Shape Point -> Boolean`

`;; is point pt inside of shape sh?`

```
(define (in? sh pt)
  false)
```

How to Design Programs -- *functional examples*

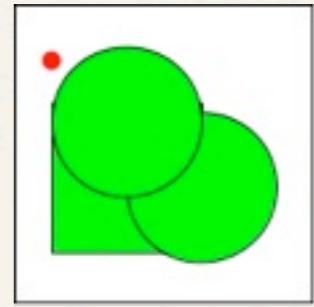
Q: Can you illustrate the purpose statement with examples?

A: It is good to start from the data examples.

Problem Statement

Problem: Design a program that determines whether a mouse click is inside some given shape. A shape is one of the following:

- a square of some size at some location;
- a circle of some radius at some location;
- or the composition of two shapes,
one shape on top of another.



```
; ; given (square (point 10 20) 30) and (point 15 25)  
; ; the answer should be true  
; ; given (circle (point 20 30) 10) and (point 90 90)  
; ; the answer should be false
```

How to Design Programs -- *functional examples*

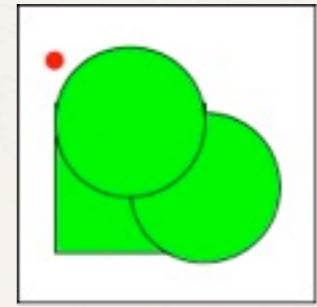
Q: Can you illustrate the purpose statement with examples?

A: It is good to start from the data examples.

Problem Statement

Problem: Design a program that determines whether a mouse click is inside some given shape. A shape is one of the following:

- a square of some size at some location;
- a circle of some radius at some location;
- or the composition of two shapes,
one shape on top of another.



```
; ; given (square (point 10 20) 30) and (point 15 25)  
; ; the answer should be true  
; ; given (circle (point 20 30) 10) and (point 90 90)  
; ; the answer should be false
```

Q: Can you formulate them as test cases?

```
(check-expect (in? ex1 pt0) true)  
(check-expect (in? ex2 pt0) false)  
(check-expect (in? ex3 pt0) true) ;; or (!)
```

How to Design Programs -- *functional examples*

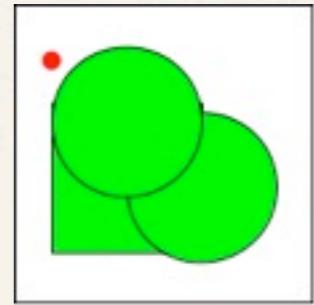
Q: Can you illustrate the purpose statement with examples?

A: It is good to start from the data examples.

Problem Statement

Problem: Design a program that determines whether a mouse click is inside some given shape. A shape is one of the following:

- a square of some size at some location;
- a circle of some radius at some location;
- or the composition of two shapes,
one shape on top of another.



```
; ; given (square (point 10 20) 30) and (point 15 25)  
; ; the answer should be true  
; ; given (circle (point 20 30) 10) and (point 90 90)  
; ; the answer should be false
```

Q: Can you formulate them as test cases?

Lesson 2d: Your PL must support *test coverage* automatically.

```
(check-expect (in? ex1 pt0) true)  
(check-expect (in? ex2 pt0) false)  
(check-expect (in? ex3 pt0) true) ;; or (!)
```

How to Design Programs -- *time for an outline*

```
(struct square (location size))
(struct circle (location radius))
(struct composition (top bot))

(define ex3 (composition ex1 ex2))

;; A Shape is one of:
;; -- (square Point NaturalNumber)
;; -- (circle Point NaturalNumber)
;; -- (composition Shape Shape)

;; Shape Point -> Boolean
;; is point pt inside of shape sh?

(check-expect (in? ex3 pt0) true)

(define (in? sh pt)
  false)
```

How to Design Programs -- *time for an outline*

```
(struct square (location size))  
(struct circle (location radius))  
(struct composition (top bot))  
  
(define ex3 (composition ex1 ex2))
```

```
;; A Shape is one of:  
;; -- (square Point NaturalNumber)  
;; -- (circle Point NaturalNumber)  
;; -- (composition Shape Shape)
```

```
;; Shape Point -> Boolean  
;; is point pt inside of shape sh?
```

```
(check-expect (in? ex3 pt0) true)
```

```
(define (in? sh pt)  
  false)
```

The goal of this step is to systematically create the body of the function from the data definition.

How to Design Programs -- *time for an outline*

```
(struct square (location size))  
(struct circle (location radius))  
(struct composition (top bot))  
  
(define ex3 (composition ex1 ex2))
```

;; A Shape is one of:
;; -- (square Point NaturalNumber)
;; -- (circle Point NaturalNumber)
;; -- (composition Shape Shape)

;; Shape Point -> Boolean
;; is point pt inside of shape sh?

```
(check-expect (in? ex3 pt0) true)
```

```
(define (in? sh pt)  
  (cond  
    [ ... ]  
    [ ... ]  
    [ ... ])))
```

Q: How many subclasses does the data definition mention?

A: Three. And therefore we need a 3-way conditional.

How to Design Programs -- *time for an outline*

Q: Which predicates distinguish the subclasses?

A: *square?* and *circle?* and *composition?* of *sh*

```
(struct square (location size))  
(struct circle (location radius))  
(struct composition (top bot))  
  
(define ex3 (composition ex1 ex2))
```

```
;; A Shape is one of:  
;; -- (square Point NaturalNumber)  
;; -- (circle Point NaturalNumber)  
;; -- (composition Shape Shape)
```

```
;; Shape Point -> Boolean  
;; is point pt inside of shape sh?
```

```
(check-expect (in? ex3 pt0) true)
```

```
(define (in? sh pt)  
  (cond  
    [ (square? sh) ... ]  
    [ (circle? sh) ... ]  
    [ (composition? sh) ... ])))
```

```
(struct square (location size))  
(struct circle (location radius))  
(struct composition (top bot))  
  
(define ex3 (composition ex1 ex2))
```

;; A Shape is one of:
;; -- (square Point NaturalNumber)
;; -- (circle Point NaturalNumber)
;; -- (composition Shape Shape)

;; Shape Point -> Boolean
;; is point pt inside of shape sh?

```
(check-expect (in? ex3 pt0) true)
```

(define (in? sh pt)
 (cond
 [(square? sh) ... sh ...]
 [(circle? sh) ... sh ...]
 [(composition? sh)
 ... (composition-top sh) ...
 ... (composition-bot sh) ...])))

Q: Which cases deal with structures?

A: All and the outline should list their fields.

```
(struct square (location size))  
(struct circle (location radius))  
(struct composition (top bot))  
  
(define ex3 (composition ex1 ex2))
```

;; A Shape is one of:
;; -- (square Point NaturalNumber)
;; -- (circle Point NaturalNumber)
;; -- (composition Shape Shape)

;; Shape Point -> Boolean
;; is point pt inside of shape sh?

