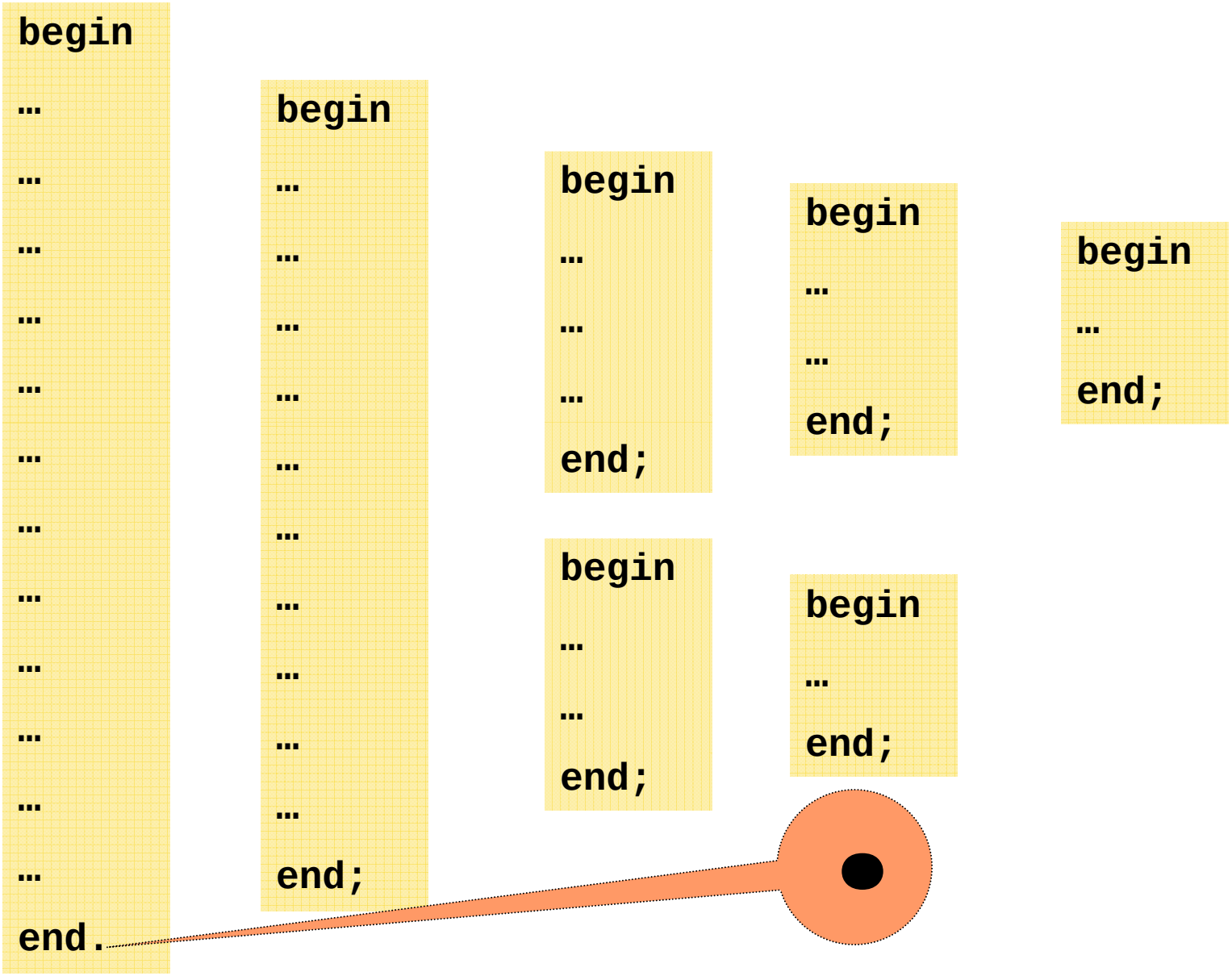
Reactive Functional Programming

(or, Is That a Parenthesis in Your Pocket?)

Shriram Krishnamurthi

BROWN

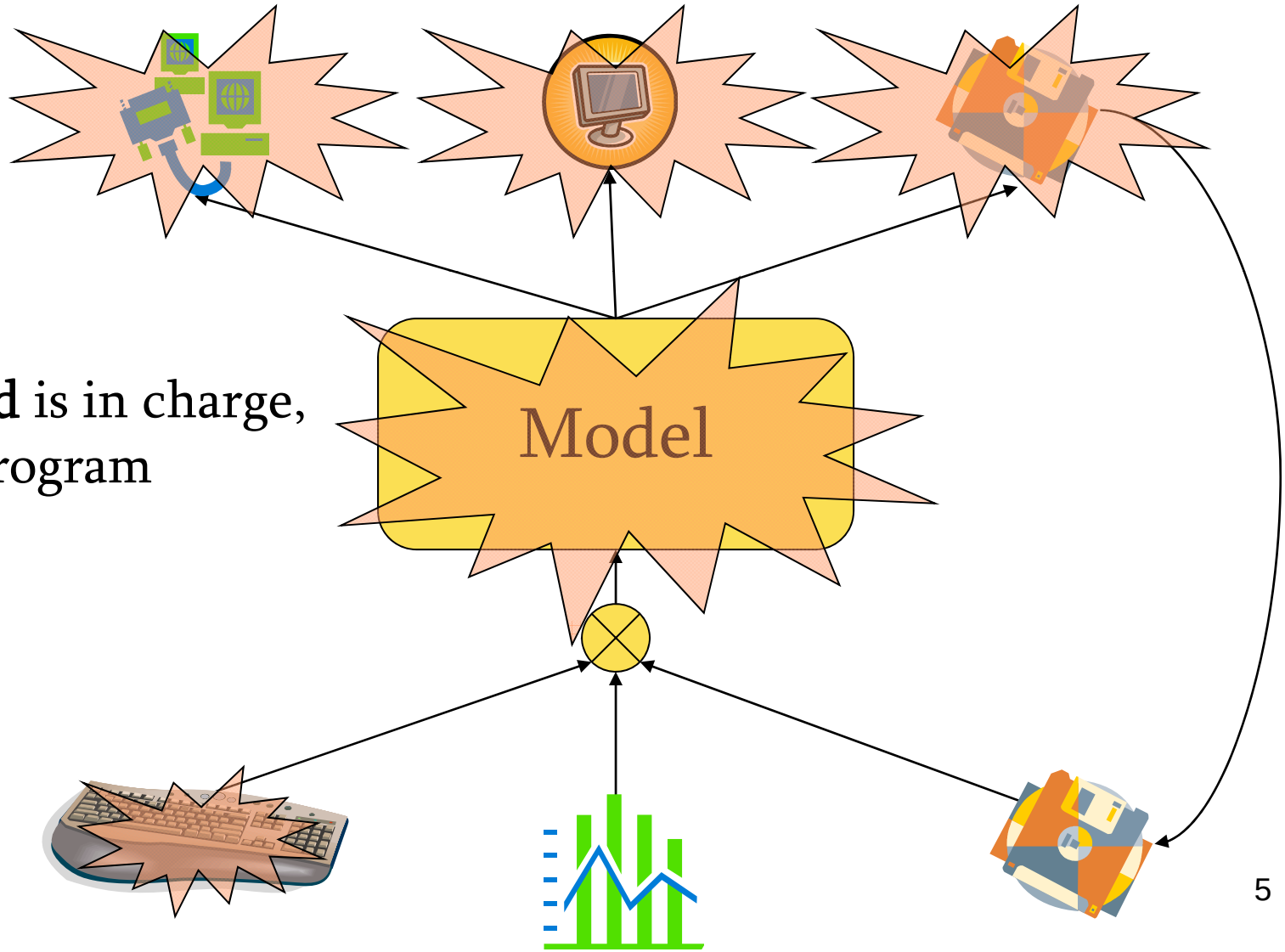
Pascal


begin
…
end.
begin
…
end;
begin
…
end;
begin
…
end;
begin
…
end;
begin
…
end;

