

```

1  ;; DO NOT PUT ANYTHING PERSONALLY IDENTIFYING BEYOND YOUR CWL IN
1  THIS FILE.
2  ;; YOUR CWLS WILL BE SUFFICIENT TO IDENTIFY YOU AND, IF YOU HAVE
2  ONE, YOUR
3  ;; PARTNER.
4  (require 2htdp/universe)
5  (require 2htdp/image)
6  (require spd/tags)
7
8  (@assignment psets/pset-05);Do not edit or remove this tag
9
10 ;; If you are:
11 ;;   - A 110 or 107 student replace the first set of '???'s with
11 your cwl.
12 ;;   For problem sets, If you have a partner, please replace the
12 second
13 ;;   set of '???'s with their cwl. Remember this, it is what
13 you will
14 ;;   do with these @cwl annotations for the whole course.
15 ;;   - A UBC Extended Learning student, replace the first set of
15 ??? with
16 ;;   your email address as confirmed in the email you received
16 from
17 ;;   extended learning. The handin password is also in that
17 email.
18 ;;   Remember this, it is what you will do with these @cwl
18 annotations
19 ;;   for the whole course.
20 ;;
21 (@cwl dana28cl ???)
22
23 ;; Bounce any number of balls around the screen.
24
25
26 ;;
27 ;; In this problem set you are given our official solution to
27 problem
28 ;; set 4 (with a few additional things added) as a starting point.
29 ;; We have given you some more constants, a helper function called
30 ;; touch-paddle? which you may use, and a new data definition
30 called Game.
31 ;; You need to revise the program so that:
32 ;;   - the game includes a paddle that moves back and forth across
32 the
33 ;;   bottom of the screen
34 ;;   - the paddle is controlled by the left and right arrow keys
35 ;;   - when a ball hits the paddle it disappears
36 ;;   - as before the mouse can be used to add balls to the game
37 ;;
38 ;; As stated above, we have given you a new data definition called
38 Game.
39 ;; You MUST revise the program so that it uses Game as the world
39 state.
40 ;; You MUST NOT change the Game data definition in anyway (though
40 you are
41 ;; allowed to add more Game constants).
42 ;;
43 ;; We suggest you work in three distinct phases, making sure your
43 program

```

```

44 ;; works correctly at the end of each phase before going on to the
44 next.
45 ;; - change the program's world state to Game
46 ;; - provide left/right arrow key control over the paddle
47 ;; - make it so that when a ball hits the paddle it disappears
48 ;;
49 ;; In each of these phases you should follow the design recipes!
49 Re-work
50 ;; the domain analysis for changing and constant information,
50 update the
51 ;; data definitions, revise the main function, and so on. Make
51 sure that
52 ;; your tags are correct and that all your tests work correctly
52 before you
53 ;; proceed to the next phase.
54 ;;
55 ;; NOTE: Your on-tick function MUST be designed as a composition
55 of two other
56 ;; functions called game-with-next-balls and
56 game-with-caught-balls.
57 ;;
58 ;; Note that we are giving you significant help in the starter
58 file.
59 ;; You absolutely MUST USE OUR STARTER AS THE BASIS FOR YOUR WORK.
60 ;;
61 ;; We recommend that you begin by printing this file and planning
61 out what
62 ;; needs to change, what needs to be added, and what will be
62 unchanged.
63 ;;
64 (@problem 1)
65 (@htdw ListOfBall)
66
67 ;; Constants:
68 (define WIDTH 605)
69 (define HEIGHT 535)
70
71 (define PADDLE-WIDTH 60)
72 (define PADDLE-THICKNESS 10)
73 (define PADDLE (rectangle PADDLE-WIDTH PADDLE-THICKNESS "solid"
73 "white"))
74 (define PADDLE-CTR-Y (- HEIGHT 40))
75 (define PADDLE-MOVE-PER-KEY 10)
76
77 (define BALL-RADIUS 10)
78
79 (define TOP BALL-RADIUS)
80 (define BOT (- HEIGHT 1 BALL-RADIUS))
81 (define LEF BALL-RADIUS)
82 (define RIG (- WIDTH 1 BALL-RADIUS))
83
84 (define BALL (circle BALL-RADIUS "solid" "white"))
85
86 (define MTS (rectangle WIDTH HEIGHT "solid" "green"))
87
88
89 ;;
89 =====
89 =====