(check-expect (in? ex3 pt0) true)

```
(define (in? sh pt)  
  (cond  
    [ (square? sh) ... sh ... ]  
    [ (circle? sh) ... sh ... ]  
    [ (composition? sh)  
      ..(in? (composition-top sh))...  
      ..(in? (composition-bot sh)).] )))
```

Q: Do any of the fields refer back to the data definition?

A: The two in the third case, and that is where *in?* needs recursion.

```
(struct square (location size))  
(struct circle (location radius))  
(struct composition (top bot))  
  
(define ex3 (composition ex1 ex2))
```

Q: Do any of the fields refer back to the data definition?

A: The two in the third case, and that is where *in?* needs recursion.

```
;; A Shape is one of:  
;; -- (square Point NaturalNumber)  
;; -- (circle Point NaturalNumber)  
;; -- (composition Shape Shape)
```

```
;; Shape Point -> Boolean  
;; is point pt inside of shape sh?  
  
(check-expect (in? ex3 pt0) true)
```

```
(define (in? sh pt)  
  (cond  
    [ (square? sh) ... sh ... ]  
    [ (circle? sh) ... sh ... ]  
    [ (composition? sh)  
      .. (in? (composition-top sh) ...) ...  
      .. (in? (composition-bot sh) ...) ]))
```

How to Design Programs -- code!

```
(struct square (location size))  
(struct circle (location radius))  
(struct composition (top bot))  
  
(define ex1 (square (point 10 20) 50))  
(define ex2 (circle (point 30 10) 40))  
(define ex3 (composition ex1 ex2))
```

Q: Can you infer from the examples the return value for the base cases?

A: ... left to auxiliaries ...

~~;; Shape Point -> Boolean~~

```
(check-expect (in? ex2 pt0) false)  
(check-expect (in? ex3 pt0) true)  
  
(define (in? sh pt)  
  (cond  
    [ (square? sh) ... sh ... ]  
    [ (circle? sh) ... sh ... ]  
    [ (composition? sh)  
      ..(in? (composition-top sh)) ...  
      ..(in? (composition-bot sh)).] ))
```

How to Design Programs -- code!

```
(struct square (location size))  
(struct circle (location radius))  
(struct composition (top bot))  
  
(define ex1 (square (point 10 20) 50))  
(define ex2 (circle (point 30 10) 40))  
(define ex3 (composition ex1 ex2))
```

Q: Can you infer from the examples how to combine the values of the expressions in the *recursive case(s)*?

A: Yes a point is in a composite shape if it is either in one or the other.

```
;; Shape Point -> Boolean  
  
(check-expect (in? ex2 pt0) false)  
(check-expect (in? ex3 pt0) true)  
  
(define (in? sh pt)  
  (cond  
    [ (square? sh) (in-square? sh pt) ]  
    [ (circle? sh) (in-circle? sh pt) ]  
    [ (composition? sh)  
      ... (in? (composition-top sh) ) ...  
      ... (in? (composition-bot sh) ) . . . ] ) )
```

How to Design Programs -- *code!*

```
(struct square (location size))  
(struct circle (location radius))  
(struct composition (top bot))  
  
(define ex1 (square (point 10 20) 50))  
(define ex2 (circle (point 30 10) 40))  
(define ex3 (composition ex1 ex2))
```

```
; ; Shape Point -> Boolean  
  
(check-expect (in? ex2 pt0) false)  
(check-expect (in? ex3 pt0) true)  
  
(define (in? sh pt)  
  (cond  
    [ (square? sh) (in-square? sh pt) ]  
    [ (circle? sh) (in-circle? sh pt) ]  
    [ (composition? sh)  
      (or (in? (composition-top sh pt))  
          (in? (composition-bot sh pt))) ] ) )
```

How to Design Programs -- *test and explore coverage*

The screenshot shows the DrRacket IDE interface with the title bar "Untitled 6 – DrRacket". The code editor contains the following Racket code:

```
(define pt0 (make-point 12 23))

(define ex1 (make-square (make-point 10 20) 50))
(define ex2 (make-circle (make-point 30 10) 40))
(define ex3 (make-composition ex1 ex2))

;; Shape Point -> Boolean
;; is the Point pt located within the Shape sh?

(check-expect (in? ex1 pt0) true)
(check-expect (in? ex2 pt0) false)

(define (in? sh pt)
  (cond
    [(square? sh) (in-square? sh pt)]
    [(circle? sh) (in-circle? sh pt)]
    [(composition? sh)
     (or (in? (composition-top sh) pt)
         (in? (composition-bot sh) pt))]))
```

The DrRacket status bar at the bottom displays:

- Welcome to [DrRacket](#), version 5.1.0.3--2011-03-02(4afd36c/g) [3m].
- Language: Beginning Student.
- Both tests passed!
- > |
- Beginning Student ▾
- 4:2
- 205.46 MB
- 👤

How to Design Programs -- *test and explore coverage*

Q: Can you run your tests and check the coverage?

A: Red is like a bug.

The screenshot shows the DrRacket interface with the following code:

```
(define pt0 (make-point 12 23))
(define ex1 (make-square (make-point 10 20) 50))
(define ex2 (make-circle (make-point 30 10) 40))
(define ex3 (make-composition ex1 ex2))

;; Shape Point -> Boolean
;; is the Point pt located within the Shape sh?

(check-expect (in? ex1 pt0) true)
(check-expect (in? ex2 pt0) false)

(define (in? sh pt)
  (cond
    [(square? sh) (in-square? sh pt)]
    [(circle? sh) (in-circle? sh pt)]
    [(composition? sh)
     (or (in? (composition-top sh) pt)
         (in? (composition-bot sh) pt))]))
```

The code defines several shapes (pt0, ex1, ex2, ex3) and a test function (check-expect). A red box highlights the definition of the `in?` function. Below the code, the DrRacket output window shows:

Language: Beginning Student.
Both tests passed!
> |

The bottom status bar indicates: Beginning Student ▾ 4:2 205.46 MB

How to Design Programs -- *test and explore coverage*

Q: Can you run your tests and check the coverage?

A: Red is like a bug.

Untitled 6 – DrRacket

```
(define pt0 (make-point 12 23))

(define ex1 (make-square (make-point 10 20) 50))
(define ex2 (make-circle (make-point 30 10) 40))
(define ex3 (make-composition ex1 ex2))

;; Shape Point -> Bool
;; is the Point pt located in sh?
(define (in? sh pt)
  (cond
    [(square? sh) (in-square? sh pt)]
    [(circle? sh) (in-circle? sh pt)]
    [(composition? sh)
     (or (in? (composition-top sh) pt)
         (in? (composition-bot sh) pt))]))
```

Language: Beginning Student.
Both tests passed!

4:2 205.46 MB

Beginning Student ▾

Lesson 1b: Your IDE must support
test coverage automatically.

