CS5010 - Problem Set 02 - Test Results

pdp-91sanjay

October 2, 2013

This test suite tests your implementation of Problem Set 02

1 File: two-bouncing-cats.rkt

Tests your implementation of Bouncing cats
Common Definitions

```
(define CANVAS-WIDTH 450)

(define CANVAS-HEIGHT 400)

(define CANVAS-HALF-WIDTH (/ CANVAS-WIDTH 2))

(define CANVAS-HALF-HEIGHT (/ CANVAS-HEIGHT 2))

(define CAT-IMAGE (bitmap "cat.png"))

(define HALF-CAT-WIDTH (/ (image-width CAT-IMAGE) 2))

(define HALF-CAT-HEIGHT (/ (image-height CAT-IMAGE) 2))

(define INITIAL-Y-POS 100)

(define CAT1-X-COORD (/ CANVAS-WIDTH 3))

(define CAT2-X-COORD (* 2 CAT1-X-COORD))

(define INITIAL_WORLD (initial-world INITIAL-Y-POS))
```

```
(define RBORDER (- CANVAS-WIDTH HALF-CAT-WIDTH))

(define LBORDER HALF-CAT-WIDTH)

(define BBORDER (- CANVAS-HEIGHT HALF-CAT-HEIGHT))

(define TBORDER HALF-CAT-HEIGHT)

(define cat-1 (lambda (world) (world-cat1 world)))

(define cat-2 (lambda (world) (world-cat2 world)))

(define cat1-posn
(lambda (world)
(list (cat-x-pos (cat-1 world)) (cat-y-pos (cat-1 world)))))

(define cat2-posn
(lambda (world)
(list (cat-x-pos (cat-2 world)) (cat-y-pos (cat-2 world)))))

(define get-cat-direction
(lambda (cat)
(cond
 ((cat-north? cat) "north")
 ((cat-south? cat) "south")
 ((cat-east? cat) "east")
 ((cat-west? cat) "west"))))

(define get-cats-direction
(lambda (world)
(list
 (get-cat-direction (cat-1 world))
 (get-cat-direction (cat-2 world)))))

(define get-cat
(lambda (cat w)
(list
 (cat-x-pos (cat w))
 (cat-y-pos (cat w))
 (cat-selected? (cat w))
 (get-cat-direction (cat w)))))
```

```
(define cat-speed
(-
 (cat-y-pos (cat-1 (world-after-tick INITIAL_WORLD)))
 (cat-y-pos (cat-1 INITIAL_WORLD))))


(define get-cats
(lambda (w) `(,(get-cat cat-1 w) ,(get-cat cat-2 w))))


(define simulate-until-at-wall
(lambda (cat w)
(let ((cat-towards-east? (cat-east? (cat w)))
      (cat-towards-west? (cat-west? (cat w)))
      (cat-towards-north? (cat-north? (cat w)))
      (cat-towards-south? (cat-south? (cat w)))
      (cat-curr-x (cat-x-pos (cat w)))
      (cat-curr-y (cat-y-pos (cat w))))
(cond
 ((or (and cat-towards-east? (> cat-curr-x RBORDER))
      (and cat-towards-west? (< cat-curr-x LBORDER))
      (and cat-towards-south? (> cat-curr-y BBORDER))
      (and cat-towards-north? (< cat-curr-y TBORDER)))
(error "Moved past the edge"))
((and cat-towards-west?
      (or (= cat-curr-x RBORDER) (= cat-curr-x (- RBORDER 1))))
w)
((and cat-towards-east?
      (or (= cat-curr-x (+ LBORDER 1)) (= cat-curr-x LBORDER)))
w)
((and cat-towards-north?
      (or (= cat-curr-y BBORDER) (= cat-curr-y (- BBORDER 1))))
w)
((and cat-towards-south?
      (or (= cat-curr-y (+ TBORDER 1)) (= cat-curr-y TBORDER)))
w)
(else
 (begin
(let* ((after-tick (world-after-tick w))
       (cat-next-x (cat-x-pos (cat after-tick)))
       (cat-next-y (cat-y-pos (cat after-tick))))
(if (or (and cat-towards-east? (> cat-next-x cat-curr-x))
        (and cat-towards-west? (< cat-next-x cat-curr-x))
        (and cat-towards-south? (> cat-next-y cat-curr-y))
        (and cat-towards-north?
             (< cat-next-y cat-curr-y)))
(if (or (equal?
```

```
         (abs (- cat-next-x cat-curr-x))
         cat-speed)
       (equal?
        (abs (- cat-next-y cat-curr-y))
        cat-speed)
       (= cat-next-x RBORDER)
       (= cat-next-x (- RBORDER 1))
       (= cat-next-x LBORDER)
       (= cat-next-x (+ LBORDER 1))
       (= cat-next-x (+ LBORDER 2))
       (= cat-next-y BBORDER)
       (= cat-next-y (- BBORDER 1))
       (= cat-next-y TBORDER)
       (= cat-next-y (+ TBORDER 1)))
 (begin (simulate-until-at-wall cat after-tick))
 (error "Does not move at full speed when it should"))
(error "Does not move towards correct wall")))))))))


(define world-after-kev
(lambda (world kev) (world-after-key-event world kev)))


(define world-after-mev
(lambda (world x y mev) (world-after-mouse-event world x y mev)))


(define CX 300)


(define CY 300)


(define change-dir
(lambda (world cat mx my key)
(let* ((select-cat
        (world-after-mev
         world
         (+ (cat-x-pos (cat INITIAL_WORLD)) 10)
         (+ (cat-y-pos (cat INITIAL_WORLD)) 10)
         "button-down"))
(drag-cat (world-after-mev select-cat mx my "drag"))
(cat-after-key (world-after-kev drag-cat key))
(release-cat
 (world-after-mev cat-after-key mx my "button-up")))
release-cat)))
```