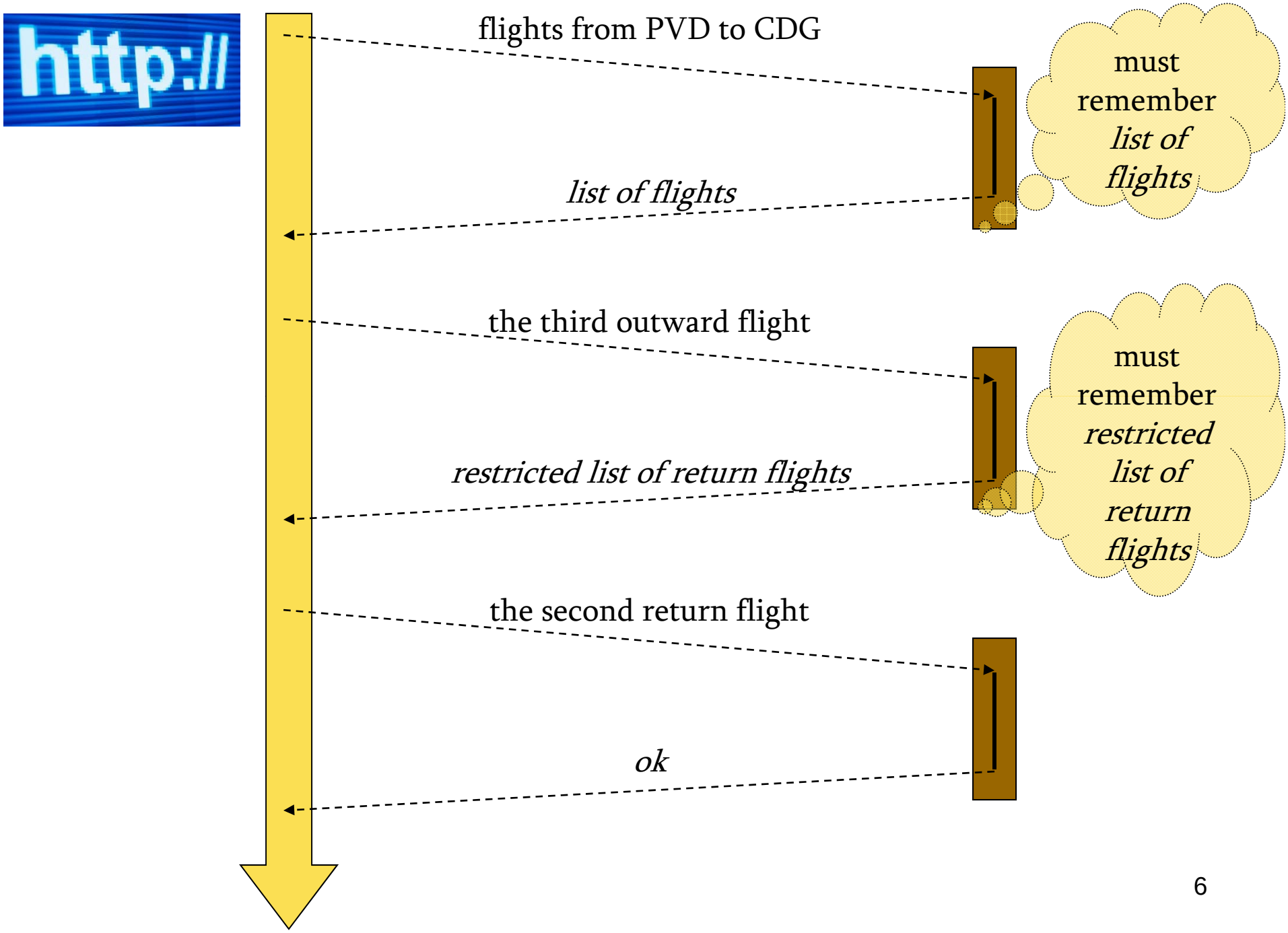
"Pascal is part of the same machinery as hall passes, dress codes, advisors' signatures, single-sex dorms, and so on."