How to Design Programs

Structural Design: Forms of Data

The Design Recipe

	atomic	enumer.	structs	hier.	union	recursive	mut. rec.
data description							
purpose statement							
functional examples							
template outline							
code!							
examples to testing							

How to Design Programs

Structural Design: Forms of Data

The Design Recipe

atomic

enumer.

structs

hier.

union

recursive

mut. rec.

None of the questions referred to the problem.

Process Steps

purpose statement

functional examples

template | outline

code!

examples to testing

How to Design Programs

Structural Design: Forms of Data

The Design Recipe

atomic

enum.

structs

hier.

union

recursive

mut. rec.

None of the questions referred to the problem.

This is *good*:

- it helps you diagnose students' problems
- it empowers your student eventually

Process Steps

functional examples

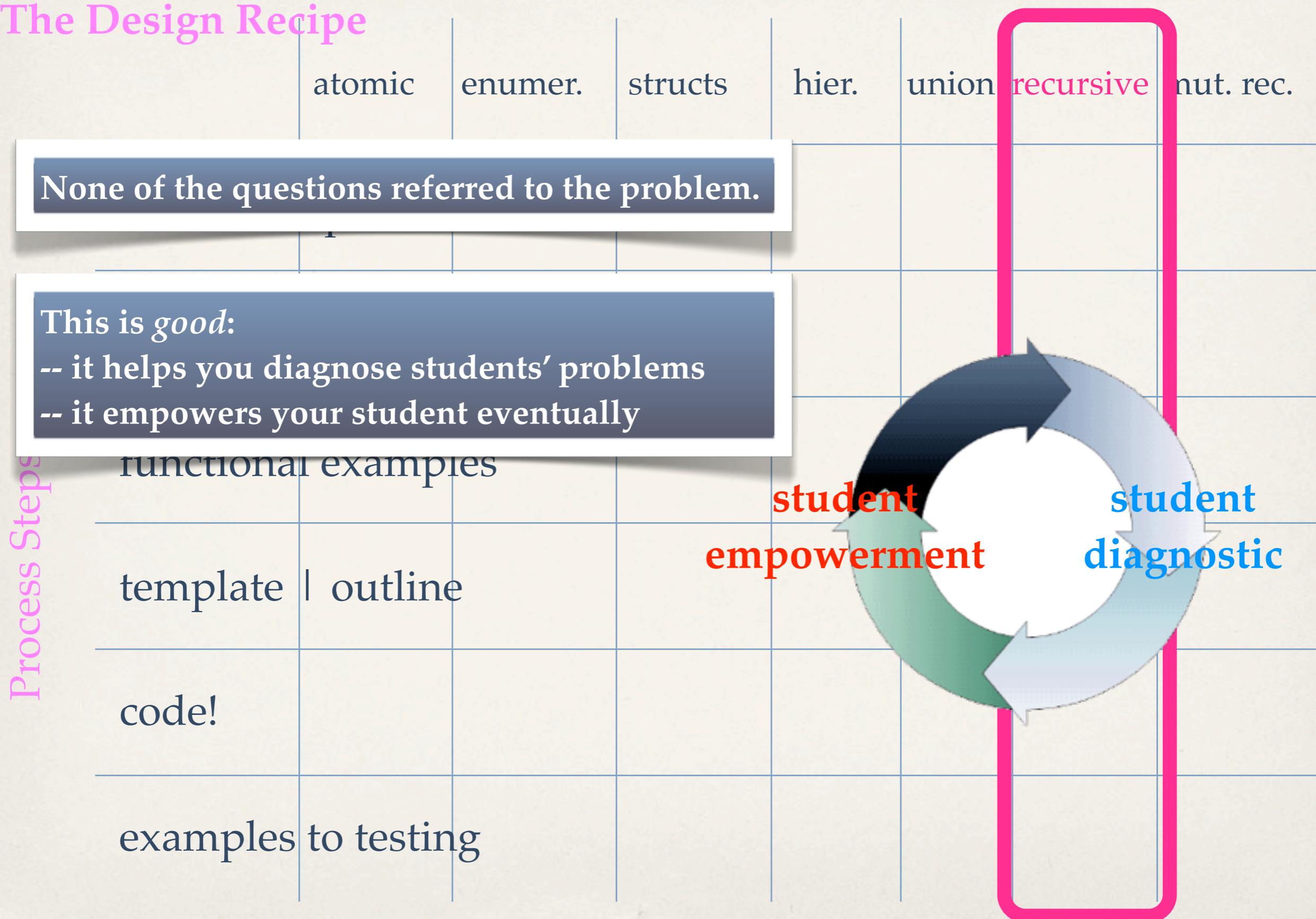
template | outline

code!

examples to testing

student
empowerment

student
diagnostic



How to Design Programs

Structural Design: Forms of Data

The Design Recipe

Process Steps

atomic
data description

purpose statement

functional example

template | outline

code!

examples to testing

atomic

enum

structs

hier

union

recursive

mut. rec.

There is lots more on design:

- structural design
- abstracting design
- generative recursion
- design with mutation

