CS5010 - Problem Set 10 - Test Results

pdp-group-dantuluru-rosefox911

April 8, 2014

This test suite tests your implementation of Problem Set 09

1 File: rectangles.rkt

Tests your bouncing rectangles problem
Common Definitions

```
(define CX 200)

(define CY 250)

(define SPEED 13)

(define LBORDER 15)

(define RBORDER 385)

(define check-world-selected?
(lambda (t w) (equal? t (send w get-selected?))))

(define check-world-x (lambda (t w) (equal? t (send w get-x))))

(define check-world-y (lambda (t w) (equal? t (send w get-y))))

(define check-rectangle-selected?
(lambda (t r) (equal? t (send r is-selected?))))

(define check-rectangle-x (lambda (t r) (equal? t (send r get-x))))
```

```
(define check-rectangle-y (lambda (t r) (equal? t (send r get-y))))


(define check-rectangles
(lambda (lst w)
(let ((rmap (map map-rect-components (send w get-shapes))))
(if (or (equal? lst rmap) (equal? (reverse lst) rmap))
#t
rmap))))


(define map-rect-components
(lambda (r)
(list (send r get-x) (send r get-y) (send r is-selected?))))


(define check-rectangle
(lambda (x y sel?)
(lambda (w)
(and (check-rectangle-x x w)
     (check-rectangle-y y w)
     (check-rectangle-selected? sel? w)))))


(define check-world
(lambda (x y sel? w)
(and (check-world-x x w)
     (check-world-y y w)
     (check-world-selected? sel? w))))


(define rectangle-equal?
(lambda (r1 r2)
(and (equal? (send r1 get-x) (send r2 get-x))
     (equal? (send r1 get-y) (send r2 get-y))
     (equal? (send r1 is-selected?) (send r2 is-selected?)))))


(define world-equal?
(lambda (w1 w2)
(and (equal? (send w1 get-x) (send w2 get-x))
     (equal? (send w1 get-y) (send w2 get-y))
     (equal? (send w1 get-selected?) (send w2 get-selected?))
     (andmap
      rectangle-equal?
      (send w1 get-shapes)
      (send w2 get-shapes)))))
```

```
(define simulate-until-at-wall
(lambda (right? speed w)
(let ((cur-x (send (first (send w get-shapes)) get-x)))
(cond
 ((or (> cur-x RBORDER) (< cur-x LBORDER))
  (error "Moved past the edge"))
((and right? (or (= cur-x RBORDER) (= cur-x (- RBORDER 1)))) w)
((and (not right?) (= cur-x LBORDER)) w)
(else
 (begin
(send w on-tick)
(let* ((next-x (send (first (send w get-shapes)) get-x)))
(if (or (and right? (> next-x cur-x))
        (and (not right?) (< next-x cur-x)))
(if (or (equal? (abs (- next-x cur-x)) speed)
        (= next-x RBORDER)
        (= next-x (- RBORDER 1))
        (= next-x LBORDER))
 (begin (simulate-until-at-wall right? speed w))
 (error "Does not move at full speed when it should"))
(error "Does not move towards correct wall")))))))))
```