```

```

90 ;;
90 =====
90 =====
91 ;; Data definitions:
92
93 (@htdd Ball)
94 (define-struct ball (x y dx dy))
95 ;; Ball is (make-ball Number Number Number Number)
96 ;; interp. (make-ball x y dx dy) is ball
97 ;;   - position x, y    in screen coordinates
98 ;;   - velocity dx, dy  in pixels/tick
99 ;; CONSTRAINT: x is in [LEF, RIG]; y is in [TOP, BOT]
100 (define B1 (make-ball (/ WIDTH 2) (/ HEIGHT 2) 4 -3))
101
102 (@dd-template-rules compound)
103
104 (define (fn-for-ball b)
105   (... (ball-x b)
106        (ball-y b)
107        (ball-dx b)
108        (ball-dy b)))
109
110 (@htdd ListOfBall)
111 ;; ListOfBall is one of:
112 ;;   - empty
113 ;;   - (cons Ball ListOfBall)
114 ;; interp. a list of balls
115 (define LOB1 empty)
116 (define LOB2 (cons B1 empty))
117
118 (@dd-template-rules one-of
119                               atomic-distinct
120                               compound
121                               ref
122                               self-ref)
123
124 (define (fn-for-lob lob)
125   (cond [(empty? lob) (...)]
126         [else
127          (... (fn-for-ball (first lob))
128              (fn-for-lob (rest lob)))]))
129
130
131 (@htdd Game)
132 (define-struct game (balls paddle))
133 ;; Game is (make-game ListOfBall Number)
134 ;; interp. the current state of a game, with all the balls in play,
135 ;;         as well as the x-position of the paddle in screen
136 coordinates
137 (define G0 (make-game empty (/ WIDTH 2)))
138 (define G1 (make-game (cons B1 empty) (/ WIDTH 2)))
139
140 (@dd-template-rules compound ref)
141
142 (define (fn-for-game g)
143   (... (fn-for-lob (game-balls g))
144        (game-paddle g)))
145

```

```

146
147 ;;
147 =====
147 =====
148 ;;
148 =====
148 =====
149 ;; Functions:
150
151 (@htdf main)
152 (@signature ListOfBall -> ListOfBall)
153 ;; start the game, call with (main LOB1)
154 ;; <no tests for main functions>
155
156 (@template-origin htdw-main)
157
158 (define (main lob)
159   (big-bang lob
160     (on-draw   render-balls)      ;ListOfBall -> Image
161     (on-tick   next-balls)        ;ListOfBall -> ListOfBall
162     (on-key    handle-key)        ;ListOfBall KeyEvent -> ListOfBall
163     (on-mouse  handle-mouse))) ;ListOfBall Integer Integer
164                               ;    -> ListOfBall
165
166 (@htdf render-balls)
167 (@signature ListOfBall -> Image)
168 ;; render all balls onto MTS
169 (check-expect (render-balls empty) MTS)
170 (check-expect (render-balls (cons (make-ball 10 20 3 4)
171                                   (cons (make-ball 30 40 1 2)
172                                           empty))))
173               (place-ball (make-ball 10 20 3 4)
174                           (place-ball (make-ball 30 40 1 2)
175                                         MTS)))
176
177 ;(define (render-balls lob) MTS) ;stub
178
179 (@template-origin ListOfBall)
180
181 (@template
182   (define (render-balls lob)
183     (cond [(empty? lob) (...)]
184           [else
185            (... (fn-for-ball (first lob))
186                  (render-balls (rest lob)))])))
187
188 (define (render-balls lob)
189   (cond [(empty? lob) MTS]
190         [else
191          (place-ball (first lob)
192                      (render-balls (rest lob)))]))
193
194
195 (@htdf place-ball)
196 (@signature Ball Image -> Image)
197 ;; place BALL on image at appropriate x, y coordinate
198 (check-expect (place-ball (make-ball 20 30 3 3) MTS)
199               (place-image BALL 20 30 MTS))