1.1 Test-Group: Basic Cats behavior (2 Points)

2/2

Common Definitions

```
(define INITWORLD-AFTER-A-TICK (world-after-tick INITIAL_WORLD))
```

1.1.1 Test (equality, 1/2 partial points)

The cats should be created in given initial y position
Input:

```
(get-cats INITIAL_WORLD)
```

Expected Output:

```
‘(,‘(,CAT1-X-COORD ,INITIAL-Y-POS ,false ,"south")
,‘(,CAT2-X-COORD ,INITIAL-Y-POS ,false ,"south"))
```

Expected Output Value:

```
((150 100 #f "south") (300 100 #f "south"))
```

Correct

1.1.2 Test (equality, 1/2 partial points)

The cats should fall at specified speed
Input:

```
(get-cats INITWORLD-AFTER-A-TICK)
```

Expected Output:

```
‘(,‘(,CAT1-X-COORD ,(+ INITIAL-Y-POS cat-speed) ,false ,"south")
,‘(,CAT2-X-COORD ,(+ INITIAL-Y-POS cat-speed) ,false ,"south"))
```

Expected Output Value:

```
((150 108 #f "south") (300 108 #f "south"))
```

Correct

1.1.3 Test (or, 1 partial points)

The cat should bounce, once it reaches the bottom
Test (equality)
The cat should bounce, once it reaches the bottom no threshold
Input:

```
(get-cat cat-1 (simulate-until-at-wall cat-1 INITWORLD-AFTER-A-
TICK))
```

Expected Output:

```
`(,CAT1-X-COORD ,BBORDER ,false ,"north")
```

Expected Output Value:

```
(150 683/2 #f "north")
```

Correct

Test (equality)

The cat should bounce, once it reaches the bottom threshold 2

Input:

```
(get-cat cat-1 (simulate-until-at-wall cat-1 INITWORLD-AFTER-A-
TICK))
```

Expected Output:

```
`(,CAT1-X-COORD ,(- BBORDER 2) ,false ,"north")
```

Expected Output Value:

```
(150 679/2 #f "north")
```

Wrong Output:

```
(150 683/2 #f "north")
```

Test (equality)

The cat should bounce, once it reaches the bottom threshold 1

Input:

```
(get-cat cat-1 (simulate-until-at-wall cat-1 INITWORLD-AFTER-A-
TICK))
```