1.1 Test-Group: Basic (non-)reactions (1 Points)

1/1

Common Definitions

```
(define INITIAL-WORLD-ORIG (make-world-1 SPEED))
```

1.1.1 Test (state)

Common Definitions

```
(define INITIAL-WORLD (make-world-1 SPEED))
```

1.1.2 Test (action/test)

Action:

```
(send INITIAL-WORLD on-tick)
```

Test:

```
(world-equal? INITIAL-WORLD-ORIG INITIAL-WORLD)
```

Correct

1.1.3 Test (state)

Common Definitions

```
(define INITIAL-WORLD (make-world-1 SPEED))
```

1.1.4 Test (action/test)

Action:

```
(send INITIAL-WORLD on-mouse 10 10 "button-down")
```

Test:

```
(world-equal? INITIAL-WORLD-ORIG INITIAL-WORLD)
```

Correct

1.1.5 Test (state)

Common Definitions

```
(define INITIAL-WORLD (make-world-1 SPEED))
```

1.1.6 Test (action/test)

Action:

```
(send INITIAL-WORLD on-mouse CX CY "drag")
```

Test:

```
(world-equal? INITIAL-WORLD-ORIG INITIAL-WORLD)
```

Correct

1.1.7 Test (state)

Common Definitions

```
(define INITIAL-WORLD (make-world-1 SPEED))
```

1.1.8 Test (action/test)

Action:

```
(send INITIAL-WORLD on-key "d")
```

Test:

```
(world-equal? INITIAL-WORLD-ORIG INITIAL-WORLD)
```

Correct

1.1.9 Test (state)

Common Definitions

```
(define BALL-WORLD
(let ((w (make-world-1 SPEED))) (send w on-key "r") w))
```

1.1.10 Test (action/test)

Action:

```
BALL-WORLD
```

Test:

```
(string=?
(send (first (send BALL-WORLD get-shapes)) get-color)
"green")
```

Correct

1.2 Test-Group: Dragging the target (1 Points)

1/1

1.2.1 Test (state)

Common Definitions

```
(define drag-world (make-world-1 SPEED))
```

1.2.2 Test (action/test)

Action:

```
(send drag-world on-mouse (+ CX 3) (- CY 4) "button-down")
```

Test:

```
(check-world CX CY true drag-world)
```

Correct

1.2.3 Test (action/test)

Action:

```
(send drag-world on-mouse 50 150 "drag")
```

Test:

```
(check-world 47 154 true drag-world)
```

Correct

1.2.4 Test (action/test)

Action:

```
(send drag-world on-mouse 300 25 "drag")
```

Test:

```
(check-world 297 29 true drag-world)
```

Correct

1.2.5 Test (action/test)

Action:

```
(send drag-world on-mouse 300 25 "button-up")
```

Test:

```
(check-world 297 29 false drag-world)
```

Correct

1.3 Test-Group: Basic ball behavior (1 Points)

1/1

1.3.1 Test (state)

Common Definitions

```
(define BALL-WORLD
(let ((w (make-world-1 SPEED))) (send w on-key "r") w))
```

1.3.2 Test (action/test)

Action:

```
BALL-WORLD
```

Test:

```
(check-rectangles `((,CX ,CY ,false)) BALL-WORLD)
```

Correct

1.3.3 Test (action/test)

Action:

```
(send BALL-WORLD on-tick)
```

Test:

```
(check-rectangles `((,(+ CX SPEED) ,CY ,false)) BALL-WORLD)
```

Correct

1.3.4 Test (action/test)

Action:

```
(send BALL-WORLD on-tick)
```

Test:

```
(check-rectangles `((,(+ CX SPEED SPEED) ,CY ,false)) BALL-WORLD)
```

Correct

1.4 Test-Group: Bouncing (1 Points)

1/1

Common Definitions

```
(define SPEED 13)
```

1.4.1 Test (or)

Right bounce
Test (state)
Common Definitions

```
(define BALL-WORLD
(let ((w (make-world-1 SPEED))) (send w on-key "r") w))
```

1.4.2 Test (action/test)

Action:

```
(begin (simulate-until-at-wall true SPEED BALL-WORLD))
```

Test:

```
(check-rectangles `((,RBORDER ,CY ,false)) BALL-WORLD)
```

Correct
Test (state)
Common Definitions

```
(define BALL-WORLD
(let ((w (make-world-1 SPEED))) (send w on-key "r") w))
```

1.4.3 Test (action/test)

Action:

```
(begin (simulate-until-at-wall true SPEED BALL-WORLD))
```

Test:

```
(check-rectangles `((,(- RBORDER 1) ,CY ,false)) BALL-WORLD)
```

Wrong State:

```
((385 250 #f))
```

1.4.4 Test (state)

Left bounce
Common Definitions

```
(define BALL-WORLD
(let ((w (make-world-1 SPEED))) (send w on-key "r") w))
```