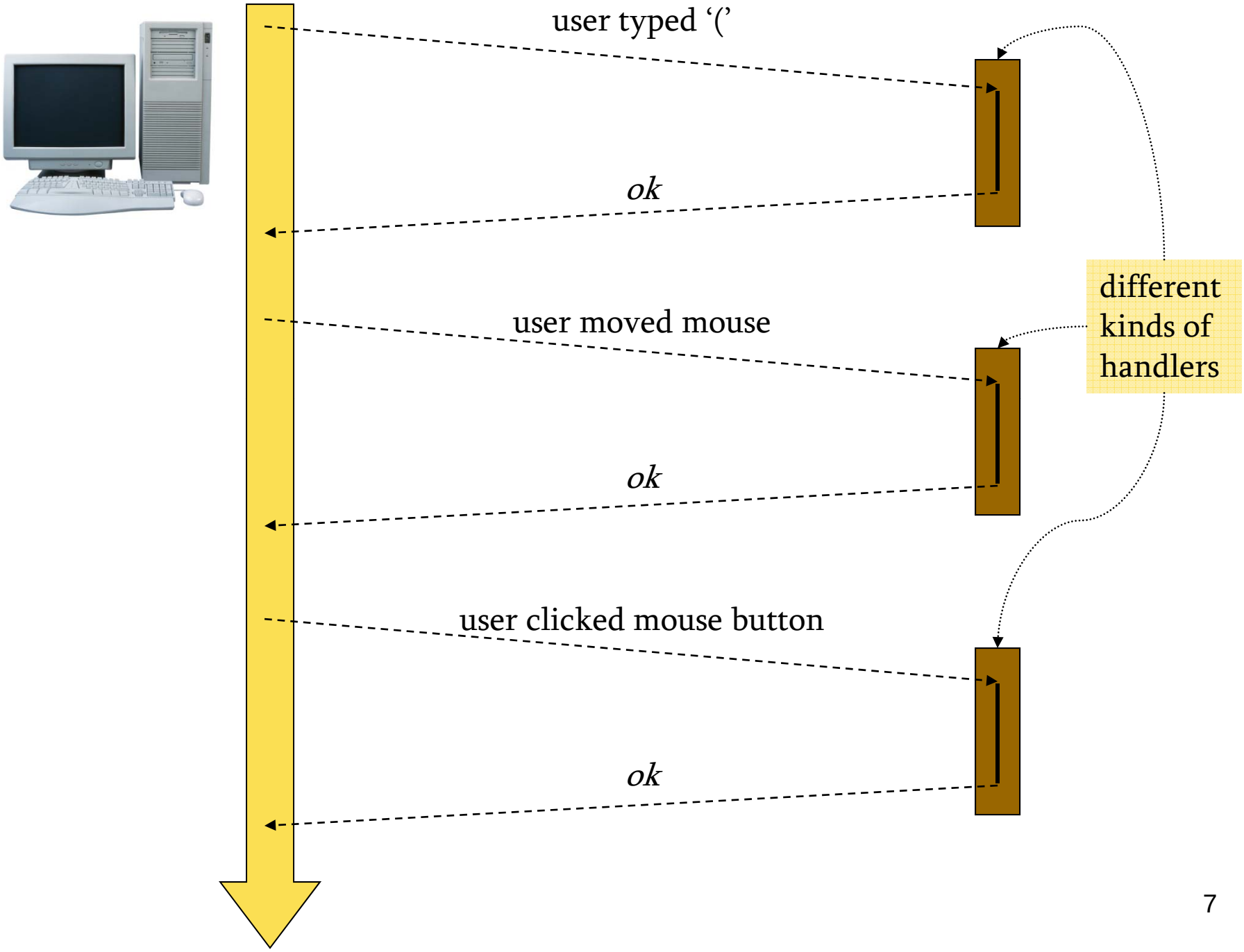
—Brian Harvey

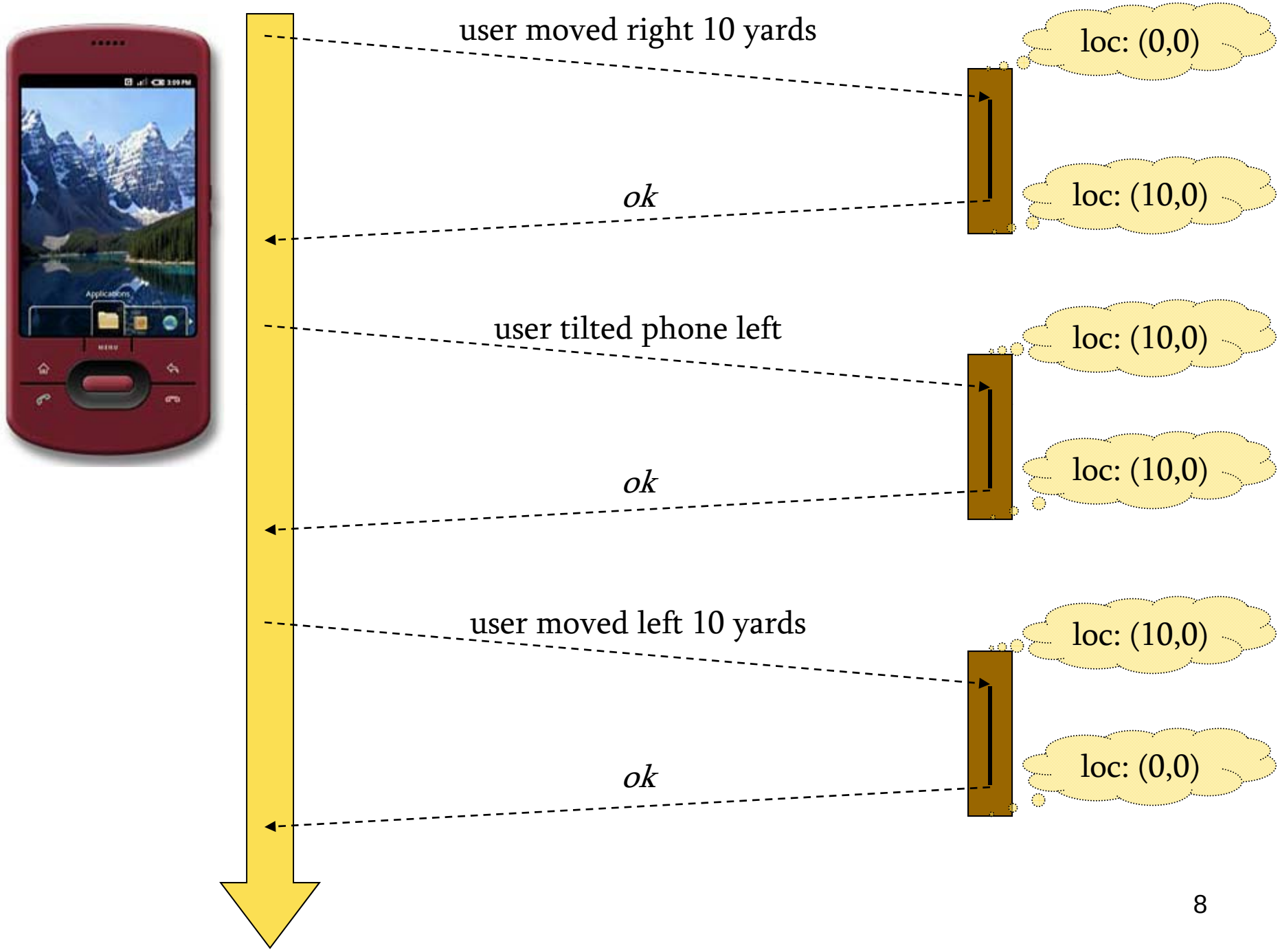
The Flow of Control

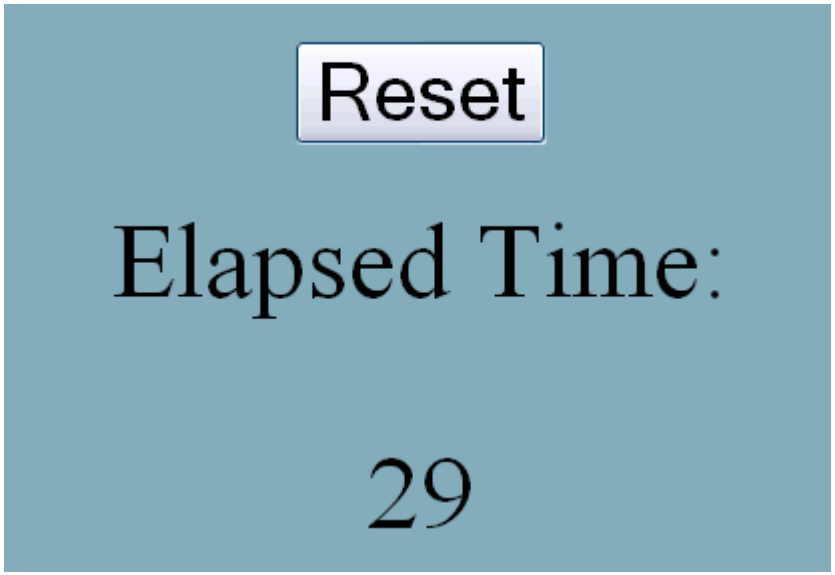


The **world** is in charge, not the program


http://
flights from PVD to CDG
must remember *list of flights*
list of flights
the third outward flight
must remember *restricted list of return flights*
restricted list of return flights
the second return flight
ok


user typed '('
ok
user moved mouse
ok
user clicked mouse button
ok
different kinds of handlers


user moved right 10 yards
loc: (0,0)
ok
loc: (10,0)
user tilted phone left
loc: (10,0)
ok
loc: (10,0)
user moved left 10 yards
loc: (10,0)
ok
loc: (0,0)

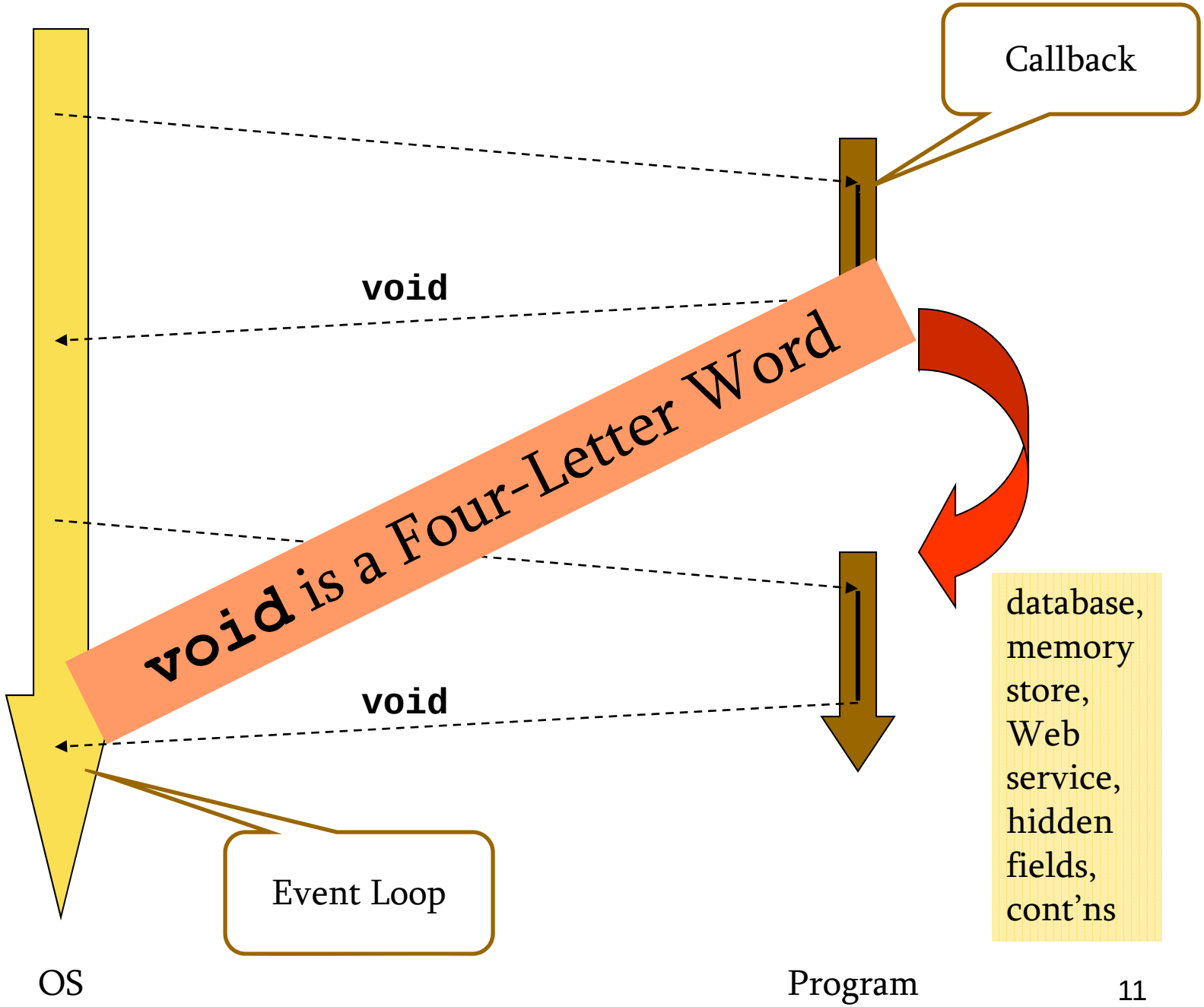

Reset
Elapsed Time:
29

timers, initialization, overlap, interference, callbacks… oh my!

```
var timerID = null;
var elapsedTime = 0;

function doEverySecond() {
  elapsedTime += 1;
  document.getElementById('curTime').innerHTML =
  elapsedTime; }
function startTimer() {
  timerId = setInterval(doEverySecond, 1000); }
function resetElapsed() {
  elapsedTime = 0; }

<body onload="startTimer()">
<input id="reset" type="button" value="Reset"
  onclick="resetElapsed()" />
<div id='curTime'></div>
</body>
```


Callback
void
void is a Four-Letter Word
void
database, memory store, Web service, hidden fields, cont'ns
Event Loop
OS
Program

```
(define gauge
  (new gauge% [label "Elapsed"]
              [range 150]))

(new timer%
  [interval 100]
  [callback (lambda (event)
              (send gauge set-value
                (add1 (send gauge get-value))))])

(new button%
  [label "Reset"]
  [callback (lambda (event)
              (send gauge set-value 0))])
```

```
(define gauge
  (new gauge% [label "Elapsed"]
              [range 150]))

(new timer%
  [interval 100]
  [callback (lambda (event)
              (send gauge set-value
                (add1 (send gauge get-value))))])

(new button%
  [label "Reset"]
  [callback (lambda (event)
              (send gauge set-value 0))])
```

```
interface ChangeListener extends EventListener {
  void stateChanged(ChangeEvent e) { ... } }

interface ActionListener extends EventListener {
  void actionPerformed(ActionEvent e) { ... } }

interface MouseListener extends EventListener {
  void mouseClicked(MouseEvent e) { ... }
  void mouseEntered(MouseEvent e) { ... }
  void mouseExited(MouseEvent e) { ... }
  void mousePressed(MouseEvent e) { ... }
  void mouseReleased(MouseEvent e) { ... } }
```

```
interface ChangeListener extends EventListener {
  void stateChanged(ChangeEvent e) { ... } }

interface ActionListener extends EventListener {
  void actionPerformed(ActionEvent e) { ... } }

interface MouseListener extends EventListener {
  void mouseClicked(MouseEvent e) { ... }
  void mouseEntered(MouseEvent e) { ... }
  void mouseExited(MouseEvent e) { ... }
  void mousePressed(MouseEvent e) { ... }
  void mouseReleased(MouseEvent e) { ... } }
```

```
mainLoop : unit -> unit
closeTk : unit -> unit

destroy : 'a Widget.widget -> unit
update : unit -> unit

pack : ... -> 'd Widget.widget list -> unit
grid : ... -> 'b Widget.widget list -> unit

raise_window : ?above:'a Widget.widget
  -> 'b Widget.widget -> unit
bind : events:event list
  -> 'a Widget.widget -> unit
```

```
mainLoop : unit -> unit
closeTk : unit -> unit

destroy : 'a Widget.widget -> unit
update : unit -> unit

pack : ... -> 'd Widget.widget list -> unit
grid : ... -> 'b Widget.widget list -> unit

raise_window : ?above:'a Widget.widget
  -> 'b Widget.widget -> unit
bind : events:event list
  -> 'a Widget.widget -> unit
```

```
propagateEvent :: IO ()
select :: Selecting w => Event w (IO ())
mouse :: Reactive w => Event w (EventMouse -> IO ())
keyboard :: Reactive w =>
  Event w (EventKey -> IO ())
resize :: Reactive w => Event w (IO ())
focus :: Reactive w => Event w (Bool -> IO ())
activate :: Reactive w => Event w (Bool -> IO ())
enter :: Reactive w => Event w (Point -> IO ())
leave :: Reactive w => Event w (Point -> IO ())
motion :: Reactive w => Event w (Point -> IO ())
drag :: Reactive w => Event w (Point -> IO ())
click :: Reactive w => Event w (Point -> IO ())
unclick :: Reactive w => Event w (Point -> IO ())
doubleClick :: Reactive w =>
  Event w (Point -> IO ())
```

```
propagateEvent :: IO ()
select :: Selecting w => Event w (IO ())
mouse :: Reactive w => Event w (EventMouse -> IO ())
keyboard :: Reactive w =>
  Event w (EventKey -> IO ())
resize :: Reactive w => Event w (IO ())
focus :: Reactive w => Event w (Bool -> IO ())
activate :: Reactive w => Event w (Bool -> IO ())
enter :: Reactive w => Event w (Point -> IO ())
leave :: Reactive w => Event w (Point -> IO ())
motion :: Reactive w => Event w (Point -> IO ())
drag :: Reactive w => Event w (Point -> IO ())
click :: Reactive w => Event w (Point -> IO ())
unclick :: Reactive w => Event w (Point -> IO ())
doubleClick :: Reactive w =>
  Event w (Point -> IO ())
```