```

```

200 (check-expect (place-ball (make-ball 10 20 -2 -1) empty-image)
201               (place-image BALL 10 20 empty-image))
202 #;
203 (define (place-ball b img) img)
204
205 (@template-origin Ball)
206
207 (@template
208   (define (place-ball b img)
209     (... (ball-x b)
210          (ball-y b)
211          (ball-dx b)
212          (ball-dy b)
213          img)))
214
215 (define (place-ball b img)
216   (place-image BALL (ball-x b) (ball-y b) img))
217
218
219 (@htdf next-balls)
220 (@signature ListOfBall -> ListOfBall)
221 ;; produce list of balls at their next x, y coordinates
222 (check-expect (next-balls empty) empty)
223 (check-expect (next-balls (cons (make-ball (+ LEF 1) TOP 3 -4)
224                                 (cons (make-ball 200 100 3 4)
225                                       empty))))
226               (cons (next-ball (make-ball (+ LEF 1) TOP 3 -4))
227                     (cons (next-ball (make-ball 200 100 3 4))
228                           empty)))
229
230 #;
231 (define (next-balls lob) empty)
232
233 (@template-origin ListOfBall)
234
235 (@template
236   (define (next-balls lob)
237     (cond [(empty? lob) (...)]
238           [else
239            (... (fn-for-ball (first lob))
240                  (next-balls (rest lob)))])))
241
242 (define (next-balls lob)
243   (cond [(empty? lob) empty]
244         [else
245          (cons (next-ball (first lob))
246                (next-balls (rest lob)))]))
247
248
249 (@htdf next-ball)
250 (@signature Ball -> Ball)
251 ;; produce ball at next x,y; checks bounces off
252 top/right/bottom/left wall
252 (check-expect (next-ball (make-ball (+ LEF 1) TOP 3 -4))
253               (bounce-top (make-ball (+ LEF 1) TOP 3 -4)))
254 (check-expect (next-ball (make-ball (+ LEF 1) BOT 3 4))
255               (bounce-bottom (make-ball (+ LEF 1) BOT 3 4)))
256 (check-expect (next-ball (make-ball LEF (+ TOP 1) -3 4))
257               (bounce-left (make-ball LEF (+ TOP 1) -3 4)))

```

```

258 (check-expect (next-ball (make-ball RIG (+ TOP 1) 3 4))
259 (bounce-right (make-ball RIG (+ TOP 1) 3 4)))
260 (check-expect (next-ball (make-ball (/ WIDTH 2) (/ HEIGHT 2) 3
260 4))
261 (glide (make-ball (/ WIDTH 2) (/ HEIGHT 2) 3
261 4)))
262 #;
263 (define (next-ball b) b)
264
265 (@template-origin Number) ; because b is treated as atomic
266
267 (@template
268 (define (next-ball b)
269 (... b)))
270
271 (define (next-ball b)
272 (cond [(touch-top? b) (bounce-top b)]
273 [(touch-bottom? b) (bounce-bottom b)]
274 [(touch-right? b) (bounce-right b)]
275 [(touch-left? b) (bounce-left b)]
276 [else
277 (glide b)]))
278
279
280 (@htdf handle-mouse)
281 (@signature ListOfBall Integer Integer MouseEvent -> ListOfBall)
282 ;; adds new ball at x, y to lob
283 ;; NOTE: uses random, so testing has to use check-random
284 (check-random (handle-mouse empty 100 200 "button-down")
285 (cons (make-ball 100 200 (- 5 (random 11)) (- 5
285 (random 11)))
286 empty))
287 (check-random (handle-mouse (cons (make-ball 10 20 -3 3) empty)
288 300 100 "button-down")
289 (cons (make-ball 300 100 (- 5 (random 11)) (- 5
289 (random 11)))
290 (cons (make-ball 10 20 -3 3) empty)))
291 (check-random (handle-mouse empty 100 200 "button-up") empty)
292 (check-random (handle-mouse (cons (make-ball 10 20 -3 3) empty)
293 100 200 "button-up")
294 (cons (make-ball 10 20 -3 3) empty))
295 #;
296 (define (handle-mouse lob x y me) empty)
297
298 (@template-origin MouseEvent)
299
300 (@template
301 (define (handle-mouse lob x y me)
302 (cond [(mouse=? me "button-down") (... lob x y)]
303 [else
304 (... lob x y)]))
305
306 (define (handle-mouse lob x y me)
307 (cond [(mouse=? me "button-down")
308 (cons (make-ball x y (- 5 (random 11)) (- 5 (random 11)))
308 lob)]
309 [else lob]))
310
311