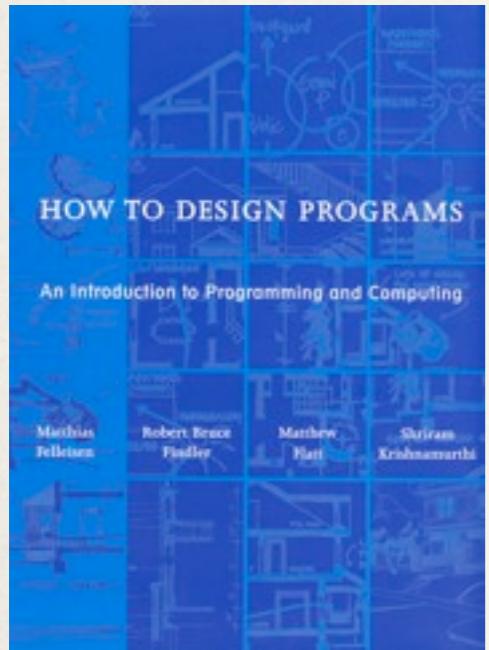
- iterative refinement

- designing GUIs

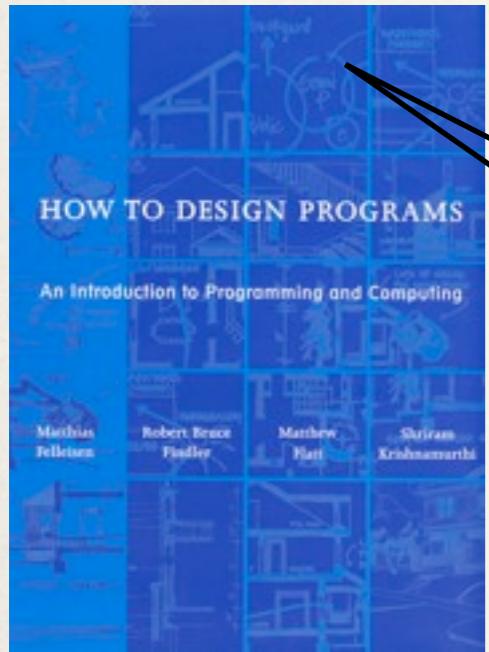
- designing batch

How to Design Programs: the book

two text books in one



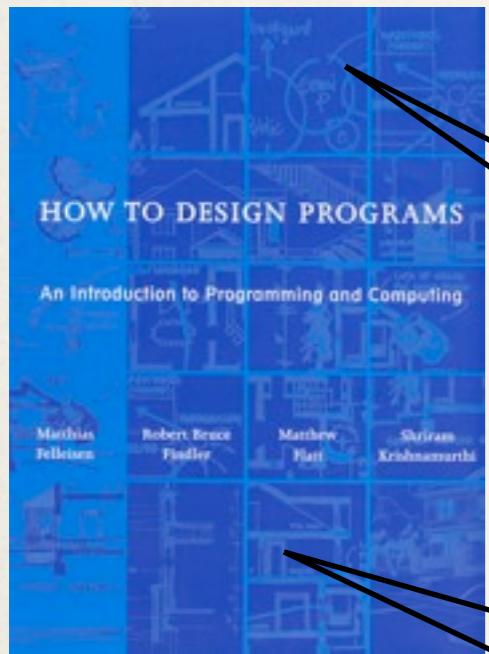
How to Design Programs: the book



two text books in one

regular text book for
students (college freshmen
and high school)

How to Design Programs: the book



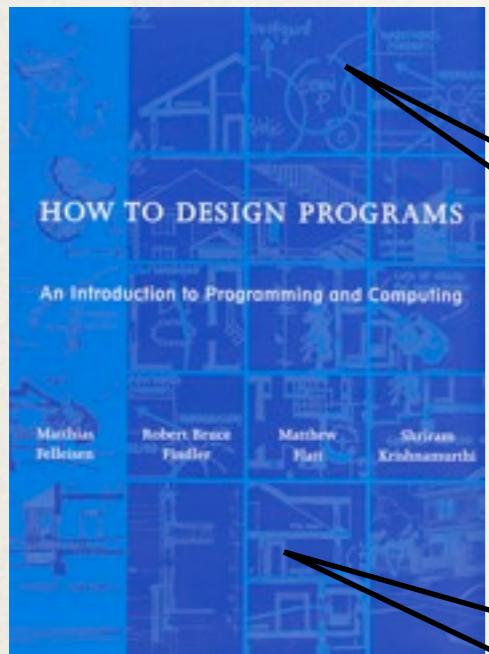
two text books in one

regular text book for
students (college freshmen
and high school)

how to teach systematic design
and computational problem solving

Felleisen, Findler, Flatt, Krishnamurthi.
The Structure and Interpretation of the Computer Science Curriculum.
Journal of Functional Programming. Vol. 14(4), 2004, 365-378.

How to Design Programs: the book



two text books in one

regular text book for
students (college freshmen
and high school)

how to teach systematic design
and computational problem solving

Felleisen, Findler, Flatt, Krishnamurthi.
The Structure and Interpretation of the Computer Science Curriculum.
Journal of Functional Programming. Vol. 14(4), 2004, 365-378.

<http://www.htdp.org/> [edition 1]

<http://www.ccs.neu.edu/home/matthias> [edition 2]

Failures & Challenges

Error Message: How well did we do?

The screenshot shows the DrRacket IDE interface with the title bar "Untitled 8 – DrRacket". The left sidebar has icons for file operations, undo, redo, and search. The code editor contains two definitions:

```
;; -----
;; Number -> {-1,0,+1}
(define (sign.v1 x)
  (cond
    [(positive? x) +1]
    [(zero? x) 0]
    [(negative? x) -1]))


;; -----
;; Number -> {-1,0,+1}
(define (sign.v2 x)
  (cond
    (positive? x) +1
    [(zero? x) 0]
    [(negative? x) -1]))
```

In the bottom right pane, the welcome message is displayed:

Welcome to [DrRacket](#), version 5.1.0.3--2011-03-02(4afd36c/g) [3m].
Language: Beginning Student.

A red error message is shown:

cond: expected a question--answer clause, but found a number

The status bar at the bottom shows "Beginning Student" in a dropdown, "4:7", "205.46 MB", and a battery icon.

Error Message: How well did we do?

The screenshot shows the DrRacket interface with two definitions of a sign function:

```
;; -----  
;; Number -> {-1,0,+1}  
(define (sign.v1 x)  
  (cond  
    [(positive? x) +1]  
    [(zero? x) 0]  
    [(negative? x) -1]))  
  
;; -----  
;; Number -> {-1,0,+1}  
(define (sign.v2 x)  
  (cond  
    (positive? x) +1  
    [(zero? x) 0]  
    [(negative? x) -1]))
```

Two specific parts of the code are highlighted with red rounded rectangles:

- The first clause of the `sign.v1` definition: `[(positive? x) +1]`
- The expression `(positive? x)` in the first clause of the `sign.v2` definition.