So What?

We care deeply about **functions** because of

Testing

Composition

Education

Solutions

Lazy evaluation

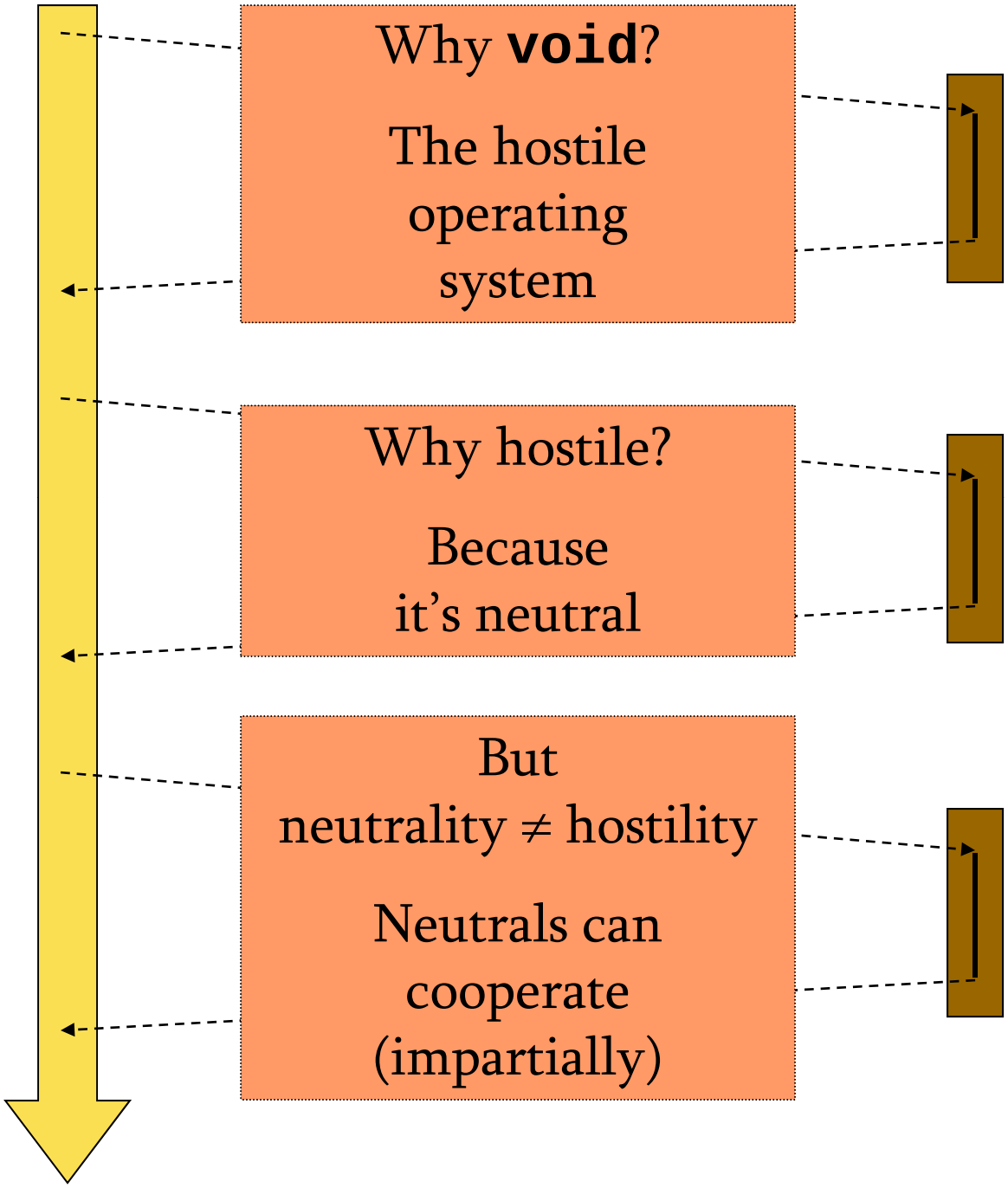
(Haskell FRP)