1.4.5 Test (action/test)

Action:

```
(begin
(simulate-until-at-wall true SPEED BALL-WORLD)
(send BALL-WORLD on-tick)
(simulate-until-at-wall false SPEED BALL-WORLD)
(send BALL-WORLD on-tick))
```

Test:

```
(check-rectangles `((,(+ LBORDER SPEED) ,CY ,false)) BALL-WORLD)
```

Correct

1.5 Test-Group: Dragging (2 Points)

2/2

1.5.1 Test (state, 2 partial points)

Common Definitions

```
(define BALL-WORLD
(let ((w (make-world-1 SPEED))) (send w on-key "r") w))
```

1.5.2 Test (action/test)

Action:

```
(send BALL-WORLD on-mouse (+ CX 12) (- CY 5) "button-down")
```

Test:

```
(check-rectangles `((,CX ,CY ,true)) BALL-WORLD)
```

Correct

1.5.3 Test (action/test)

Action:

```
(send BALL-WORLD on-mouse 50 150 "drag")
```

Test:

```
(check-rectangles `((38 155 ,true)) BALL-WORLD)
```

Correct

1.5.4 Test (action/test, 1/4 partial points)

Action:

```
(send BALL-WORLD on-mouse 50 150 "drag")
```

Test:

```
(check-rectangles `((38 155 ,true)) BALL-WORLD)
```

Correct

1.5.5 Test (action/test, 1/4 partial points)

Action:

```
(send BALL-WORLD on-tick)
```

Test:

```
(check-rectangles `((38 155 ,true)) BALL-WORLD)
```

Correct

1.5.6 Test (action/test, 1/4 partial points)

Action:

```
(send BALL-WORLD on-key "r")
```

Test:

```
(check-rectangles `((,CX ,CY ,false) (38 155 ,true)) BALL-WORLD)
```

Correct

1.5.7 Test (action/test, 1/4 partial points)

Action:

```
(send BALL-WORLD on-mouse 300 25 "drag")
```

Test:

```
(check-rectangles `((,CX ,CY ,false) (288 30 ,true)) BALL-WORLD)
```

Correct

1.5.8 Test (action/test)

Action:

```
(send BALL-WORLD on-mouse 300 25 "button-up")
```

Test:

```
(check-rectangles `((,CX ,CY ,false) (288 30 ,false)) BALL-WORLD)
```

Correct

1.5.9 Test (action/test)

The target should not have moved
Action:

```
BALL-WORLD
```

Test:

```
(check-world CX CY false BALL-WORLD)
```

Correct

1.6 Test-Group: Special Cases (1 Points)

1/1

1.6.1 Test (state)

Common Definitions

```
(define INITIAL-WORLD (make-world-1 SPEED))
```

1.6.2 Test (action/test)

Even when we would drag the rectangle outside of the canvas, normal behavior should resume when we get back inside of the canvas
Action:

```
(begin
(send INITIAL-WORLD on-key "r")
(send INITIAL-WORLD on-mouse (+ CX 12) (- CY 5) "button-down")
(send INITIAL-WORLD on-mouse 2 2 "drag")
(send INITIAL-WORLD on-mouse (+ CX 12) (- CY 5) "drag")
(send INITIAL-WORLD on-mouse (+ CX 12) (- CY 5) "button-up"))
```

Test:

```
(check-rectangles `((,CX ,CY ,false)) INITIAL-WORLD)
```

Correct

1.6.3 Test (state)

Common Definitions

```
(define INITIAL-WORLD (make-world-1 SPEED))
```

1.6.4 Test (action/test)

The rectangle should be created at the center of the target, even when it is dragged
Action:

```
(begin
(send INITIAL-WORLD on-mouse CX CY "button-down")
(send INITIAL-WORLD on-mouse 50 50 "drag")
(send INITIAL-WORLD on-key "r"))
```