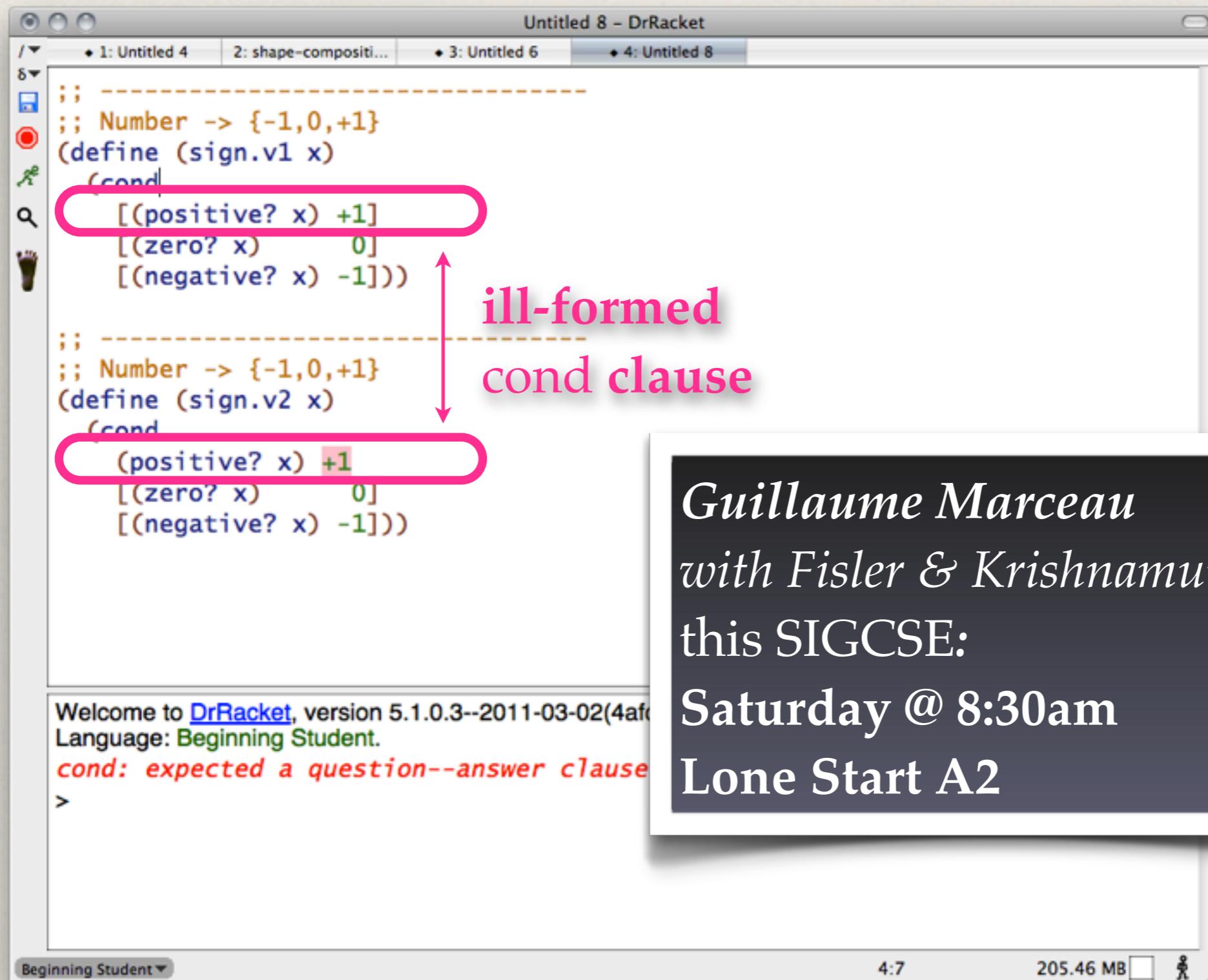
A pink arrow points from the text "ill-formed cond clause" to the error in the second definition.

In the DrRacket interaction area, the following error message is displayed:

```
Welcome to DrRacket, version 5.1.0.3--2011-03-02(4afd36c/g) [3m].  
Language: Beginning Student.  
cond: expected a question--answer clause, but found a number  
>
```

The status bar at the bottom shows: Beginning Student ▾ 4:7 205.46 MB

Error Message: How well did we do?



```
Untitled 8 – DrRacket
1: Untitled 4 2: shape-compositi... 3: Untitled 6 4: Untitled 8

;; -----
;; Number -> {-1,0,+1}
(define (sign.v1 x)
  (cond
    [(positive? x) +1]
    [(zero? x) 0]
    [(negative? x) -1])))

;; -----
;; Number -> {-1,0,+1}
(define (sign.v2 x)
  (cond
    (positive? x) +1
    [(zero? x) 0]
    [(negative? x) -1])))

Welcome to DrRacket, version 5.1.0.3--2011-03-02(4af0...
Language: Beginning Student.
cond: expected a question--answer clause
>
```

ill-formed cond clause

*Guillaume Marceau
with Fisler & Krishnamurthi
this SIGCSE:
Saturday @ 8:30am
Lone Start A2*

Delivery: Desk Top vs Web Browser

The screenshot shows the DrRacket IDE interface. The top window is titled "Untitled 4 - DrRacket". The code in the editor pane is:

```
(require 2htdp/image)
(define (launch t)
  (place-image/align rocket 50 300 'center 'bottom))
```

The preview pane below shows a white rectangle with a small red rocket ship positioned at its bottom center. In the bottom-left corner of the preview pane, there is some faint text that appears to be part of the code or a comment.

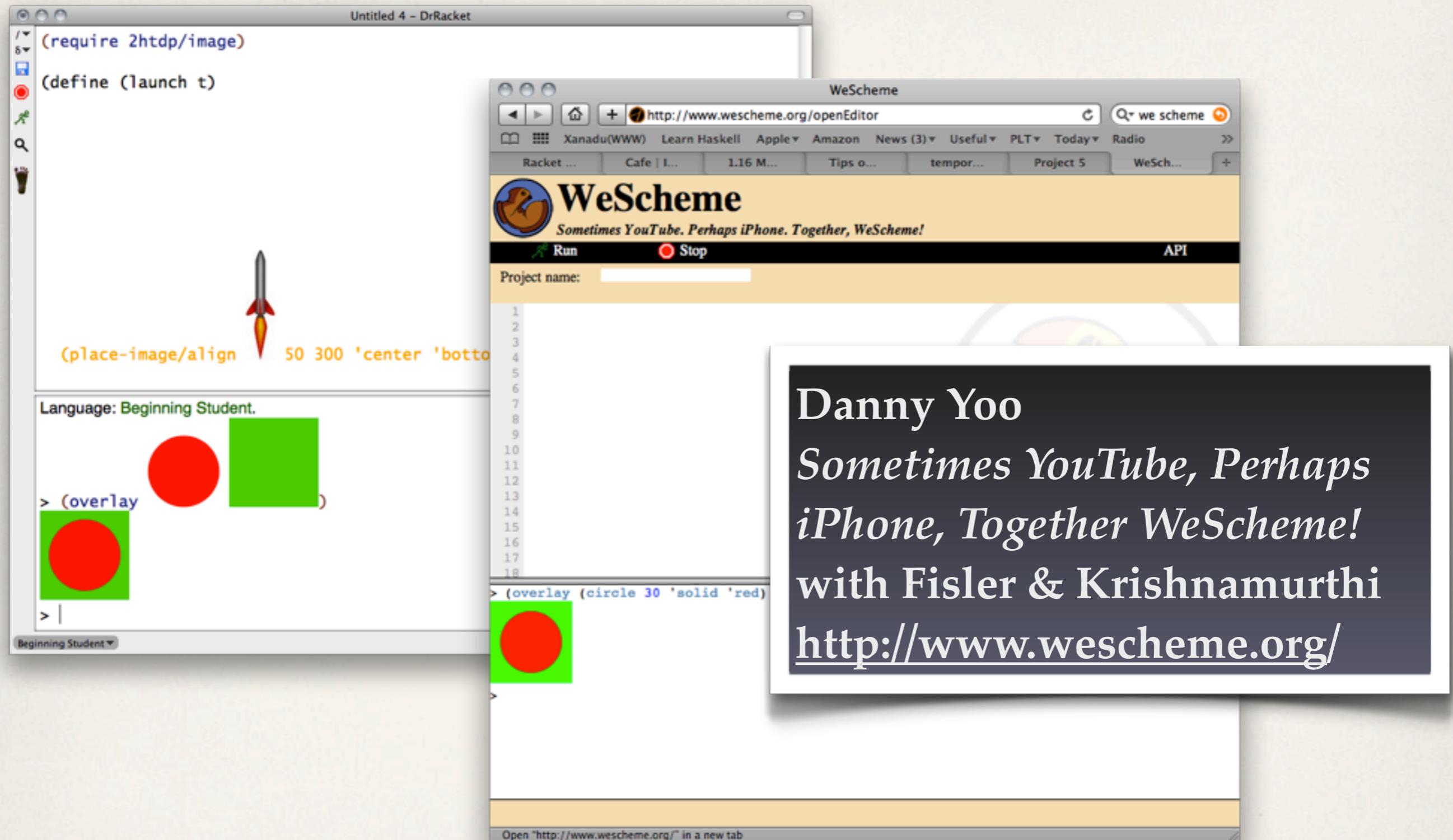
Below the preview pane, the language setting is set to "Beginning Student". The interaction pane shows the result of the code execution:

```
> (overlay
  (circle 100 "solid" "red")
  (square 100 "solid" "green"))
```