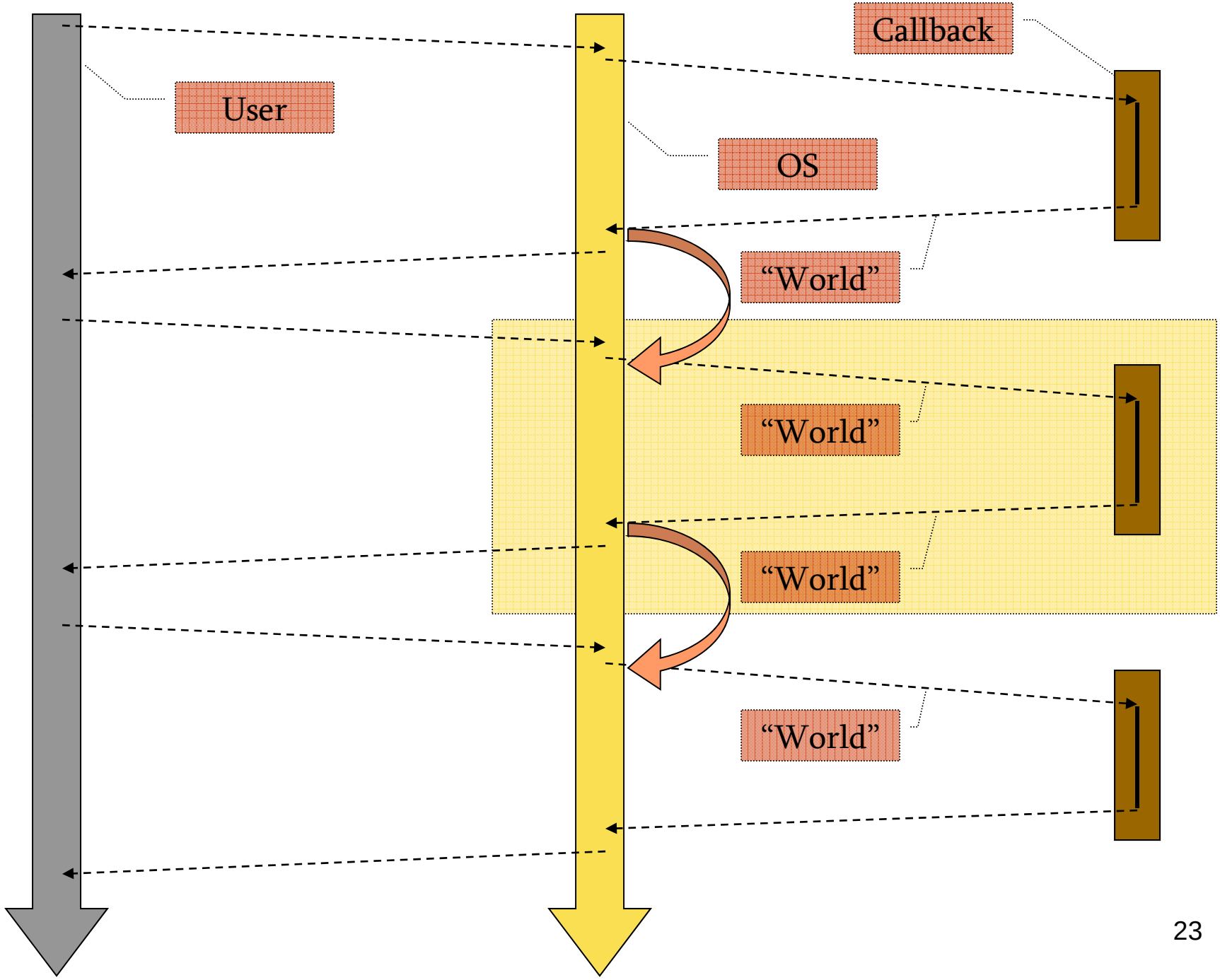
Continuations

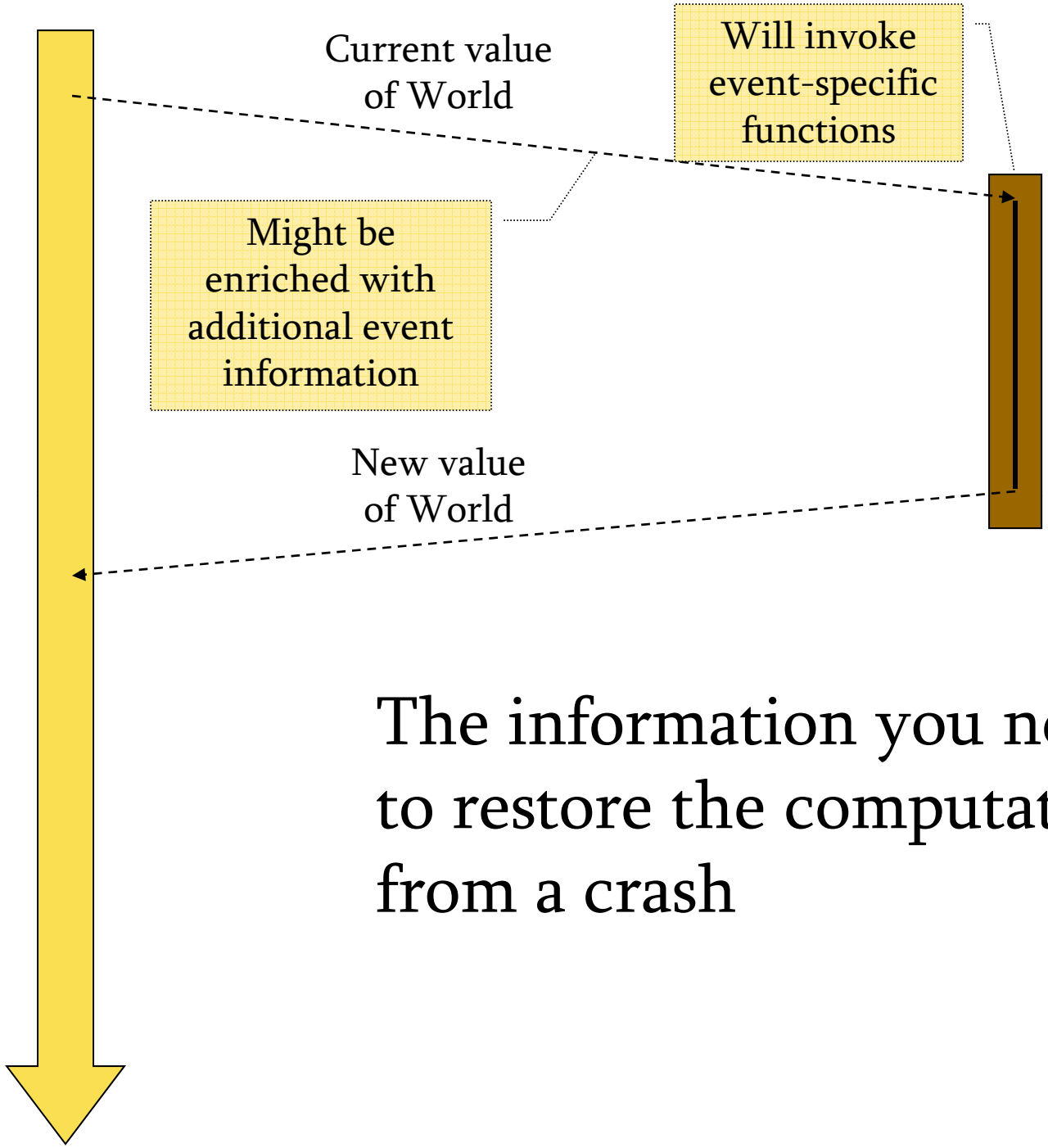
(PLT Scheme Web Server)

Dataflow + state + GUIs in call-by-value

(FrTime, Flapjax, …)


Why **void**?
The hostile operating system
Why hostile?
Because it's neutral
But neutrality ≠ hostility
Neutrals can cooperate (impartially)


Callback
User
OS
"World"
"World"
"World"
"World"



The information you need to restore the computation from a crash

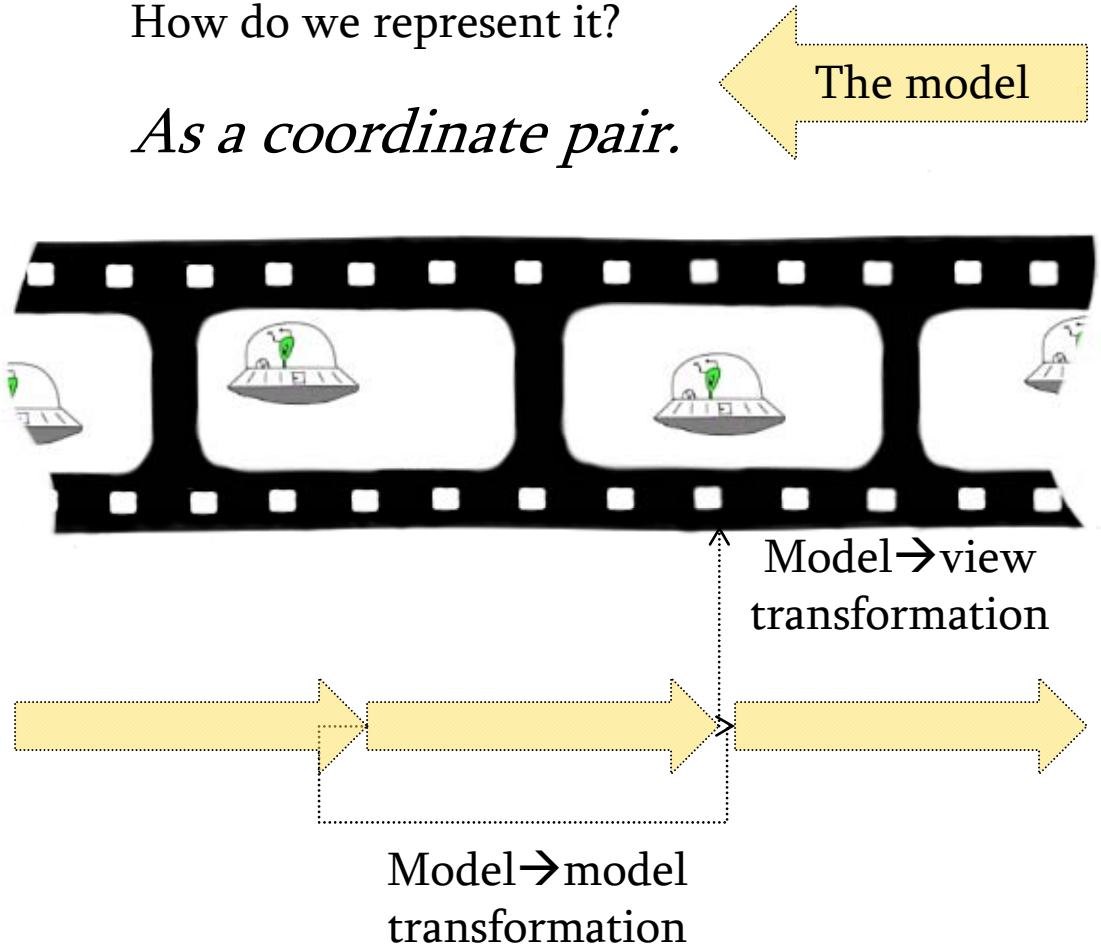
The Hollywood Principle

What's changing?

The location of the UFO.

How do we represent it?

As a coordinate pair.



```
(define width 400)
(define height 300)
(define initial-world 0)
(define PLANE  )
```

world → *scene*

```
(define (render w)
 (place-image PLANE
              (* w 10)
              (image-height PLANE)
              (empty-scene width height)))
```

world → *world*

```
(define (incr-time w)
 (add1 w))
```

```
(big-bang initial-world
          (on-tick incr-time)
          (to-draw render))
```