```

```

312 (@htdf handle-key)
313 (@signature ListOfBall KeyEvent -> ListOfBall)
314 ;; clear all balls if space key pressed; else do nothing
315 (check-expect (handle-key (cons (make-ball (/ WIDTH 2) (/ HEIGHT
316 2) 2 4)
317                                     (cons (make-ball (+ TOP 2) (+ LEF
318 5) 3 2)
319                                             empty)))
320                                     " ")
321 (check-expect (handle-key (cons (make-ball (/ WIDTH 3) (/ HEIGHT
322 4) 1 -3)
323                                     (cons (make-ball (+ TOP 5) (+ LEF
324 2) 3 -2)
325                                             empty)))
326                                     "c")
327 (cons (make-ball (/ WIDTH 3) (/ HEIGHT 4) 1 -3)
328       (cons (make-ball (+ TOP 5) (+ LEF 2) 3 -2)
329             empty)))
330 #;
331 (define (handle-key lob ke) empty)
332 (@template-origin KeyEvent)
333 (@template
334   (define (handle-key lob ke)
335     (cond [(key=? ke " ") (... lob)]
336           [else
337            (... lob)])))
338 (define (handle-key lob ke)
339   (cond [(key=? ke " ") empty]
340         [else lob]))
341
342
343 (@htdf touch-paddle?)
344 (@signature Ball Number -> Boolean)
345 ;; produce true if ball's center is inside the paddle
346 ;; NOTE: There are many better and more complex ways to design
346 this function.
347 ;; This design is fairly primitive (just checks that the
347 center of the
348 ;; ball is in the paddle), but people playing the game
348 shouldn't see
349 ;; much difference if the balls are moving quickly.
350 (check-expect (touch-paddle? (make-ball (- 100 (/ PADDLE-WIDTH 2)
351 1)
352                                     PADDLE-CTR-Y
353                                     1 2)
354               100)
355               false)
356 (check-expect (touch-paddle? (make-ball (- 100 (/ PADDLE-WIDTH 2))
357                                     PADDLE-CTR-Y
358                                     1 2)
359               100)
360               true)
361 (check-expect (touch-paddle? (make-ball (+ 100 (/ PADDLE-WIDTH 2))
362                                     PADDLE-CTR-Y
363                                     1 2)

```

```

363                                     100)
364                                     true)
365 (check-expect (touch-paddle? (make-ball (+ 100 (/ PADDLE-WIDTH 2)
366 1)
367                                     PADDLE-CTR-Y
368                                     1 2)
369                                     100)
370 (check-expect (touch-paddle?
371 (make-ball (+ 100 (/ PADDLE-WIDTH 2))
372 (- PADDLE-CTR-Y (/ PADDLE-THICKNESS 2) 1)
373 1 2)
374 100)
375 false)
376 (check-expect (touch-paddle?
377 (make-ball (+ 100 (/ PADDLE-WIDTH 2))
378 (- PADDLE-CTR-Y (/ PADDLE-THICKNESS 2))
379 1 2)
380 100)
381 true)
382 (check-expect (touch-paddle?
383 (make-ball (+ 100 (/ PADDLE-WIDTH 2))
384 (+ PADDLE-CTR-Y (/ PADDLE-THICKNESS 2))
385 1 2)
386 100)
387 true)
388 (check-expect (touch-paddle?
389 (make-ball (+ 100 (/ PADDLE-WIDTH 2))
390 (+ PADDLE-CTR-Y (/ PADDLE-THICKNESS 2) 1)
391 1 2)
392 100)
393 false)
394 (check-expect (touch-paddle? (make-ball (+ 30 (/ PADDLE-WIDTH 2))
395 PADDLE-CTR-Y
396 1 2)
397 30)
398 true)
399
400 (@template-origin Ball)
401
402 (@template
403 (define (touch-paddle? b p)
404 (... (ball-x b)
405 (ball-y b)
406 (ball-dx b)
407 (ball-dy b)
408 p)))
409
410 (define (touch-paddle? b p)
411 (and (<= (- p (/ PADDLE-WIDTH 2))
412 (ball-x b)
413 (+ p (/ PADDLE-WIDTH 2)))
414 (<= (- PADDLE-CTR-Y (/ PADDLE-THICKNESS 2))
415 (ball-y b)
416 (+ PADDLE-CTR-Y (/ PADDLE-THICKNESS 2)))))
417
418
419 (@htdf touch-top?)
420 (@signature Ball -> Boolean)