Test:

```
(check-rectangles `((50 50 ,false)) INITIAL-WORLD)
```

Correct

2 File: buddies.rkt

Tests basic functionality and buddy system for Q2
Common Definitions

```
(define CX 200)


(define CY 250)


(define LBORDER 15)


(define RBORDER 385)


(define SPEED 13)


(define check-world-selected?
(lambda (t w) (equal? t (send w get-selected?))))


(define check-world-x (lambda (t w) (equal? t (send w get-x))))


(define check-world-y (lambda (t w) (equal? t (send w get-y))))


(define check-rectangle-selected?
(lambda (t r) (equal? t (send r is-selected?))))


(define check-rectangle-color
(lambda (t r) (string=? t (send r get-color))))


(define check-rectangle-x (lambda (t r) (equal? t (send r get-x))))


(define check-rectangle-y (lambda (t r) (equal? t (send r get-y))))


(define check-rectangles
(lambda (lst w)
(let ((rmap (map map-rect-components (send w get-shapes))))
(if (or (equal? lst rmap) (equal? (reverse lst) rmap))
#t
(list rmap lst)))))
```

```
(define map-rect-components
(lambda (r)
(list
 (send r get-x)
 (send r get-y)
 (send r is-selected?)
 (send r get-color))))

(define check-rectangle
(lambda (x y sel? col)
(lambda (w)
(and (check-rectangle-x x w)
     (check-rectangle-y y w)
     (check-rectangle-selected? sel? w)
     (check-rectangle-color col w)))))

(define check-world
(lambda (x y sel? w)
(and (check-world-x x w)
     (check-world-y y w)
     (check-world-selected? sel? w))))

(define rectangle-equal?
(lambda (r1 r2)
(and (equal? (send r1 get-x) (send r2 get-x))
     (equal? (send r1 get-y) (send r2 get-y))
     (equal? (send r1 is-selected?) (send r2 is-selected?)))))

(define world-equal?
(lambda (w1 w2)
(and (equal? (send w1 get-x) (send w2 get-x))
     (equal? (send w1 get-y) (send w2 get-y))
     (equal? (send w1 get-selected?) (send w2 get-selected?))
     (andmap
      rectangle-equal?
      (send w1 get-shapes)
      (send w2 get-shapes)))))
```

2.1 Test-Group: Basic (non-)reactions (1 Points)

1/1

2.1.1 Test (state)

Common Definitions

```
(define INITIAL-WORLD (make-world-2))

(define INITIAL-WORLD-ORIG (make-world-2))
```

2.1.2 Test (action/test)

Action:

```
(send INITIAL-WORLD on-tick)
```

Test:

```
(world-equal? INITIAL-WORLD INITIAL-WORLD-ORIG)
```

Correct

2.1.3 Test (state)

Common Definitions

```
(define BALL-WORLD-ORIG
(let ((w (make-world-2))) (send w on-key "r") w))

(define BALL-WORLD (let ((w (make-world-2))) (send w on-key "r") w))
```