Expected Output:

```
`(,CAT1-X-COORD ,(- BBORDER 1) ,false ,"north")
```

Expected Output Value:

```
(150 681/2 #f "north")
```

Wrong Output:

```
(150 683/2 #f "north")
```

1.2 Test-Group: Dragging Cat(s) (5 Points)

5/5

These tests will check behavior of cats after mouse events
Common Definitions

```
(define WORLD-AFTER-MEV-BD-1
(world-after-mev
 INITIAL_WORLD
 (/ CANVAS-WIDTH 2)
 (/ CANVAS-HEIGHT 2)
 "button-down"))


(define WORLD-AFTER-MEV-BD-2
(world-after-mev
 INITIAL_WORLD
 (+ (cat-x-pos (cat-1 INITIAL_WORLD)) 15)
 (+ (cat-y-pos (cat-1 INITIAL_WORLD)) 15)
 "button-down"))

(define WORLD-AFTER-MEV-DRAG
(world-after-mev WORLD-AFTER-MEV-BD-2 CX CY "drag"))

(define WORLD-AFTER-MEV-BU
(world-after-mev WORLD-AFTER-MEV-DRAG CX CY "button-up"))

(define WORLD-AFTER-MEV-BD-3
(world-after-mev
 WORLD-AFTER-MEV-BU
 (+ CAT2-X-COORD 15)
 (- (cat-y-pos (cat-2 WORLD-AFTER-MEV-BU)) 10)
 "button-down"))

(define WORLD-AFTER-MEV-DRAG-2
(world-after-mev WORLD-AFTER-MEV-BD-3 CX CY "drag"))

(define WORLD-AFTER-MEV-DRAG-RBORDER
(world-after-mev WORLD-AFTER-MEV-DRAG-2 RBORDER CY "drag"))

(define WORLD-AFTER-MEV-BU-RBORDER
(world-after-mev
 WORLD-AFTER-MEV-DRAG-RBORDER
 RBORDER
 CY
 "button-up"))
```

```
(define WORLD-AFTER-SELECTING-BOTH-CATS
(world-after-mev WORLD-AFTER-MEV-DRAG-2 CX CY "button-down"))


(define WORLD-AFTER-DRAGGING-BOTH-CATS
(world-after-mev
 WORLD-AFTER-SELECTING-BOTH-CATS
 (+ CX 50)
 (+ CY 50)
 "drag"))


(define WORLD-AFTER-DROPPING-BOTH-CATS
(world-after-mev
 WORLD-AFTER-DRAGGING-BOTH-CATS
 (+ CX 50)
 (+ CY 50)
 "button-up"))


(define WORLD-AFTER-MEV-DRAG-CAT2
(world-after-mev WORLD-AFTER-SELECTING-BOTH-CATS CX CY "drag"))
```

1.2.1 Test (equality, 1/2 partial points)

The cats should not get selected if button down occurs outside them
Input:

```
(get-cats WORLD-AFTER-MEV-BD-1)
```

Expected Output:

```
`(,`(,CAT1-X-COORD
,(cat-y-pos (cat-1 WORLD-AFTER-MEV-BD-1))
,false
,"south")
,`(,CAT2-X-COORD
   ,(cat-y-pos (cat-2 WORLD-AFTER-MEV-BD-1))
   ,false
   ,"south"))
```

Expected Output Value:

```
((150 100 #f "south") (300 100 #f "south"))
```

Correct

1.2.2 Test (equality, 1/2 partial points)

The cat should be selected on button down, only if the mouse is inside it. It's position and direction should not change on button down. Other cat should not be affected
Input:

```
(get-cats WORLD-AFTER-MEV-BD-2)
```

Expected Output:

```
`(,`(,CAT1-X-COORD
,(cat-y-pos (cat-1 WORLD-AFTER-MEV-BD-1))
,true
,"south")
,`(,CAT2-X-COORD
   ,(cat-y-pos (cat-2 WORLD-AFTER-MEV-BD-1))
   ,false
   ,"south"))
```

Expected Output Value:

```
((150 100 #t "south") (300 100 #f "south"))
```

Correct

1.2.3 Test (equality, 1/2 partial points)

The cat's position should change when it is dragged but its direction shouldn't change Other cat should not be affected
Input:

```
(get-cats WORLD-AFTER-MEV-DRAG)
```

Expected Output:

```
`(,`(,CX ,CY ,true ,"south")
,`(,CAT2-X-COORD
   ,(cat-y-pos (cat-2 WORLD-AFTER-MEV-BD-1))
   ,false
   ,"south"))
```

Expected Output Value:

```
((300 300 #t "south") (300 100 #f "south"))
```

Correct

1.2.4 Test (equality, 1/2 partial points)

The selected cat should be unselected when the mouse button is released, the cat should be placed in the new position
Input:

```
(get-cats WORLD-AFTER-MEV-BU)
```