model → model

model → view

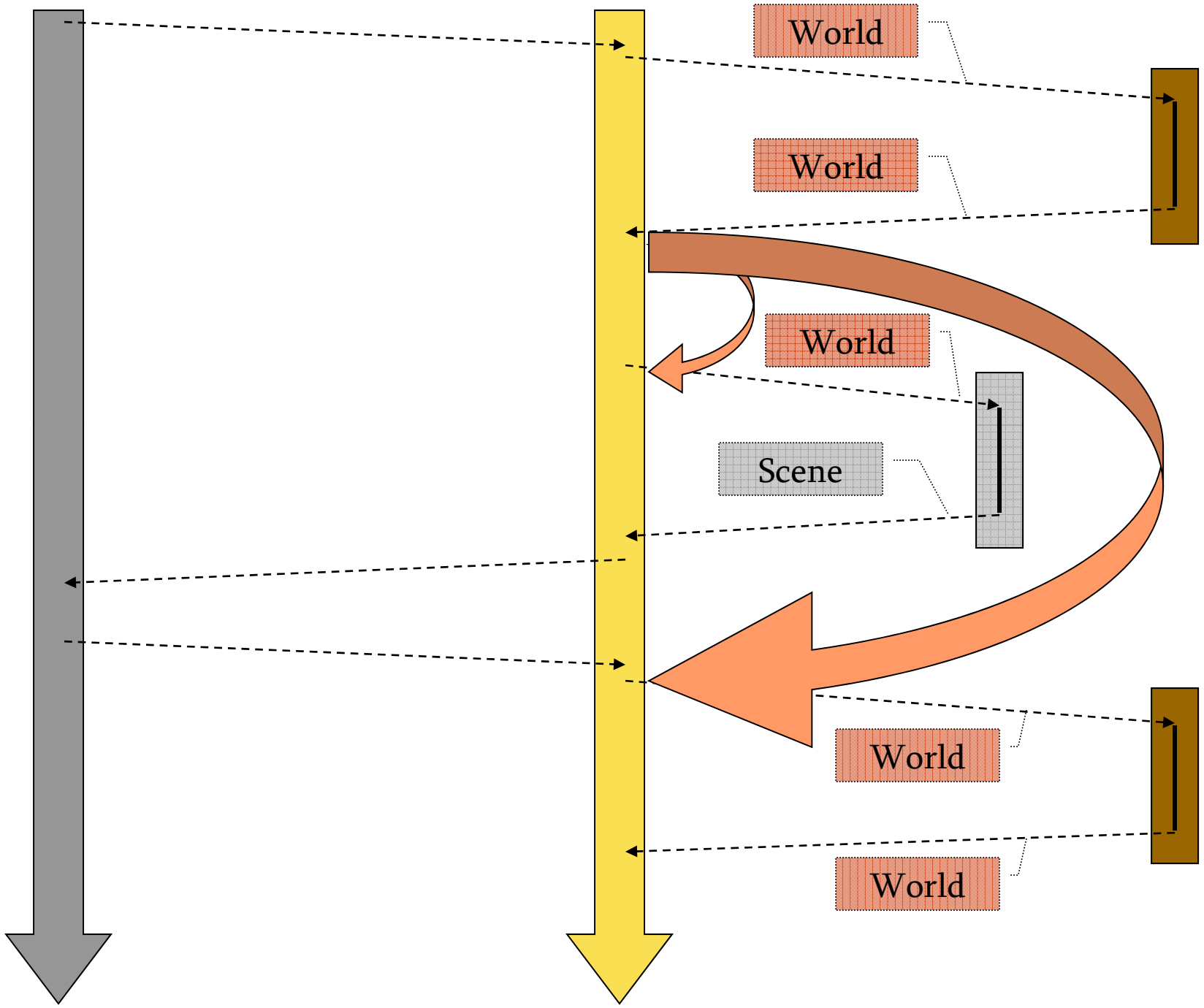
```
(define width 400)
(define height 300)
(define initial-world 0)
(define PLANE  )
```

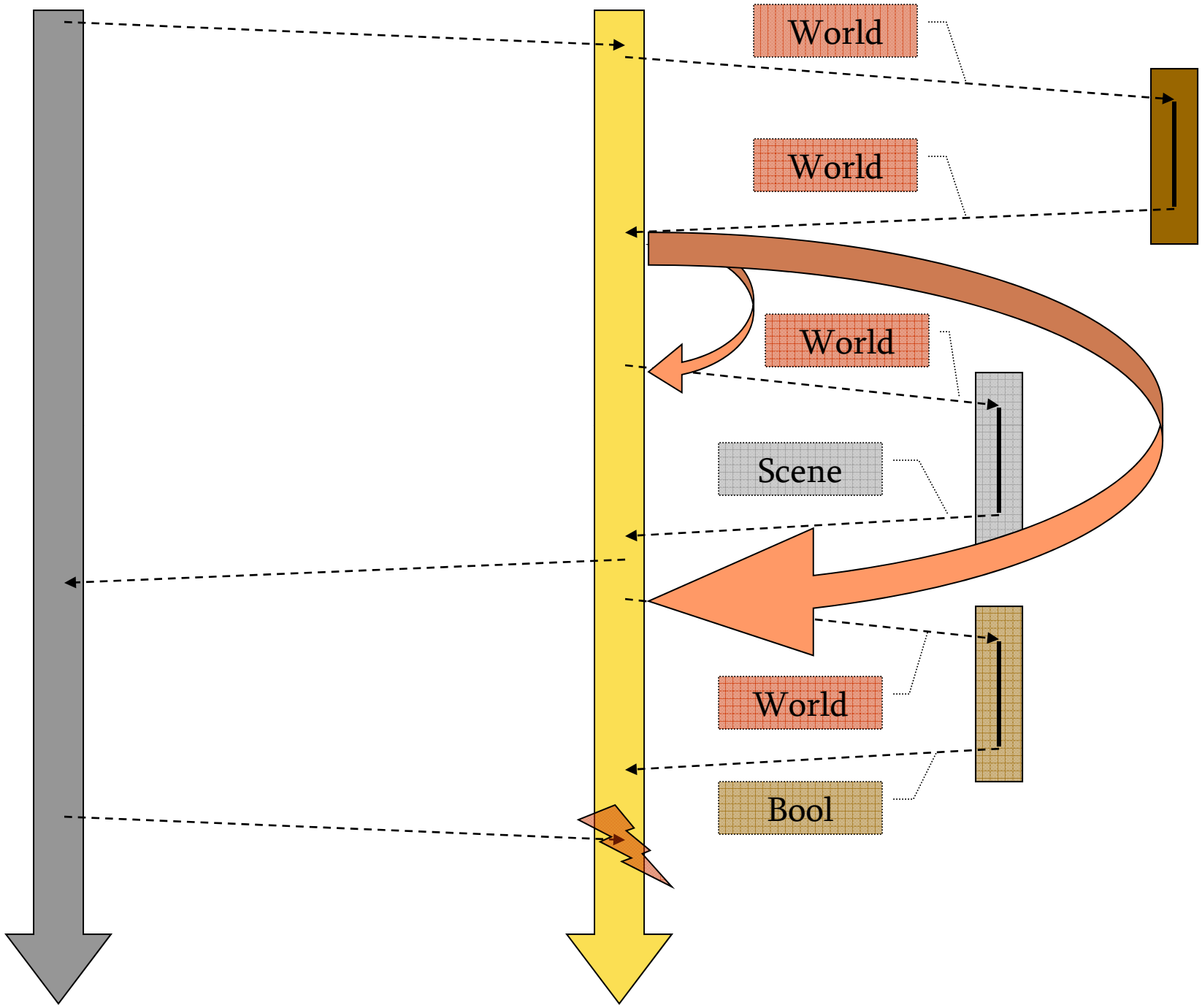
world → *scene*

```
(define (render w)
 (place-image PLANE
              (* w 10)
              (image-height PLANE)
              (empty-scene width height)))
```

```
(animate initial-world render)
```

Functions as arguments (sssh!)