```



```

421 ;; true if ball is going up and edge will hit top edge of box
422 (check-expect (touch-top? (make-ball LEF (+ TOP 5) 3 -4))
422 false)
423 (check-expect (touch-top? (make-ball LEF (+ TOP 4) 3 -4)) true)
424 (check-expect (touch-top? (make-ball LEF (+ TOP 1) 3 -2)) true)
425 (check-expect (touch-top? (make-ball LEF (+ TOP 0) 3 2))
425 false)
426 #;
427 (define (touch-top? b) false)
428
429 (@template-origin Ball)
430
431 (@template
432 (define (touch-top? b)
433   (... (ball-x b)
434         (ball-y b)
435         (ball-dx b)
436         (ball-dy b))))
437
438 (define (touch-top? b)
439   (<= (+ (ball-y b) (ball-dy b)) TOP))
440
441
442 (@htdf touch-bottom?)
443 (@signature Ball -> Boolean)
444 ;; true if ball is going down and edge will hit bottom edge of box
445 (check-expect (touch-bottom? (make-ball LEF (- BOT 3) 3 2)) false)
446 (check-expect (touch-bottom? (make-ball LEF (- BOT 2) 3 2)) true)
447 (check-expect (touch-bottom? (make-ball LEF (- BOT 0) 3 2)) true)
448 (check-expect (touch-bottom? (make-ball LEF (- BOT 0) 3 -2)) false)
449 #;
450 (define (touch-bottom? b) false)
451
452 (@template-origin Ball)
453
454 (@template
455 (define (touch-bottom? b)
456   (... (ball-x b)
457         (ball-y b)
458         (ball-dx b)
459         (ball-dy b))))
460
461 (define (touch-bottom? b)
462   (>= (+ (ball-y b) (ball-dy b)) BOT))
463
464
465 (@htdf touch-left?)
466 (@signature Ball -> Boolean)
467 ;; true if ball is going left and edge will hit left edge of box
468 (check-expect (touch-left? (make-ball (+ LEF 6) TOP -5 2)) false)
469 (check-expect (touch-left? (make-ball (+ LEF 5) TOP -5 2)) true)
470 (check-expect (touch-left? (make-ball (+ LEF 0) TOP -5 2)) true)
471 (check-expect (touch-left? (make-ball (+ LEF 0) TOP 3 2)) false)
472 #;
473 (define (touch-left? b) false)
474
475 (@template-origin Ball)
476
477 (@template

```

```

478 (define (touch-left? b)
479   (... (ball-x b)
480         (ball-y b)
481         (ball-dx b)
482         (ball-dy b))))
483
484 (define (touch-left? b)
485   (<= (+ (ball-x b) (ball-dx b)) LEF))
486
487
488 (@htdf touch-right?)
489 (@signature Ball -> Boolean)
490 ;; true if ball is going right and edge will hit right edge of box
491 (check-expect (touch-right? (make-ball (- RIG 6) TOP 5 2)) false)
492 (check-expect (touch-right? (make-ball (- RIG 5) TOP 5 2)) true)
493 (check-expect (touch-right? (make-ball (- RIG 0) TOP 5 2)) true)
494 (check-expect (touch-right? (make-ball (- RIG 0) TOP -3 2)) false)
495 #;
496 (define (touch-right? b) false)
497
498 (@template-origin Ball)
499
500 (@template
501 (define (touch-right? b)
502   (... (ball-x b)
503         (ball-y b)
504         (ball-dx b)
505         (ball-dy b))))
506
507 (define (touch-right? b)
508   (>= (+ (ball-x b) (ball-dx b)) RIG))
509
510
511 (@htdf bounce-top)
512 (@signature Ball -> Ball)
513 ;; produce a ball with top edge 1 pixel off top of box, moving down
514 ;; CONSTRAINT: assume ball is close to top edge and moving up
515 (check-expect (bounce-top (make-ball (+ LEF 1) (+ TOP 3) 2 -4))
516               (make-ball (+ LEF 1) (+ TOP 1) 2 4))
517 (check-expect (bounce-top (make-ball (+ LEF 2) (+ TOP 6) 3 -7))
518               (make-ball (+ LEF 2) (+ TOP 1) 3 7))
519 #;
520 (define (bounce-top b) b)
521
522 (@template-origin Ball)
523
524 (@template
525 (define (bounce-top b)
526   (... (ball-x b)
527         (ball-y b)
528         (ball-dx b)
529         (ball-dy b))))
530
531 (define (bounce-top b)
532   (make-ball (ball-x b) (+ TOP 1) (ball-dx b) (- (ball-dy b))))
533
534
535 (@htdf bounce-bottom)
536 (@signature Ball -> Ball)