Expected Output:

```
‘(,‘(,CX ,CY ,false ,"south")
,‘(,CAT2-X-COORD
   ,(cat-y-pos (cat-2 WORLD-AFTER-MEV-BD-1))
   ,false
   ,"south"))
```

Expected Output Value:

```
((300 300 #f "south") (300 100 #f "south"))
```

Correct

1.2.5 Test (equality, 1/2 partial points)

Dragging a cat over the other should not affect the underlying cat
Input:

```
(get-cats WORLD-AFTER-MEV-DRAG-2)
```

Expected Output:

```
‘(,‘(,CX ,CY ,false ,"south") ,‘(,CX ,CY ,true ,"south"))
```

Expected Output Value:

```
((300 300 #f "south") (300 300 #t "south"))
```

Correct

1.2.6 Test (equality, 1/2 partial points)

Both the cats should be selected on button down, when one of them is overlapping the other
Input:

```
(get-cats WORLD-AFTER-SELECTING-BOTH-CATS)
```

Expected Output:

```
‘(,‘(,CX ,CY ,true ,"south") ,‘(,CX ,CY ,true ,"south"))
```

Expected Output Value:

```
((300 300 #t "south") (300 300 #t "south"))
```

Correct

1.2.7 Test (equality, 1/2 partial points)

Both cats should be dragged together when both of them are selected
Input:

```
(get-cats WORLD-AFTER-DRAGGING-BOTH-CATS)
```

Expected Output:

```
`(,`(,(+ CX 50) ,(+ CY 50) ,true ,"south")
,`(,(+ CX 50) ,(+ CY 50) ,true ,"south"))
```

Expected Output Value:

```
((350 350 #t "south") (350 350 #t "south"))
```

Correct

1.2.8 Test (equality, 1/2 partial points)

Both cats should be unselected and their direction should not change
Input:

```
(get-cats WORLD-AFTER-DROPPING-BOTH-CATS)
```

Expected Output:

```
`(,`(,(+ CX 50) ,(+ CY 50) ,false ,"south")
,`(,(+ CX 50) ,(+ CY 50) ,false ,"south"))
```

Expected Output Value:

```
((350 350 #f "south") (350 350 #f "south"))
```

Correct

1.2.9 Test (equality, 1 partial points)

Placing the cat near the border should not affect its direction and its position
Input:

```
(get-cats WORLD-AFTER-MEV-BU-RBORDER)
```

Expected Output:

```
`(,`(,CX ,CY ,false ,"south") ,`(,RBORDER ,CY ,false ,"south"))
```

Expected Output Value:

```
((300 300 #f "south") (825/2 300 #f "south"))
```

Correct

1.3 Test-Group: Ket events test (8 Points)

8/8

These tests will check behavior of world after key events
Common Definitions

```
(define CX 200)


(define CY 200)


(define PAUSED-WORLD (world-after-key-event INITIAL_WORLD " "))


(define PAUSED-WORLD-after-tick (world-after-tick PAUSED-WORLD))


(define UNPAUSED-WORLD (world-after-kev PAUSED-WORLD-after-tick "
"))


(define UNPAUSED-WORLD-AFTER-TICK (world-after-tick UNPAUSED-WORLD))


(define WORLD-WITH-CAT1-DIR->RIGHT
(change-dir INITIAL_WORLD cat-1 CX CY "right"))


(define WORLD-WITH-CAT1-DIR->LEFT
(change-dir INITIAL_WORLD cat-1 CX CY "left"))


(define WORLD-WITH-CAT2-DIR->UP
(change-dir INITIAL_WORLD cat-2 CX CY "up"))


(define WORLD-WITH-CAT2-DIR->DOWN
(change-dir INITIAL_WORLD cat-2 CX CY "down"))


(define WORLD-WITH-CATS->BD
(world-after-mev
 WORLD-WITH-CAT2-DIR->UP
 CAT1-X-COORD
 INITIAL-Y-POS
 "button-down"))


(define WORLD-WITH-CAT1->LEFT
(world-after-kev WORLD-WITH-CATS->BD "left"))
```

```
(define WORLD-WITH-CATS-DIFF-DIR-DRAG
(world-after-mev WORLD-WITH-CAT1->LEFT CX CY "drag"))


(define WORLD-WITH-CATS-DIFF-DIR-RELEASE
(world-after-mev WORLD-WITH-CATS-DIFF-DIR-DRAG CX CY "button-up"))


(define WORLD-WITH-SELECTED-CATS
(world-after-mev
 WORLD-WITH-CATS-DIFF-DIR-RELEASE
 CX
 CY
 "button-down"))


(define WORLD-WITH-DRAGGED-CATS
(world-after-mev
 WORLD-WITH-SELECTED-CATS
 (+ CX 50)
 (+ CY 50)
 "drag"))


(define WORLD-WITH-RELEASED-CATS
(world-after-mev
 WORLD-WITH-DRAGGED-CATS
 (+ CX 50)
 (+ CY 50)
 "button-up"))


(define WORLD-WITH-CATS-IN-DIFF-DIR
(change-dir WORLD-WITH-CAT2-DIR->UP cat-1 CX CY "right"))


(define WORLD-WITH-CATS-DIFF-DIR->BD
(world-after-mev WORLD-WITH-CATS-IN-DIFF-DIR CX CY "button-down"))


(define WORLD-WITH-CATS->LEFT
(world-after-kev WORLD-WITH-CATS-DIFF-DIR->BD "left"))


(define WORLD-WITH-CATS->BU
(world-after-mev WORLD-WITH-CATS->LEFT CX CY "button-up"))
```

