# Two Draggable Cats

CS 5010 Program Design Paradigms

Lesson 3.4

# Introduction and Learning Objectives

- In this lesson, you will learn how to build more complicated worlds with more than one object.
- By the end of this lesson you should be able to
  - Write more complex data definitions, representing information in appropriate places.
  - Use templates to guide the development of programs incorporating multiple data definitions.

# Requirements

- Like draggable-cat, except:
- We have 2 cats in the scene
- Each cat can be individually selected, as in draggable-cat
- Space pauses or unpauses the entire animation
- Demo: two-draggable-cats: http://www.youtube.com/watch?v=XvODwv7ivrA

# two-draggable-cats: demo

- https://www.youtube.com/watch?v=XvODwv7ivrA

Note: I've added a bunch of tests since this video was made.  Study them!

YouTube link

# Information Analysis

- The world has two cats and a paused?
  - it is the whole world that is paused or not

# Data Definitions: World

```
(define-struct world (cat1 cat2 paused?))
;; A World is a (make-world Cat Cat Boolean)
;; cat1 and cat2 are the two cats
;; paused? describes whether or not the world
;; is paused

;; template:
;; world-fn : World -> ??
;; (define (world-fn w)
;;   (... (world-cat1 w)
;;        (world-cat2 w)
;;        (world-paused? w)))
```

# Information Analysis

- Each cat has x-pos, y-pos, and selected?
- What about paused?
  - cats aren't individually paused
  - it's the whole thing that is paused or not.

# Data Definitions: Cat

```
(define-struct cat (x-pos y-pos selected?))
;; A Cat is a
;;   (make-cat Integer Integer Boolean)
;; Interpretation:
;; x-pos, y-pos give the position of the cat.
;; selected? describes whether or not the cat is
;; selected.

;; template:
;; cat-fn : Cat -> ??
;(define (cat-fn c)
; (... (cat-x-pos w)
;      (cat-y-pos w)
;      (cat-selected? w)))
```

# Data Design Principles

- Every value of the information should be represented by some value of the data
  - otherwise, we lose immediately!
- Every value of the data should represent some value of the information
  - no meaningless or nonsensical combinations
  - if each cat had a paused? field, then what does it mean for one cat to be paused and the other not?
  - Is it possible for one cat to be paused and the other not?

# Follow the template!

- If your world has some cats in it, then your world function will just call a cat function on each cat.
- The structure of your program will follow the structure of your data definitions.
- Let's watch this at work:

# world-after-tick

```
;; world-after-tick : World -> World
;; RETURNS: the world that should follow the
;; given world after a tick
;; STRATEGY: Use template for World on w

(define (world-after-tick w)
  (if (world-paused? w)
      w
      (make-world
        (cat-after-tick (world-cat1 w))
        (cat-after-tick (world-cat2 w))
        false)))
```

**(world-cat1 w)** is a cat, so we just call a cat function on it

# cat-after-tick

```
;; cat-after-tick : Cat -> Cat
;; RETURNS: the state of the given cat after a tick in an
;; unpaused world.

;; EXAMPLES:
;; cat selected
;; (cat-after-tick selected-cat-at-20) = selected-cat-at-20
;; cat paused:
;; (cat-after-tick unselected-cat-at-20) = unselected-cat-at-28

;; STRATEGY: Use template for Cat on c

;; function definition on next slide
```

# cat-after-tick definition

```
(define (cat-after-tick c)
  (if (cat-selected? c)
    c
    (make-cat
      (cat-x-pos c)
      (+ (cat-y-pos c) CATSPEED)
      (cat-selected? c))))
```

# world-to-scene

- world-to-scene follows the same pattern:  the world consists of two cats, so we call two cat functions.
- Both cats have to appear in the same scene, so we will have to be a little clever about our cat function.