```

```

537 ;; produce a ball with bottom edge 1 pixel off bottom of box,
537 moving up
538 ;; CONSTRAINT: assume ball is close to bottom edge and moving down
539 (check-expect (bounce-bottom (make-ball (+ LEF 1) (- BOT 3) 2 4))
540               (make-ball (+ LEF 1) (- BOT 1) 2 -4))
541 (check-expect (bounce-bottom (make-ball (+ LEF 2) (- BOT 6) 3 7))
542               (make-ball (+ LEF 2) (- BOT 1) 3 -7))
543 #;
544 (define (bounce-bottom b) b)
545
546 (@template-origin Ball)
547
548 (@template
549   (define (bounce-bottom b)
550     (... (ball-x b)
551          (ball-y b)
552          (ball-dx b)
553          (ball-dy b))))
554
555 (define (bounce-bottom b)
556   (make-ball (ball-x b) (- BOT 1) (ball-dx b) (- (ball-dy b))))
557
558 (@htdf bounce-left)
559 (@signature Ball -> Ball)
560 ;; produce a ball with left edge 1 pixel off left of box, moving
560 right
561 ;; CONSTRAINT: assume ball is close to left edge and moving left
562 (check-expect (bounce-left (make-ball (+ LEF 3) (+ TOP 2) -4 4))
563               (make-ball (+ LEF 1) (+ TOP 2) 4 4))
564 (check-expect (bounce-left (make-ball (+ LEF 5) (+ TOP 2) -8 4))
565               (make-ball (+ LEF 1) (+ TOP 2) 8 4))
566 #;
567 (define (bounce-left b) b)
568
569 (@template-origin Ball)
570
571 (@template
572   (define (bounce-left b)
573     (... (ball-x b)
574          (ball-y b)
575          (ball-dx b)
576          (ball-dy b))))
577
578 (define (bounce-left b)
579   (make-ball (+ LEF 1) (ball-y b) (- (ball-dx b)) (ball-dy b)))
580
581
582 (@htdf bounce-right)
583 (@signature Ball -> Ball)
584 ;; produce a ball with right edge 1 pixel off right of box, moving
584 left
585 ;; CONSTRAINT: assume ball is close to right edge and moving right
586 (check-expect (bounce-right (make-ball (- RIG 3) (+ TOP 1) 4 4))
587               (make-ball (- RIG 1) (+ TOP 1) -4 4))
588 (check-expect (bounce-right (make-ball (- RIG 5) (+ TOP 1) 8 4))
589               (make-ball (- RIG 1) (+ TOP 1) -8 4))
590 #;
591 (define (bounce-right b) b)
592

```

```

593 | (@template-origin Ball)
594 |
595 | (@template
596 |   (define (bounce-right b)
597 |     (... (ball-x b)
598 |           (ball-y b)
599 |           (ball-dx b)
600 |           (ball-dy b))))
601 |
602 | (define (bounce-right b)
603 |   (make-ball (- RIG 1) (ball-y b) (- (ball-dx b)) (ball-dy b)))
604 |
605 |
606 | (@htdf glide)
607 | (@signature Ball -> Ball)
608 | ;; move ball by dx dy
609 | ;; CONSTRAINT: ball is not touching or about to touch any edge of
609 | the box
610 | (check-expect (glide (make-ball 100 200 2 3)) (make-ball 102 203 2
610 | 3))
611 | (check-expect (glide (make-ball 50 220 -3 -2)) (make-ball 47 218
611 | -3 -2))
612 | #;
613 | (define (glide b) b)
614 |
615 | (@template-origin Ball)
616 |
617 | (@template
618 |   (define (glide b)
619 |     (... (ball-x b)
620 |           (ball-y b)
621 |           (ball-dx b)
622 |           (ball-dy b))))
623 |
624 | (define (glide b)
625 |   (make-ball (+ (ball-x b) (ball-dx b))
626 |             (+ (ball-y b) (ball-dy b))
627 |             (ball-dx b)
628 |             (ball-dy b)))

```