World
World
World
Scene
World
World

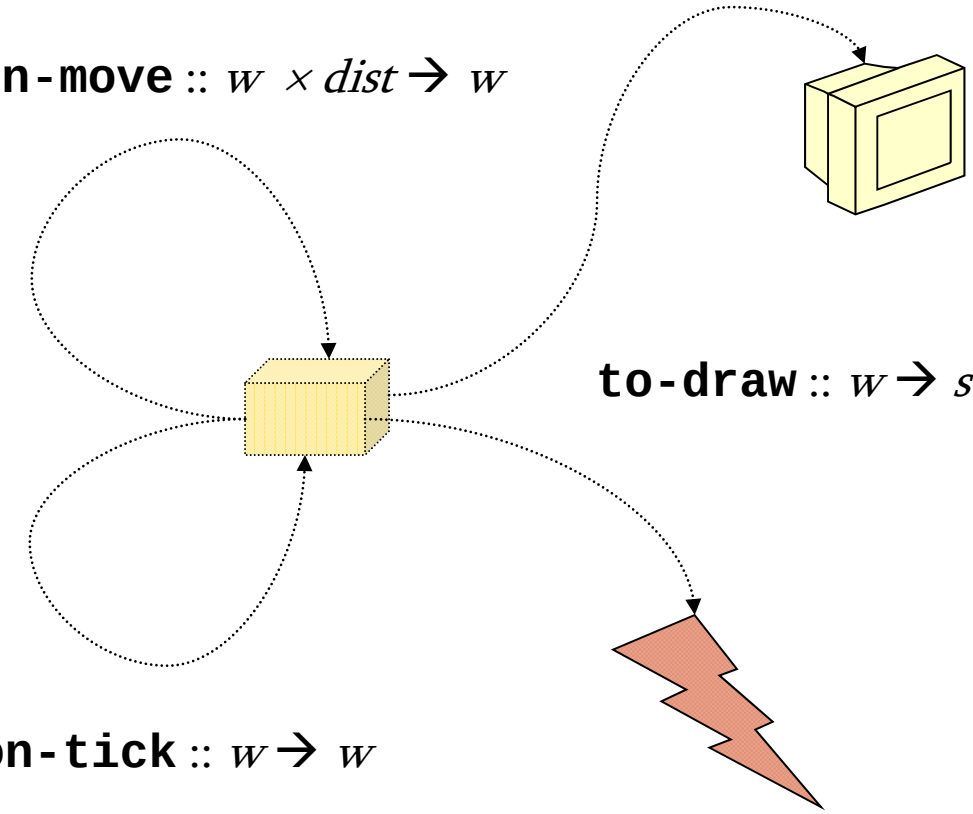

World
World
World
Scene
World
Bool

on-click :: $w \times btn \rightarrow w$

on-key :: $w \times key \rightarrow w$

on-tilt :: $w \times incl \rightarrow w$

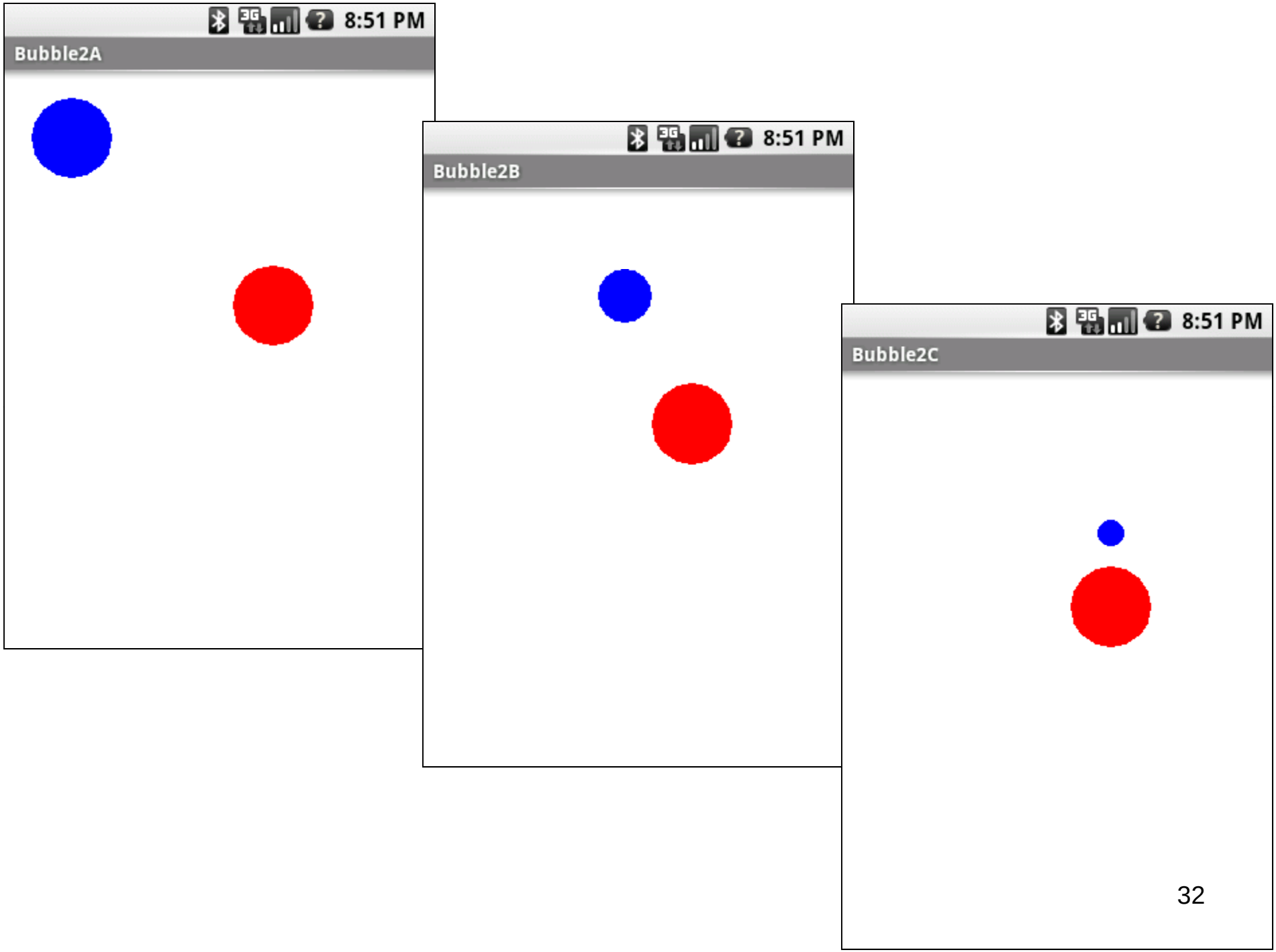
on-move :: $w \times dist \rightarrow w$



to-draw :: $w \rightarrow scene$

on-tick :: $w \rightarrow w$

stop-when :: $w \rightarrow bool$


8:51 PM
Bubble2A
8:51 PM
Bubble2B
8:51 PM
Bubble2C

```
(define WIDTH 300)
(define HEIGHT 300)

(define-struct vel (x y))

(define target (make-posn (random WIDTH) …))
```

```
(define-struct world (posn r vel))

(define initial-world
  (make-world (make-posn …)
              30 (make-vel 0 0)))
```

world → *bool*

```
(define (game-ends? w)
  (or (<= (world-r w) 1)
      (collide? w)))
```

world → *world*

```
(define (tick w)
  (make-world (posn+vel (world-posn w) (world-vel w))
              (- (world-r w) 1/3)
              (world-vel w)))
```

world × *number* × *number* × *number* → *world*

```
(define (tilt w azimuth pitch roll)
  (update-world-vel w
                    (make-vel roll (- pitch))))
```

```
(big-bang initial-world
          (to-draw render)
          (on-tick tick)
          (on-tilt tilt)
          (stop-when game-ends?))
```

CS 019

Programming with Data Structures and Algorithms

TOUR GUIDE

No doubt you remember the campus tour—the walking backwards, the explanation of meal plans, the intramural sports speech... It's a fine tradition, but the admissions department is running out of willing volunteers. That's where you come in–you've been asked to write a virtual tour guide that will run on a visitor's cell phone. Users can select the type of destinations they wish to see and receive real-time directions from their current location to the nearest destination.

Specifically, the Admissions Office wants an application with the following features:

- Your program should run on a mobile phone and give dynamically updated tour directions based on its current location relative to a map supplied by the Admissions Office (see below for the exact data definition).
- The user should be able to pick from predefined 'tours', such as a 'Campus Art Tour' or a 'Freshman Dorm Tour'. These tours will also be supplied by the Admissions Office.

BOOTSTRAP

2006 PISA scores: USA not in top 20 in math, science, or reading

"Economic Time Bomb"

—June Kronholz, *WSJ*

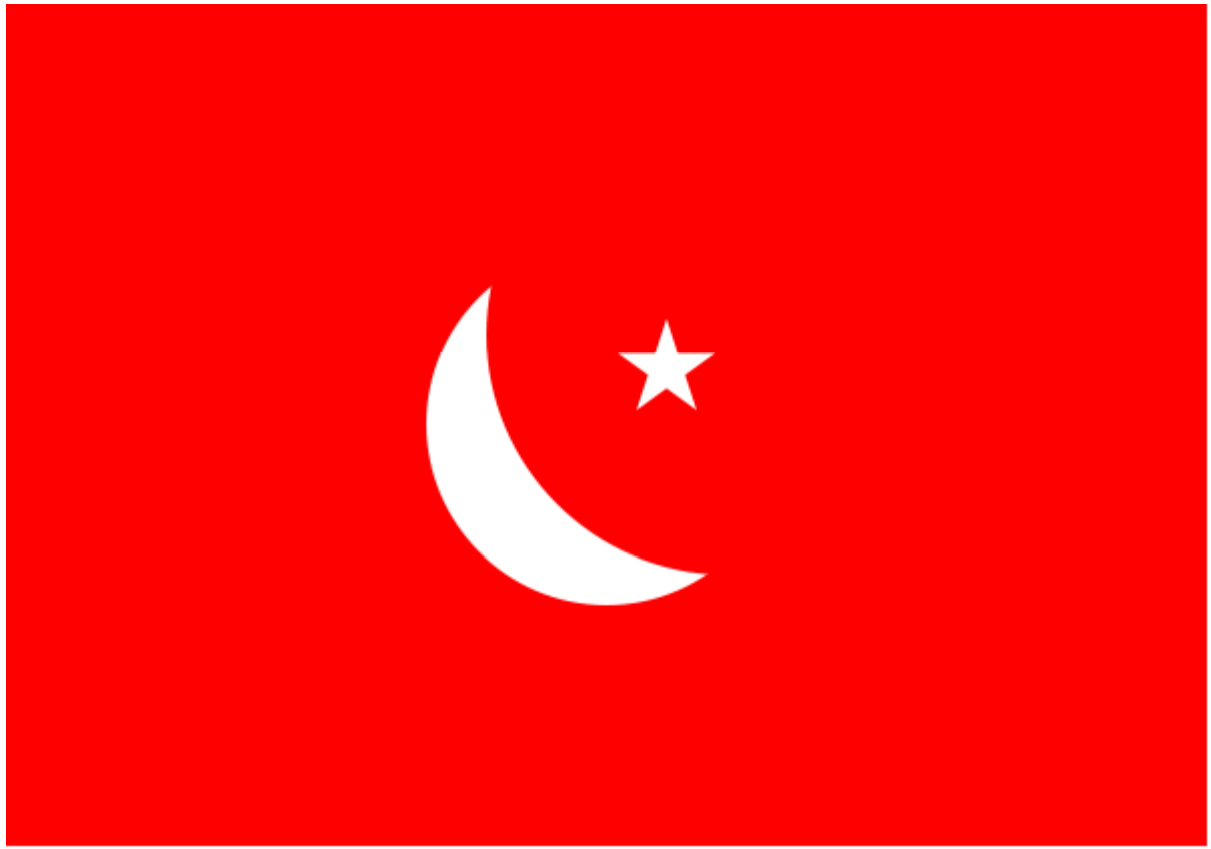
Salient Curricular Facts

STEM “sorting” occurs in middle-school

Functions are a major barrier

Algebra only over numbers is boring

A train leaves Chicago at 6pm, traveling east at 50mph…



```
(place-image (star 30 "solid" "white")
             330 180
             (place-image (circle 120 "solid" "red")
                          360 165
                          (place-image (circle 90 "solid" "white")
                                       300 210
                                       (rectangle 600 420 "solid" "red"))))
```

Early Exercise with Images: Making Fabric and Clothing

This exercise has students create images of fabrics, create clothing items from fabrics, and put logos onto clothing items. It is designed to give students practice composing functions.