# world-to-scene

```
;; world-to-scene : World -> Scene
;; RETURNS: a Scene that portrays the
;;   given world.
;; STRATEGY: Use template for World on w
(define (world-to-scene w)
  (place-cat
    (world-cat1 w)
    (place-cat
      (world-cat2 w)
      EMPTY-CANVAS)))
```

The pieces are cats, so create a wishlist function to place a cat on a scene

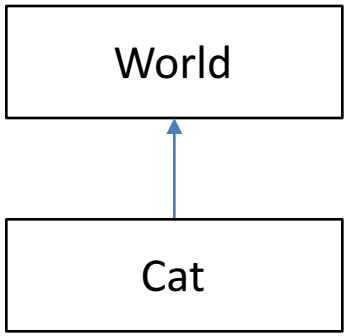
# place-cat

```
;; place-cat : Cat Scene -> Scene
;; returns a scene like the given one, but with
;; the given cat painted on it.
;; strategy : Use template for Cat on c
(define (place-cat c s)
  (place-image
    CAT-IMAGE
    (cat-x-pos c) (cat-y-pos c)
    s))
```

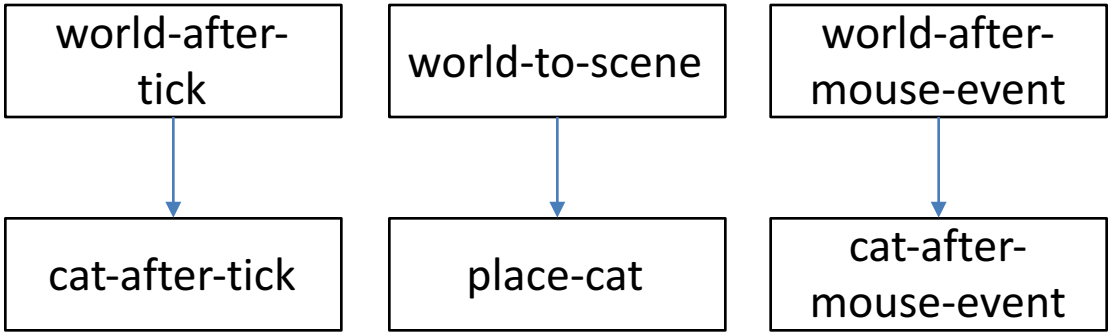
# The Structure of the Program Follows the Structure of the Data (1)

- Let's look again at the structure of our program.
- If we draw the call graph of our program (showing which functions call which), we can see that the call graph mirrors the structure of the data
- The world contains two cats, so world-after-tick calls cat-after-tick (twice).
- Let' draw some pictures:

# The Structure of the Program Follows the Structure of the Data (2)

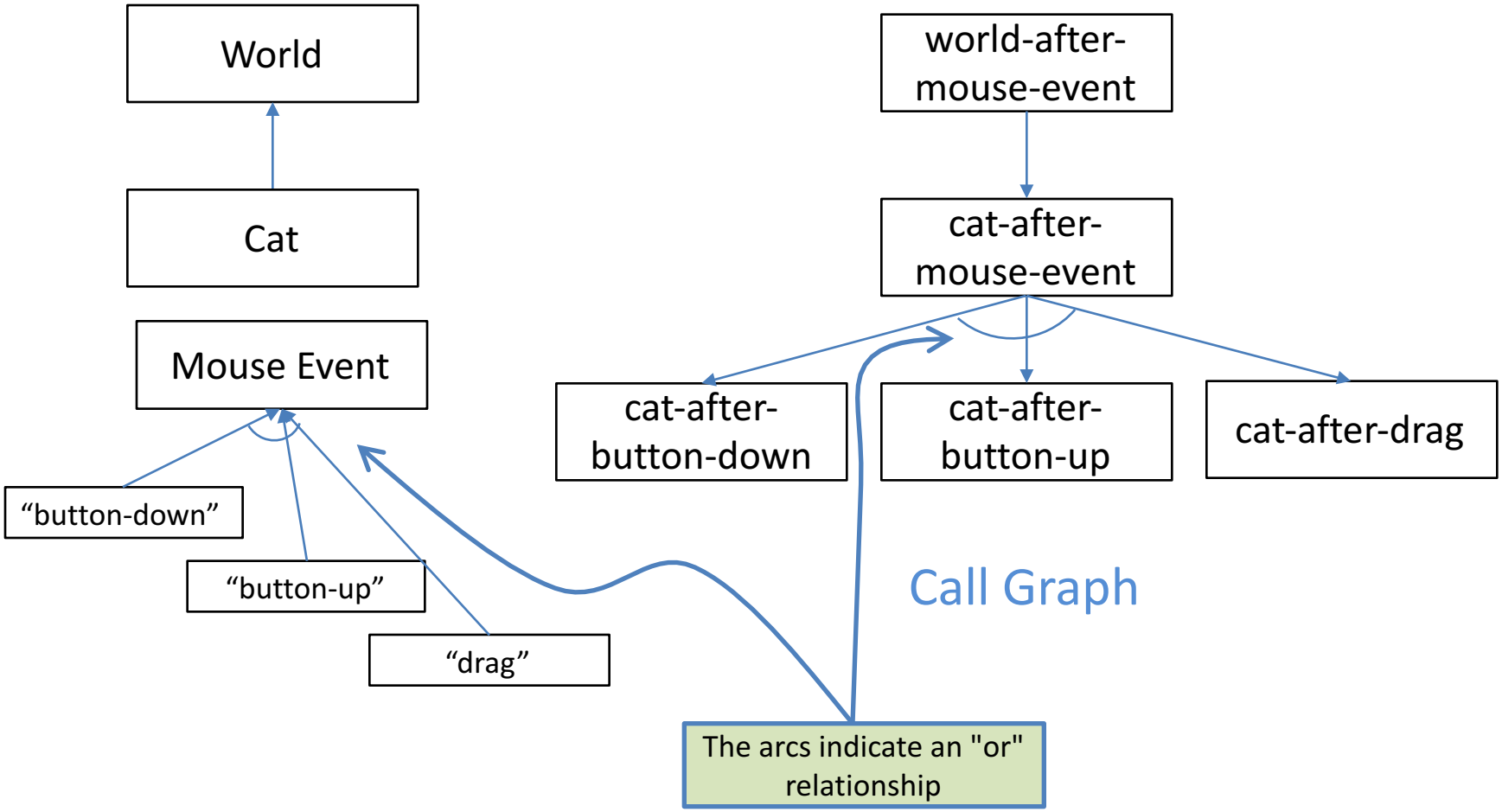



Data Definitions

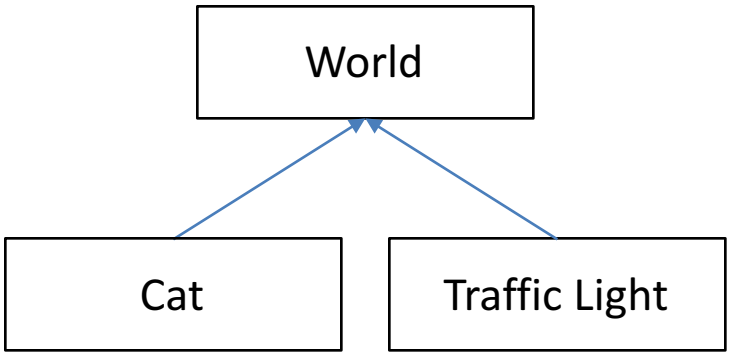



Call Graphs

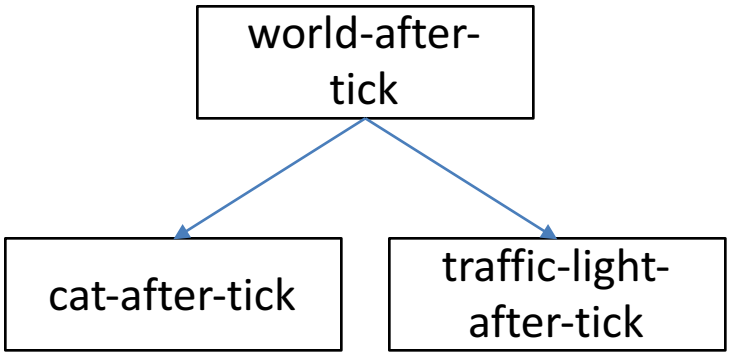
# The Structure of the Program Follows the Structure of the Data (3)