1.3.1 Test (equality, 1/2 partial points)

The cats should not move on tick, if the world is paused
Input:

```
(get-cats PAUSED-WORLD)
```

Expected Output:

```
`(,`(,CAT1-X-COORD ,(cat-y-pos (cat-1 INITIAL_WORLD)) ,false ,"south")
,`(,CAT2-X-COORD
   ,(cat-y-pos (cat-2 INITIAL_WORLD))
   ,false
   ,"south"))
```

Expected Output Value:

```
((150 100 #f "south") (300 100 #f "south"))
```

Correct

1.3.2 Test (equality, 1/2 partial points)

The cats should move after a tick if the world is unpaused
Input:

```
(get-cats UNPAUSED-WORLD-AFTER-TICK)
```

Expected Output:

```
`(,`(,CAT1-X-COORD
,(+ (cat-y-pos (cat-1 INITIAL_WORLD)) cat-speed)
,false
,"south")
,`(,CAT2-X-COORD
   ,(+ (cat-y-pos (cat-2 INITIAL_WORLD)) cat-speed)
   ,false
   ,"south"))
```

Expected Output Value:

```
((150 108 #f "south") (300 108 #f "south"))
```

Correct

1.3.3 Test (equality, 1/2 partial points)

The selected cat should change its direction as east, if it is selected, right key is pressed and unselected. The other cat should be unaffected by this behavior
Input:

```
(get-cats WORLD-WITH-CAT1-DIR->RIGHT)
```

Expected Output:

```
`(,`(,CX ,CY ,false ,"east")
,`(,(cat-x-pos (cat-2 INITIAL_WORLD))
   ,(cat-y-pos (cat-2 INITIAL_WORLD))
   ,false
   ,"south"))
```

Expected Output Value:

```
((200 200 #f "east") (300 100 #f "south"))
```

Correct

1.3.4 Test (equality, 1/2 partial points)

Cat 1 should move towards east after a tick at specified speed, if it is selected, right key is pressed and unselected
Input:

```
(get-cats (world-after-tick WORLD-WITH-CAT1-DIR->RIGHT))
```

Expected Output:

```
`(,`(,(+ CX cat-speed) ,CY ,false ,"east")
,`(,(cat-x-pos (cat-2 INITIAL_WORLD))
   ,(+ (cat-y-pos (cat-2 INITIAL_WORLD)) cat-speed)
   ,false
   ,"south"))
```

Expected Output Value:

```
((208 200 #f "east") (300 108 #f "south"))
```

Correct

1.3.5 Test (equality, 1/2 partial points)

The selected cat should change its direction as west, if it is selected, left key is pressed and unselected. The other cat should be unaffected by this behavior
Input:

```
(get-cats WORLD-WITH-CAT1-DIR->LEFT)
```

Expected Output:

```
`(,`(,CX ,CY ,false ,"west")
,`(,(cat-x-pos (cat-2 INITIAL_WORLD))
   ,(cat-y-pos (cat-2 INITIAL_WORLD))
   ,false
   ,"south"))
```

Expected Output Value:

```
((200 200 #f "west") (300 100 #f "south"))
```

Correct

1.3.6 Test (equality, 1/2 partial points)

Cat 1 should move towards west after a tick at specified speed, if it is selected, left key is pressed and unselected
Input:

```
(get-cats (world-after-tick WORLD-WITH-CAT1-DIR->LEFT))
```