The interaction pane displays two overlapping shapes: a red circle on top of a green square. To the left of the interaction pane, there is a small preview window showing the same red circle on top of a green square.

At the bottom of the interface, there is a status bar with the text "Beginning Student", a progress bar showing "5.2", the memory usage "102.27 MB", and a user icon.

Delivery: Desk Top vs Web Browser



Syntax (1): How about those parentheses?

Valid Scheme

```
(moo-baz
  (* (foo bar)
      (add1 g))
  (retrieve
    qux
    (-> zip ding)))
```

Syntax (1): How about those parentheses?

Valid Scheme

```
(moo-baz
  (* (foo bar)
      (add1 g))
  (retrieve
    qux
    (-> zip ding)))
```

Valid C++

```
((foo<bar>) * (g++) )
.baz (!&qux::zip->ding());
```

Can you determine in which
order the pieces are evaluated?

Syntax (2): Is editing a pain?

```
;; RealNumber -> {+1, 0, -1}
(define (sign x)
  (cond
    [ (positive? x) +1]
    [ (zero? x)      0]
    [ (negative? x) -1] ))
```

Editing *any syntax* is a painful
exercises for all middle school
kids and other people too.

Syntax (2): Is editing a pain?

```
;; RealNumber -> {+1, 0, -1}
(define (sign x)
  (cond
    [ (positive? x) +1]
    [ (zero? x)      0]
    [ (negative? x) -1] ))
```

Editing *any syntax* is a painful exercises for all middle school kids and other people too.

We should have added some structure-oriented editing a long time ago.

Syntax vs. mathematics

What do all these pieces of C/C++/
Java/etc alii have in common?

```
x = x + 1
```

```
for(i = 0; i < 10, i++) {  
    sum = sum + i;
```

```
while (!in.eof) {  
    next = readInt();  
    sum = sum + next;  
}
```

Syntax vs. mathematics

What do all these pieces of C/C++/
Java/etc alii have in common?

```
x = x + 1
```

```
for(i = 0; i < 10, i++) {  
    sum = sum + i;
```

```
while (!in.eof) {  
    next = readInt();  
    sum = sum + next;  
}
```

None of this is mathematics.
Worse, some of it looks like
mathematics but has nothing
to do with it.

Syntax vs. mathematics

What do all these pieces of C/C++/
Java/etc alii have in common?

x = x + 1

```
for(i=0; i < 10, i++) {  
    sum = sum + i;
```

```
while (!feof) {  
    next = readInt();  
    sum = sum + next;  
}
```

None of this is mathematics.
Worse, some of it looks like
mathematics but has nothing
to do with it.

Syntax vs. mathematics

What do all these fragments from
our teaching languages have in
common?

```
(define x 10)
```

```
(define (conversion f)
  (* 9/5 (- f 32)))
```

```
(define (edit editor ke)
  (if (control-key? ke)
      editor
      (juxtapose editor ke)))
```

Syntax vs. mathematics

What do all these fragments from our teaching languages have in common?

```
(define x 10)
```

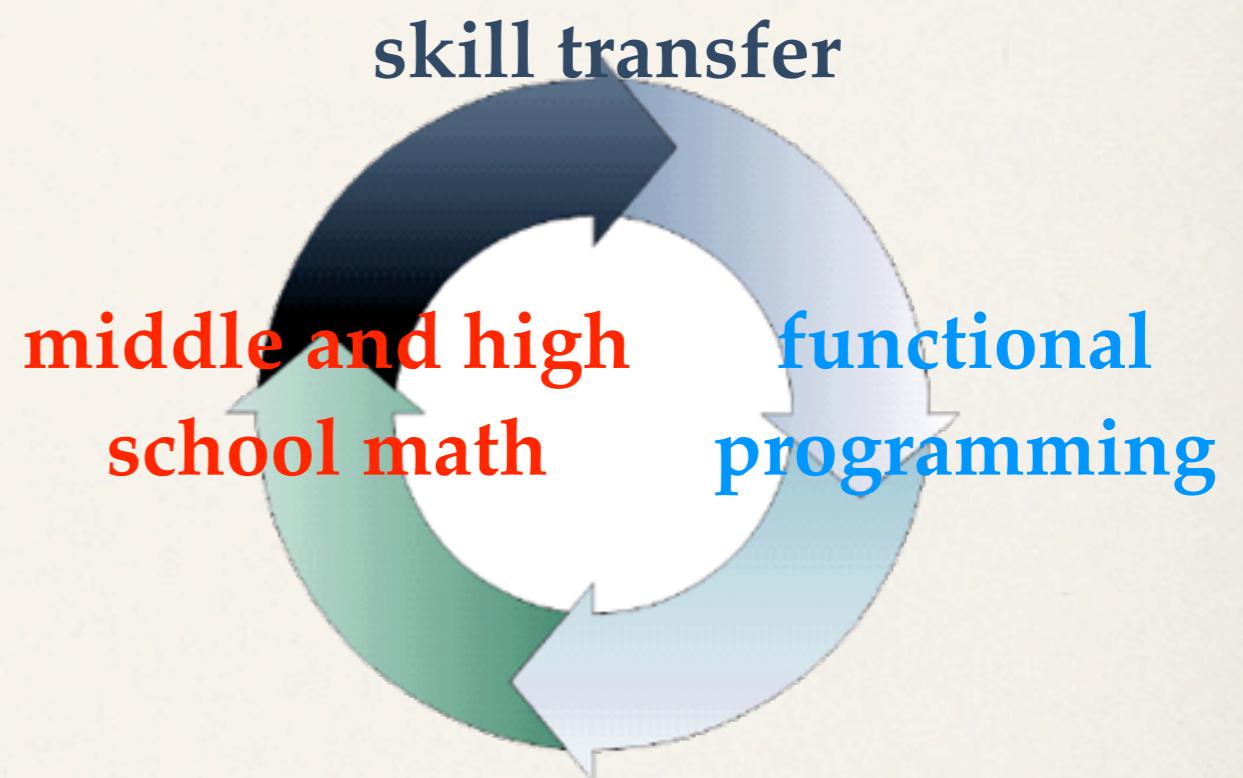
```
(define (conversion f)
  (* 9/5 (- f 32)))
```

```
(define (edit editor ke)
  (if (control-key? ke)
      editor
      (juxtapose editor ke)))
```

None of this may *look like* mathematics from middle school texts **but every construct has a direct and easily recognizable counterpart in mathematics.**

Syntax (3): Its meaning really *is* mathematics

So you may not like the parentheses, but the teaching languages really are mathematics.