# What if there were more things in the world?




Data Definitions

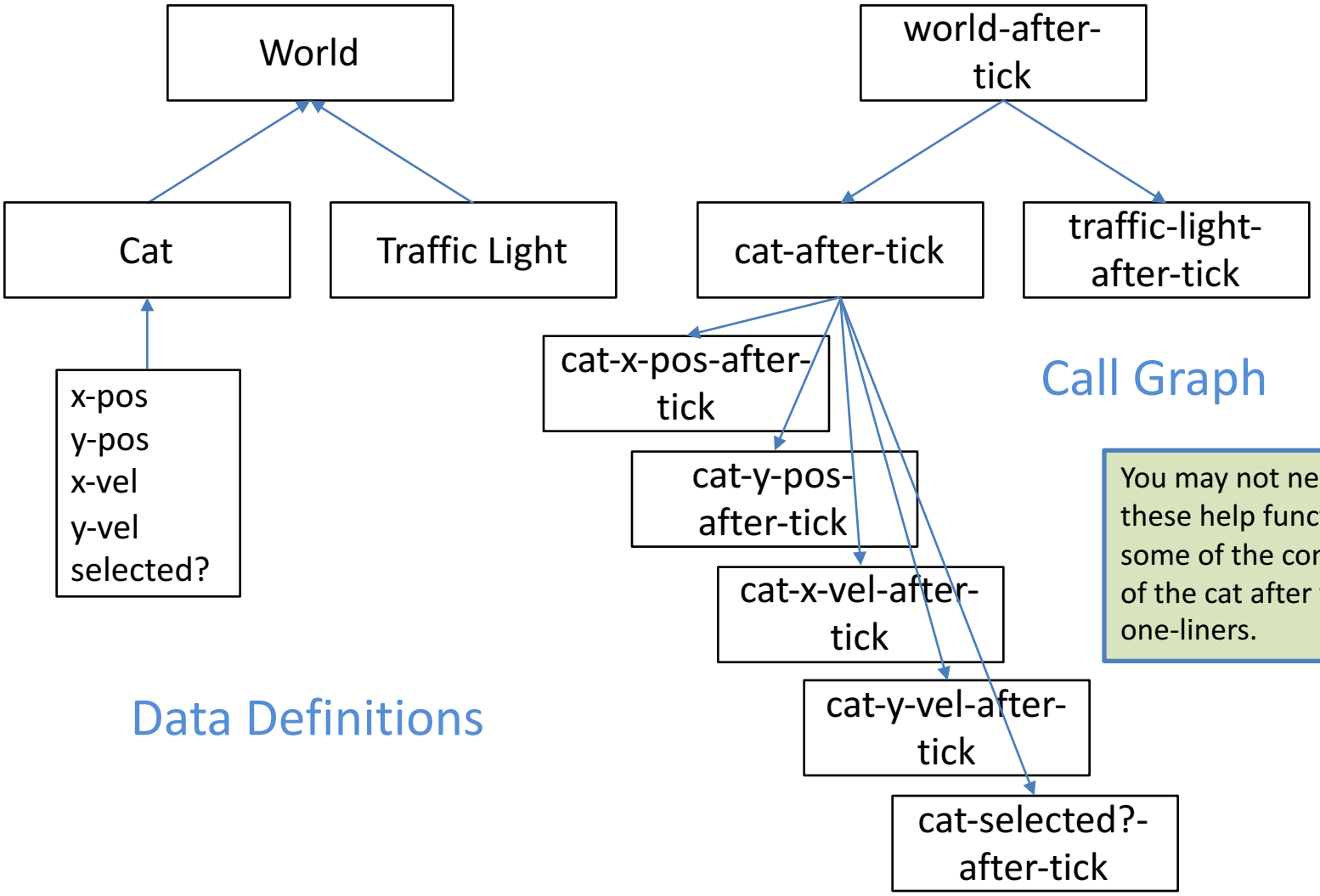



Call Graph

# What if the motion of the cat were more complicated?

- In our problem, the components of the new cat were all "one-liners"
- If the motion of the cat were more complicated, you might need to do some complicated computation to determine the next x,y position and next x,y velocities of the cat.
- You'd turn some or all of these into help functions.
- This still still winds up following the structure of the data:

# What if the motion of the cat were more complicated? (2)




Data Definitions

Call Graph

You may not need all of these help functions if some of the components of the cat after the tick are one-liners.

# Summary

- In this lesson, you had the opportunity to
  - Build a more complex world
  - Write more complex data definitions, representing information in appropriate places.
  - Use the structure of the data to guide the development of programs incorporating multiple data definitions ("the structure of the program follows the structure of the data").

# Next Steps

- Run two-draggable-cats.rkt and study the code (including the tests!)
- If you have questions about this lesson, ask them on the Discussion Board