Prerequisites: Numbers, Functions, Images, Defining Constants (Section 3.2)

$$5 + 3$$

$$17 = \square - 11$$

$$x^2 = 25 - 16$$

$$17 + 33 = 3x - 20$$

$$1729 = x^3 - 1000$$

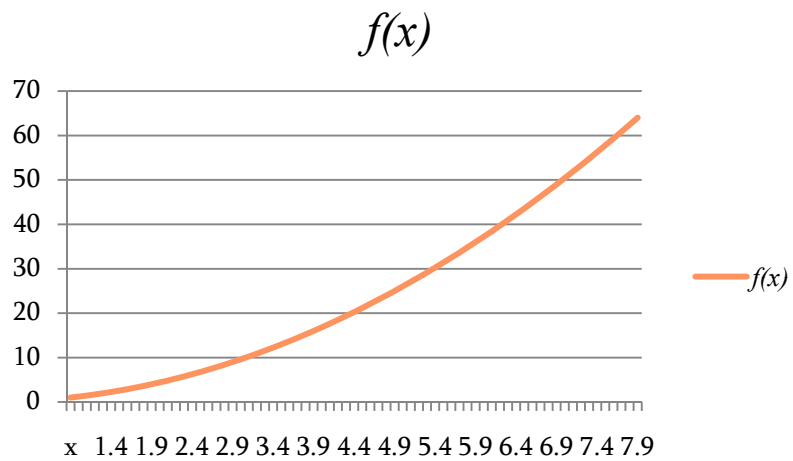
process

$$f(x) = x^2$$

object

- with attributes
- with multiple representations

Functions



$$f(x) = x^2$$

Program

x	1	2	3	4	5	6
$f(x)$	1	4	9	16	25	36

Tests

Stepper

File Edit Tabs Help

< Step | Step > | Jump... | to beginning | 1/4

```
(define (f x)
  (* x x))
(f (+ 3 4))
```

→

```
(define (f x)
  (* x x))
(f 7)
```

Stepper

File Edit Tabs Help

< Step | Step > | Jump... | to beginning | 2/4

```
(define (f x)
  (* x x))
(f 7)
```

→

```
(define (f x)
  (* x x))
(* 7 7)
```

Stepper

File Edit Tabs Help

< Step | Step > | Jump... | to beginning | 3/4

```
(define (f x)
  (* x x))
(* 7 7)
```

→

```
(define (f x)
  (* x x))
49
```

Untitled - DrRacket*

File Edit View Language Racket Insert Tabs Help

Untitled (define ...) Save Step Check Syntax Run

```
(define (f x) (* x x))
(define (g x) (* x x x))

(check-expect (f 1) 1)
(check-expect (f 2) 4)
(check-expect (f 3) 9)
(check-expect (f 4) 16)
```

Welcome to DrRacket, version 5.1 [3m].
Language: Beginning Student [custom];
memory limit: 128 MB.
Teachpack: image.ss.
All 4 tests passed!
>

Beginning Student custom 5:2

Untitled - DrRacket*

File Edit View Language Racket Insert Tabs Help

Untitled (define ...) Save Step Check Syntax Run Stop

```
(define (f x) (* x x))

(check-expect (f 1) 0)
(check-expect (f 2) 4)
(check-expect (f 3) 9)
(check-expect (f 4) 16)
```

Test Results

File Edit Tabs Help

```
Ran 4 tests.
1 of the 4 tests failed.

No signature violations.

Check failures:
                    Actua
```

Close Dock

Vital Statistics

Middle-school, after-school (predominantly)

All teaching by volunteers (professionals, students)

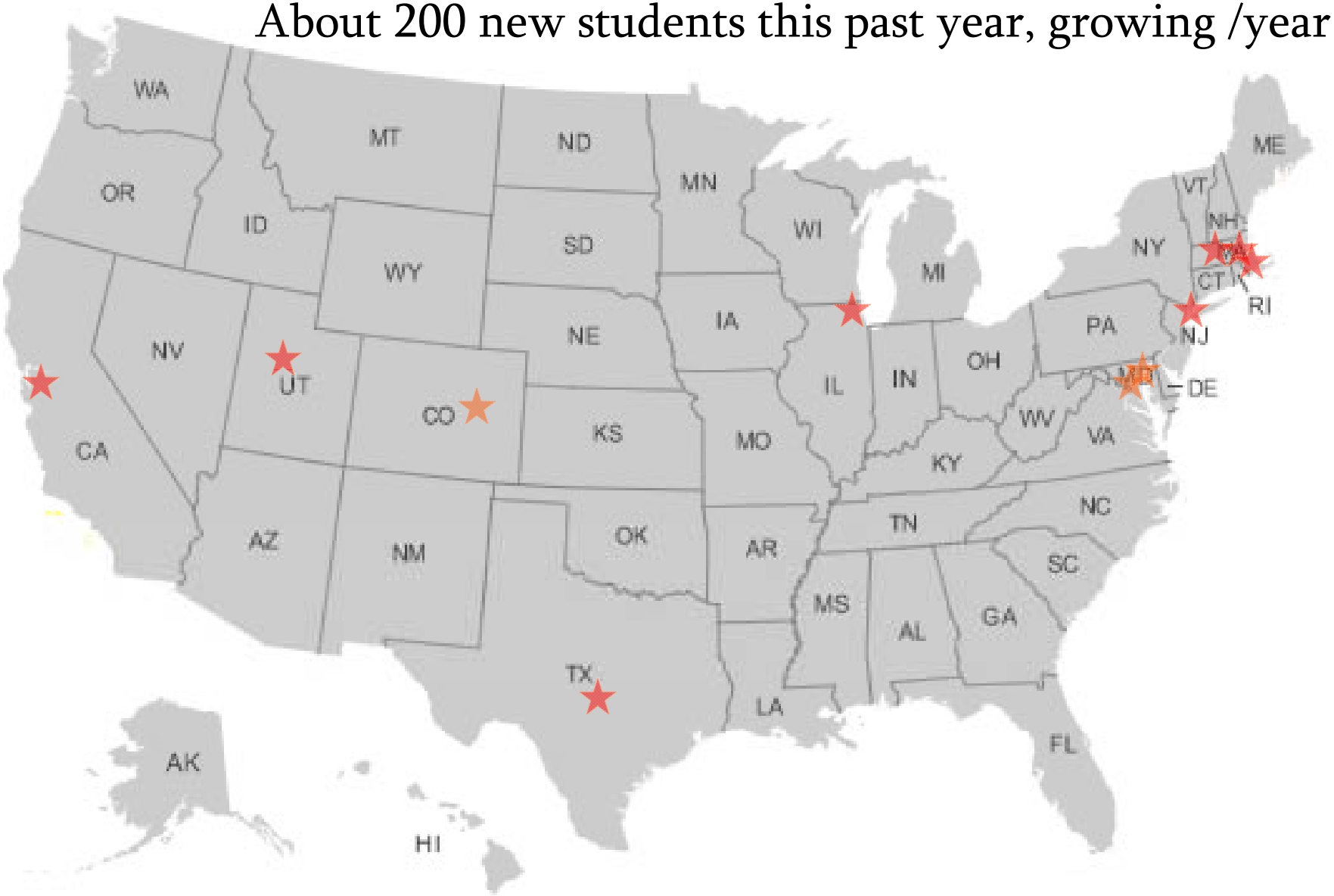
Day 1: design your own game; 9 weeks: implement it

24% female; 70% reporting race are minority;
70% on free or reduced-price lunch

Emphasis on: testing, pair-programming, code-review

Over 700 students (average age: 11y9m) so far

About 200 new students this past year, growing /year




program
ny own
games


ny own
videogame.
program
own
games.

Technology

Scheme (Racket) on the desktop

Scheme → J2ME compiler for Android

Scheme → JavaScript

- Works on both phones and browsers
- Need to deal with asynchrony
- Need to deal with re-entrance
- (Futures, continuations, …)

World ported to Racket, Java, JavaScript…

WeScheme

www.wescheme.org/openEditor?publicId=queer-decay-slush-speck-stage

WeScheme

Sometimes YouTube. Perhaps iPhone. Together, WeScheme!

Run Stop API

Project name: ticker

```
;; Simple world program.   The world is a number.

;; reset: world -> world
;; Reset the world back to zero.
(define (reset w)
  0)

;; GUI: world -> gui
(define (draw-gui w)
  (list (js-button reset '())
        (list (js-text (number->string w)))))



(big-bang 0    ;; initial world
          (on-tick add1 1)
          (on-draw draw-gui))
```

definitions

interactions

The World is Not Enough

Automatic adaptation of stateful APIs

In-place updates of world state

“Universe” for distributed computing

Moore versus Mealy

Take-Away Messages

- Programming with functions creates a virtuous cycle

- Middle-schoolers can write tests, survive code-reviews, etc.

- Programming models matter, especially across platforms

NATIONAL COUNCIL OF
TEACHERS OF MATHEMATICS

www.bootstrapworld.org

www.facebook.com/BootstrapWorld

Thanks:

Danny Yoo	Emmanuel Schanzer
Kathi Fisler	Matthias Felleisen
Zhe Zhang	**(rest)**