Scheme, it's all about Scheme

Ada Algol
C++
Python PL/I
Perl

All computer science curricula
start from a course on the
*currently fashionable
programming language.*

Scheme, it's all about Scheme

Ada Algol
C++
Python PL/I
Perl

Every first course must be
about *a programming language*.
Ergo, yours must be about Scheme.

All computer science curricula
start from a course on the
currently fashionable
programming language.

Scheme, it's all about Scheme



All computer science curricula
start from a course on the
*currently fashionable
programming language.*

Every first course must be
about *a programming language*.
Ergo, yours must be about Scheme.

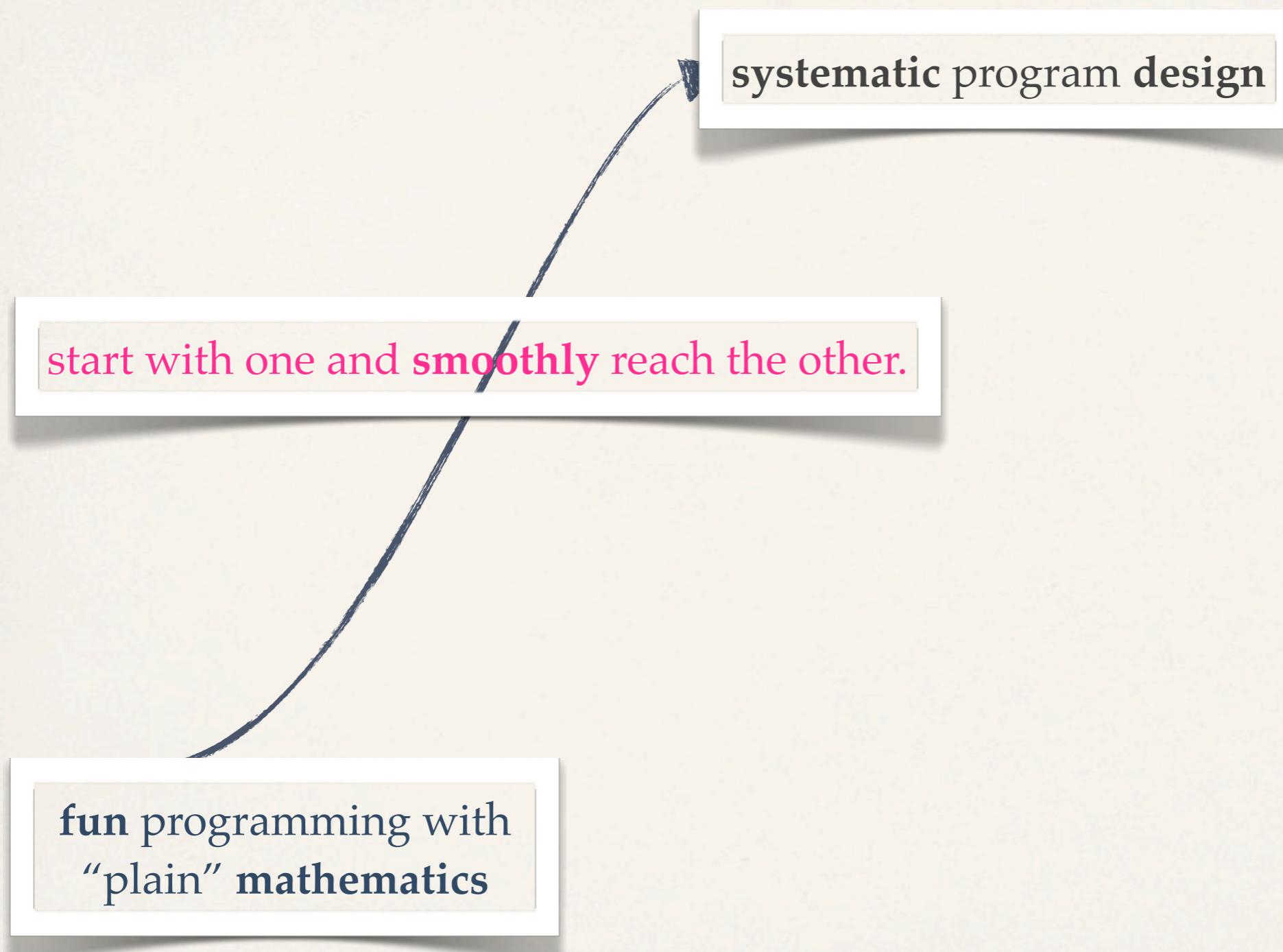
No, it really is about
connecting mathematics and
programming and design.

The *Program by Design* Project

Matthias Felleisen (PLT)
Northeastern University

Two Conclusions

The *Program by Design* Idea: Align ‘Rograming with ‘Rithmetic



The *Program by Design* Idea: Align ‘Rograming with ‘Rithmetic

start with one and **smoothly** reach the other.

fun programming with
“plain” mathematics

systematic program design

“Scheme”

Java

Python



The *Program by Design* Idea: Align ‘Rograming with ‘Rithmetic

start with one and **smoothly** reach the other.