Expected Output:

```
`(,`(,(- CX cat-speed) ,CY ,false ,"west")
,`(,(cat-x-pos (cat-2 INITIAL_WORLD))
   ,(+ (cat-y-pos (cat-2 INITIAL_WORLD)) cat-speed)
   ,false
   ,"south"))
```

Expected Output Value:

```
((192 200 #f "west") (300 108 #f "south"))
```

Correct

1.3.7 Test (equality, 1/2 partial points)

Cat 2 should move towards north after a tick at specified speed, if it is selected, up key is pressed and unselected. Other cat should be unaffected
Input:

```
(get-cats (world-after-tick WORLD-WITH-CAT2-DIR->UP))
```

Expected Output:

```
‘(,‘(,(cat-x-pos (cat-1 INITIAL_WORLD))
,(+ (cat-y-pos (cat-1 INITIAL_WORLD)) cat-speed)
,false
,"south")
,‘(,CX ,(- CY cat-speed) ,false ,"north"))
```

Expected Output Value:

```
((150 108 #f "south") (200 192 #f "north"))
```

Correct

1.3.8 Test (equality, 1/2 partial points)

Cat 2 should move towards south after a tick at specified speed, if it is selected, down key is pressed and unselected. Other cat should be unaffected
Input:

```
(get-cats (world-after-tick WORLD-WITH-CAT2-DIR->DOWN))
```

Expected Output:

```
‘(,‘(,(cat-x-pos (cat-1 INITIAL_WORLD))
,(+ (cat-y-pos (cat-1 INITIAL_WORLD)) cat-speed)
,false
,"south")
,‘(,CX ,(+ CY cat-speed) ,false ,"south"))
```

Expected Output Value:

```
((150 108 #f "south") (200 208 #f "south"))
```

Correct

1.3.9 Test (equality, 1 partial points)

The cats directions are changed separately, then selected together, dragged and released. They should continue moving in their directions unchanged
Input:

```
(get-cats (world-after-tick WORLD-WITH-RELEASED-CATS))
```

Expected Output:

```
‘(,‘(,(- (+ CX 50) cat-speed) ,(+ CY 50) ,false ,"west")
,‘(,(+ CX 50) ,(- (+ CY 50) cat-speed) ,false ,"north"))
```

Expected Output Value:

```
((242 250 #f "west") (250 242 #f "north"))
```

Correct

1.3.10 Test (equality, 1/2 partial points)

The cats were selected together, then left key is pressed, and released. They both should be travelling together towards west
Input:

```
(get-cats (world-after-tick WORLD-WITH-CATS->BU))
```

Expected Output:

```
`(,`(,(- CX cat-speed) ,CY ,false ,"west")
,`(,(- CX cat-speed) ,CY ,false ,"west"))
```

Expected Output Value:

```
((192 200 #f "west") (192 200 #f "west"))
```

Correct

1.3.11 Test (or, 1/2 partial points)

The cat should hit the west wall, bounces back and start travelling towards east
Test (equality)
The cat should hit the west wall, bounces back and start travelling towards east with no threshold
Input:

```
(get-cat
cat-1
(simulate-until-at-wall cat-1 WORLD-WITH-CAT1-DIR->LEFT))
```

Expected Output:

```
`(,LBORDER ,CY ,false ,"east")
```

Expected Output Value:

```
(75/2 200 #f "east")
```

Correct
Test (equality)
The cat should hit the west wall, bounces back and start travelling towards east with threshold 1
Input:

```
(get-cat
cat-1
(simulate-until-at-wall cat-1 WORLD-WITH-CAT1-DIR->LEFT))
```

Expected Output:

```
`(,(+ LBORDER 1) ,CY ,false ,"east")
```

Expected Output Value:

```
(77/2 200 #f "east")
```

Wrong Output:

```
(75/2 200 #f "east")
```

Test (equality)
The cat should hit the west wall, bounces back and start travelling towards east with threshold 2
Input:

```
(get-cat
cat-1
(simulate-until-at-wall cat-1 WORLD-WITH-CAT1-DIR->LEFT))
```