2.1.4 Test (action/test)

Action:

```
BALL-WORLD
```

Test:

```
(= (length (send BALL-WORLD get-shapes)) 1)
```

Correct

2.1.5 Test (action/test)

Action:

```
(send BALL-WORLD on-tick)
```

Test:

```
(world-equal? BALL-WORLD-ORIG BALL-WORLD)
```

Correct

2.2 Test-Group: Dragging the target (1 Points)

1/1

2.2.1 Test (state)

Common Definitions

```
(define drag-world (make-world-2))
```

2.2.2 Test (action/test)

Action:

```
(send drag-world on-mouse (+ CX 3) (- CY 4) "button-down")
```

Test:

```
(check-world CX CY true drag-world)
```

Correct

2.2.3 Test (action/test)

Action:

```
(send drag-world on-mouse 50 150 "drag")
```

Test:

```
(check-world 47 154 true drag-world)
```

Correct

2.2.4 Test (action/test)

Action:

```
(send drag-world on-mouse 300 25 "drag")
```

Test:

```
(check-world 297 29 true drag-world)
```

Correct

2.2.5 Test (action/test)

Action:

```
(send drag-world on-mouse 300 25 "button-up")
```

Test:

```
(check-world 297 29 false drag-world)
```

Correct

2.3 Test-Group: Test for new rectangle created overlapping (1 Points)

1/1

2.3.1 Test (state)

Common Definitions

```
(define init-world (make-world-2))
```

2.3.2 Test (action/test)

Action:

```
(send init-world on-key "r")
```

Test:

```
(check-rectangles `((,CX ,CY ,false "green")) init-world)
```

Correct

2.3.3 Test (action/test)

Action:

```
(send init-world on-mouse (+ CX 15) CY "button-down")
```

Test:

```
(check-rectangles `((,CX ,CY ,true "red")) init-world)
```

Correct

2.3.4 Test (action/test)

Action:

```
(send init-world on-mouse (+ CX 25) CY "drag")
```

Test:

```
(check-rectangles `((,(+ CX 10) ,CY ,true "red")) init-world)
```

Correct

2.3.5 Test (action/test)

Action:

```
(send init-world on-mouse (+ CX 25) CY "button-up")
```

Test:

```
(check-rectangles `((,(+ CX 10) ,CY ,false "green")) init-world)
```

Correct

2.3.6 Test (action/test)

Action:

```
(send init-world on-key "r")
```

Test:

```
(check-rectangles
`((,CX ,CY ,false "green") (,(+ CX 10) ,CY ,false "green"))
init-world)
```

Correct

2.3.7 Test (action/test)

Action:

```
(send init-world on-mouse (- CX 15) CY "button-down")
```

Test:

```
(check-rectangles
`((,CX ,CY ,true "red") (,(+ CX 10) ,CY ,false "green"))
init-world)
```

Correct

2.4 Test-Group: Buddies (5 Points)

5/5

Common Definitions

```
(define MX (+ CX 10))

(define MY (+ CY 5))
```

2.4.1 Test (state)

Common Definitions

```
(define buddy-world (make-world-2))
```

2.4.2 Test (action/test)

Action:

```
(send buddy-world on-key "r")
```

Test:

```
(check-rectangles `((,CX ,CY ,false "green")) buddy-world)
```

Correct

2.4.3 Test (action/test)

Action:

```
(send buddy-world on-mouse MX MY "button-down")
```

Test:

```
(check-rectangles `((,CX ,CY ,true "red")) buddy-world)
```

Correct

2.4.4 Test (action/test)

Action:

```
(send buddy-world on-mouse 50 50 "drag")
```

Test:

```
(check-rectangles `((40 45 ,true "red")) buddy-world)
```

Correct

2.4.5 Test (action/test)

Action:

```
(send buddy-world on-mouse 50 50 "button-up")
```

Test:

```
(check-rectangles `((40 45 ,false "green")) buddy-world)
```

Correct

2.4.6 Test (action/test)

Action:

```
(send buddy-world on-key "r")
```

Test:

```
(check-rectangles
`((,CX ,CY ,false "green") (40 45 ,false "green"))
buddy-world)
```

Correct

2.4.7 Test (action/test, 1/4 partial points)

Action:

```
(send buddy-world on-mouse MX MY "button-down")
```

Test:

```
(check-rectangles
`((,CX ,CY ,true "red") (40 45 ,false "green"))
buddy-world)
```

Correct

2.4.8 Test (action/test, 1/4 partial points)

Action:

```
(send buddy-world on-mouse 100 100 "drag")
```

Test:

```
(check-rectangles
`((90 95 ,true "red") (40 45 ,false "green"))
buddy-world)
```