fun programming with
“plain” mathematics

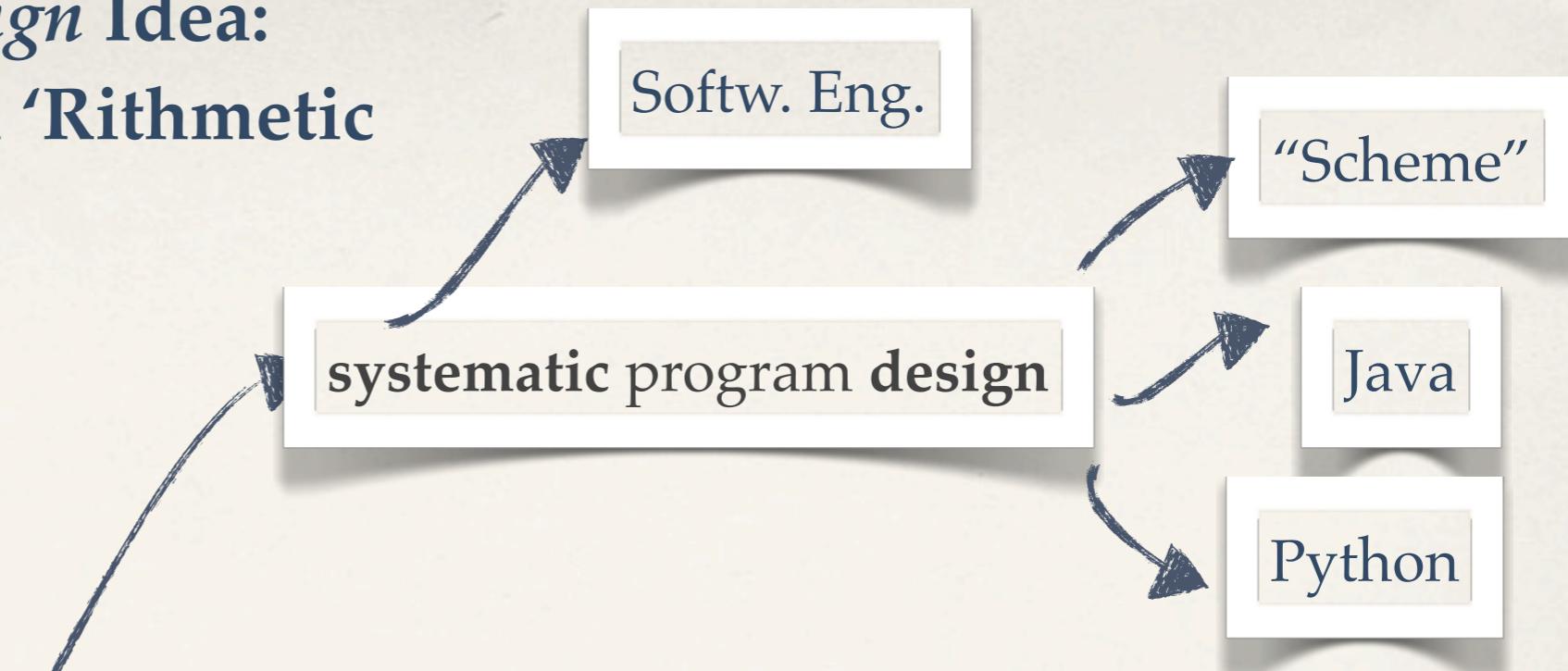
systematic program design

Softw. Eng.

“Scheme”

Java

Python



The *Program by Design* Idea: Align ‘Rograming with ‘Rithmetic

start with one and **smoothly** reach the other.

fun programming with
“plain” mathematics

systematic program design

Softw. Eng.

“Scheme”

Java

Python

transition to logic:
ACL2 & proving theorems
about video games

General Lessons

**Lesson 1: Your PL/IDE must support
an arithmetic of images.**

Get'em hooked. Programming with mathematics is fun and benefits all.

General Lessons

**Lesson 1: Your PL/IDE must support
an arithmetic of images.**

Get'em hooked. Programming with mathematics is fun and benefits all.

**Lesson 2: If you specify and define
a sublanguage, implement it and
exploit the restrictions in the parser.
And you need more than one.**

Implemented teaching languages ease kids into programming.

General Lessons

**Lesson 1: Your PL/IDE must support
an arithmetic of images.**

Get'em hooked. Programming with mathematics is fun and benefits all.

**Lesson 2: If you specify and define
a sublanguage, implement it and
exploit the restrictions in the parser.
And you need more than one.**

Implemented teaching languages ease kids into programming.

**Lesson 3: To help your students solve
computational design problems, you
need a systematic design process
from the *very* beginning.**

It helps you diagnose your students' problems and it empowers them so that they can solve problems on their own.

The End

with thanks to Shriram, Matthew, Robby, Kathi, and

The End

with thanks to Shriram, Matthew, Robby, Kathi, and

Dan Anderson

Karen Buras

John Burnette

Jack Clay

Robb Cutler

Marvin Hernandez

Jordan Johnson

Michael Hunt

Alvin Kroon

Lon Levy

Karen North

Todd O'Bryan

Stephen Bloch

David Kay

Gregor Kiczales

Viera Proulx

Emmanuel Schanzer

Marc Smith

Mike Sperber

Sharon Tuttle

Eli Barzilay

John Clements

Paul Graunke

Kathy Gray

Guillaume Marceau

Philippe Meunier

Paul Steckler

Danny Yoo

The End

with thanks to Shriram, Matthew, Robby, Kathi, and

Dan Anderson

Karen Buras

John Burnette

Jack Clay

Robb Cutler

Marvin Hernandez

Jordan Johnson

Michael Hunt

Alvin Kroon

Lon Levy

Karen North

Todd O'Bryan

Stephen Bloch

David Kay

Gregor Kiczales

Viera Proulx

Emmanuel Schanzer

Marc Smith

Mike Sperber

Sharon Tuttle

Eli Barzilay

John Clements

Paul Graunke

Kathy Gray

Guillaume Marceau

Philippe Meunier

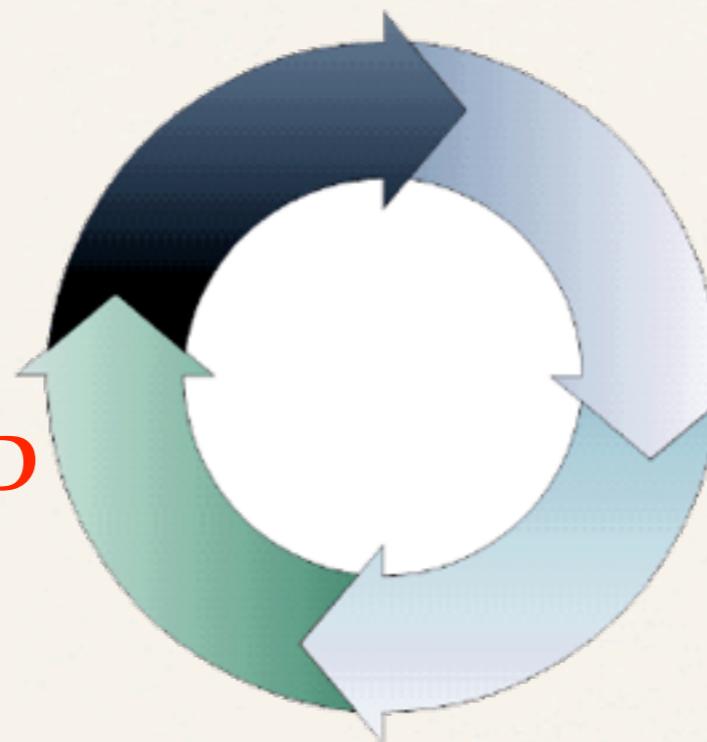
Paul Steckler

Danny Yoo

and many many more people

Research & Teaching

Teaching
at all levels
K12, college, PhD



Research
on all aspects of
PL