Expected Output:

```
`(,(+ LBORDER 2) ,CY ,false ,"east")
```

Expected Output Value:

```
(79/2 200 #f "east")
```

Wrong Output:

```
(75/2 200 #f "east")
```

1.3.12 Test (or, 1/2 partial points)

The cat should hit the east wall, bounces back and start travelling towards west
Test (equality)
The cat should hit the east wall, bounces back and start travelling towards west no threshold
Input:

```
(get-cat
cat-1
(simulate-until-at-wall cat-1 WORLD-WITH-CAT1-DIR->RIGHT))
```

Expected Output:

```
`(,RBORDER ,CY ,false ,"west")
```

Expected Output Value:

```
(825/2 200 #f "west")
```

Correct
Test (equality)
The cat should hit the east wall, bounces back and start travelling towards west with threshold 1
Input:

```
(get-cat
cat-1
(simulate-until-at-wall cat-1 WORLD-WITH-CAT1-DIR->RIGHT))
```

Expected Output:

```
`(,(- RBORDER 1) ,CY ,false ,"west")
```

Expected Output Value:

```
(823/2 200 #f "west")
```

Wrong Output:

```
(825/2 200 #f "west")
```

Test (equality)
The cat should hit the east wall, bounces back and start travelling towards west with threshold 2
Input:

```
(get-cat
cat-1
(simulate-until-at-wall cat-1 WORLD-WITH-CAT1-DIR->RIGHT))
```

Expected Output:

```
`(,(- RBORDER 2) ,CY ,false ,"west")
```

Expected Output Value:

```
(821/2 200 #f "west")
```

Wrong Output:

```
(825/2 200 #f "west")
```

1.3.13 Test (or, 1/2 partial points)

The cat should hit the north wall, bounces back and start travelling towards south
Test (equality)
The cat should hit the north wall, bounces back and start travelling towards south no threshold
Input:

```
(get-cat cat-2 (simulate-until-at-wall cat-2 WORLD-WITH-CAT2-DIR-
>UP))
```

Expected Output:

```
`(,CX ,TBORDER ,false ,"south")
```

Expected Output Value:

```
(200 117/2 #f "south")
```

Correct

Test (equality)

The cat should hit the north wall, bounces back and start travelling towards south with threshold 1

Input:

```
(get-cat cat-2 (simulate-until-at-wall cat-2 WORLD-WITH-CAT2-DIR-
>UP))
```

Expected Output:

```
`(,CX ,(+ TBORDER 1) ,false ,"south")
```

Expected Output Value:

```
(200 119/2 #f "south")
```

Wrong Output:

```
(200 117/2 #f "south")
```

Test (equality)

The cat should hit the north wall, bounces back and start travelling towards south with threshold 2

Input:

```
(get-cat cat-2 (simulate-until-at-wall cat-2 WORLD-WITH-CAT2-DIR-
>UP))
```

Expected Output:

```
`(,CX ,(+ TBORDER 2) ,false ,"south")
```

Expected Output Value:

```
(200 121/2 #f "south")
```

Wrong Output:

```
(200 117/2 #f "south")
```

1.3.14 Test (or, 1/2 partial points)

The cat should hit the south wall, bounces back and start travelling towards north no threshold

Test (equality)

The cat should hit the south wall, bounces back and start travelling towards north no threshold

Input:

```
(get-cat
cat-2
(simulate-until-at-wall cat-2 WORLD-WITH-CAT2-DIR->DOWN))
```

Expected Output:

```
`(,CX ,BBORDER ,false ,"north")
```

Expected Output Value:

```
(200 683/2 #f "north")
```

Correct

Test (equality)

The cat should hit the south wall, bounces back and start travelling towards north with threshold 1

Input:

```
(get-cat
cat-2
(simulate-until-at-wall cat-2 WORLD-WITH-CAT2-DIR->DOWN))
```

Expected Output:

```
`(,CX ,(- BBORDER 1) ,false ,"north")
```

Expected Output Value:

```
(200 681/2 #f "north")
```

Wrong Output:

```
(200 683/2 #f "north")
```

Test (equality)

The cat should hit the south wall, bounces back and start travelling towards north with threshold 2

Input:

```
(get-cat
cat-2
(simulate-until-at-wall cat-2 WORLD-WITH-CAT2-DIR->DOWN))
```

Expected Output:

```
`(,CX ,(- BBORDER 2) ,false ,"north")
```

Expected Output Value:

```
(200 679/2 #f "north")
```

Wrong Output:

```
(200 683/2 #f "north")
```

2 Results

Successes: 26
Wrong Outputs: 0
Errors: 0
Achieved Points: 15
Total Points (rounded): 15/15