Correct

2.4.9 Test (action/test, 1/4 partial points)

Action:

```
(send buddy-world on-key "r")
```

Test:

```
(check-rectangles
`((,CX ,CY ,false "green")
  (90 95 ,true "red")
  (40 45 ,false "green"))
buddy-world)
```

Correct

2.4.10 Test (action/test, 1/4 partial points)

Action:

```
(send buddy-world on-mouse 50 60 "drag")
```

Test:

```
(check-rectangles
`((,CX ,CY ,false "green") (40 55 ,true "red") (40 45 ,false "red"))
buddy-world)
```

Correct

2.4.11 Test (action/test, 1/4 partial points)

Action:

```
(send buddy-world on-mouse 100 100 "drag")
```

Test:

```
(check-rectangles
`((,CX ,CY ,false "green") (90 95 ,true "red") (40 45 ,false "red"))
buddy-world)
```

Correct

2.4.12 Test (action/test, 1/4 partial points)

Action:

```
(send buddy-world on-mouse 100 100 "button-up")
```

Test:

```
(check-rectangles
`((,CX ,CY ,false "green")
  (90 95 ,false "green")
  (40 45 ,false "green"))
buddy-world)
```

Correct

2.4.13 Test (action/test, 1/2 partial points)

Action:

```
(send buddy-world on-mouse MX MY "button-down")
```

Test:

```
(check-rectangles
`((,CX ,CY ,true "red")
  (90 95 ,false "green")
  (40 45 ,false "green"))
buddy-world)
```

Correct

2.4.14 Test (action/test, 1/2 partial points)

Action:

```
(send buddy-world on-mouse 29 100 "drag")
```

Test:

```
(check-rectangles
`((,19 ,95 ,true ,"red")
  (,90 ,95 ,false ,"green")
  (,40 ,45 ,false ,"green"))
buddy-world)
```

Correct

2.4.15 Test (action/test, 1/2 partial points)

Action:

```
(send buddy-world on-mouse 75 100 "drag")
```

Test:

```
(check-rectangles
`((,65 ,95 ,true ,"red")
  (,90 ,95 ,false ,"red")
  (,40 ,45 ,false ,"green"))
buddy-world)
```

Correct

2.4.16 Test (action/test, 1/2 partial points)

Action:

```
(send buddy-world on-mouse 75 100 "button-up")
```

Test:

```
(check-rectangles
`((,65 ,95 ,false ,"green")
  (,90 ,95 ,false ,"green")
  (,40 ,45 ,false ,"green"))
buddy-world)
```

Correct

2.4.17 Test (action/test, 1/2 partial points)

Action:

```
(send buddy-world on-mouse 100 100 "button-down")
```

Test:

```
(check-rectangles
`((,65 ,95 ,false ,"red")
  (,90 ,95 ,true ,"red")
  (,40 ,45 ,false ,"red"))
buddy-world)
```

Correct

2.4.18 Test (action/test, 1/2 partial points)

Action:

```
(send buddy-world on-mouse 100 100 "button-up")
```

Test:

```
(check-rectangles
`((,65 ,95 ,false ,"green")
  (,90 ,95 ,false ,"green")
  (,40 ,45 ,false ,"green"))
buddy-world)
```

Correct

2.4.19 Test (action/test, 1/2 partial points)

Action:

```
(send buddy-world on-mouse 50 50 "button-down")
```

Test:

```
(check-rectangles
`((,65 ,95 ,false ,"green")
  (,90 ,95 ,false ,"red")
  (,40 ,45 ,true ,"red"))
buddy-world)
```

Correct

2.4.20 Test (action/test)

Action:

```
(send buddy-world on-mouse 50 50 "button-up")
```

Test:

```
(check-rectangles
`((,65 ,95 ,false ,"green")
  (,90 ,95 ,false ,"green")
  (,40 ,45 ,false ,"green"))
buddy-world)
```

Correct

3 Results

Successes: 56
Wrong Outputs: 0
Errors: 0
Achieved Points: 15
Total Points (rounded): 15/15