

# 1. Increase the size of the game area, and make the winning score higher.

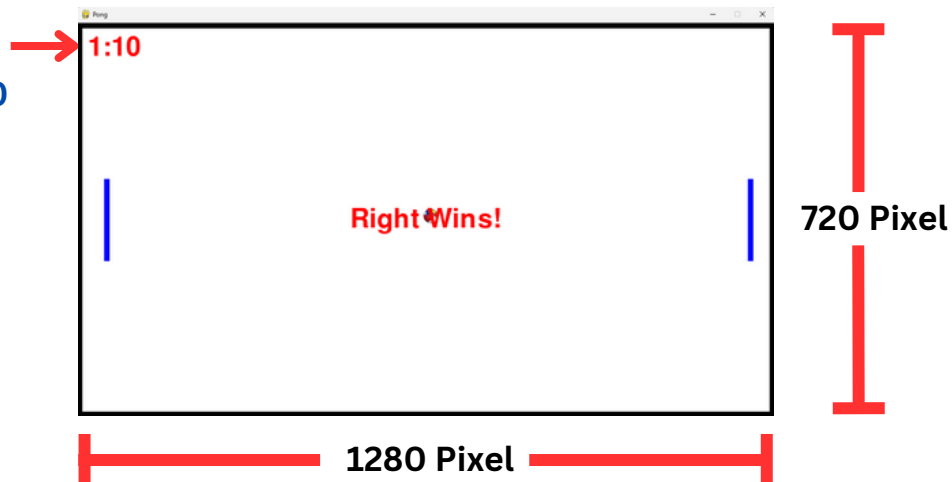
```
# ----- main -----
pygame.init()
screen = pygame.display.set_mode([640,480])
screen.fill(WHITE)
pygame.display.set_caption("Pong")
scrWidth, scrHeight = screen.get_size()
```

WINNING\_SCORE = 5

```
# ----- main -----
pygame.init()
screen = pygame.display.set_mode([1280,720])
screen.fill(WHITE)
pygame.display.set_caption("Pong")
scrWidth, scrHeight = screen.get_size()
```

WINNING\_SCORE = 10

BEFORE 1:5  
AFTER 1:10



- EXPLAIN:**
- Change a screen size from [640, 480] to [1280, 720]
  - Change winning score from 5 to 10

## 2. Make the ball gradually move faster as the game playing time increases.

1.
 

```
def __init__(self, fnm):
    super().__init__()
    self.image = pygame.image.load(fnm).convert_alpha()
    self.rect = self.image.get_rect()
    self.rect.center = [scrWidth/2, scrHeight/2]
    # start position of the ball in center of window
    self.xStep, self.yStep = self.randomSteps()
    # step size and direction along each axis
    self.time_counter = 0 # Initial playtime
```
2.
 

```
self.time_counter += 1
if self.time_counter % 250 == 0: self.accStep()

def accStep(self):
    self.xStep += STEP if self.xStep > 0 else -STEP
    self.yStep += STEP if self.yStep > 0 else -STEP
```
3.
 

```
# game vars
leftStep = 0; rightStep = 0 # move
scoreLeft = 0; scoreRight = 0
winMsg = ""
gameOver = False
count = 0
```
4.
 

```
# update game
if not gameOver:
    leftPaddle.move(leftStep)
    rightPaddle.move(rightStep)
    ball.update()

    if scoreLeft >= WINNING_SCORE:
        winMsg = "Left Wins!"
        gameOver = True
    elif scoreRight >= WINNING_SCORE:
        winMsg = "Right Wins!"
        gameOver = True

    count += 1 # count time
```
5.
 

```
font = pygame.font.Font(None, 72)
font_time = pygame.font.Font(None, 36)

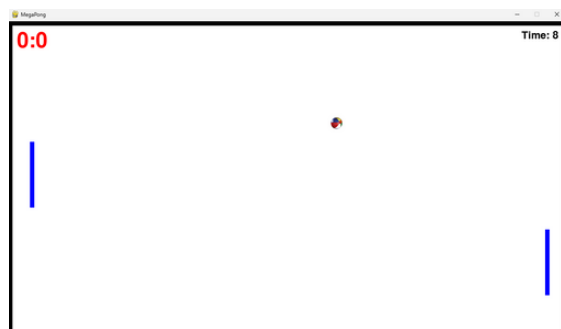
time_string = count // 30 # change count to seconds

count_text = font_time.render("Time: " + str(time_string), True, BLACK)
count_rect = count_text.get_rect()

count_rect.top_right = (scrWidth - 20, 20) # Timer top right
screen.blit(count_text, count_rect)
```



Play Time = 3  
Step = 8



Play Time = 8  
Step = 16

- EXPLAIN:**
1. Initialize play time for each round
  2. Define an acceleration function and use it when  $\text{time\_counter} \% 250 = 0$
  3. Initialize game time
  4. Update game time
  5. Display game time

### 3. As a player's score increases, the length of their paddle decreases.

```
class Paddle(BlockSprite):
    def __init__(self, x, y, length):
        super().__init__(x, y-length//2, 10, length, BLUE) # pad
        self.size = length
        self.length = length

    def update_length(self, score):
        self.size = max(self.length - score * 20, 50)
        self.image = pygame.Surface((10, self.size))
        self.image.fill(BLUE)
        self.rect = self.image.get_rect(center=self.rect.center)
```

```
# create two paddles
leftPaddle = Paddle(50, scrHeight/2, 150)
rightPaddle = Paddle(scrWidth-50, scrHeight/2, 150)
```

```
if scoreLeft < WINNING_SCORE:
    leftPaddle.update_length(scoreLeft)
if scoreRight < WINNING_SCORE:
    rightPaddle.update_length(scoreRight)
```



**EXPLAIN:**

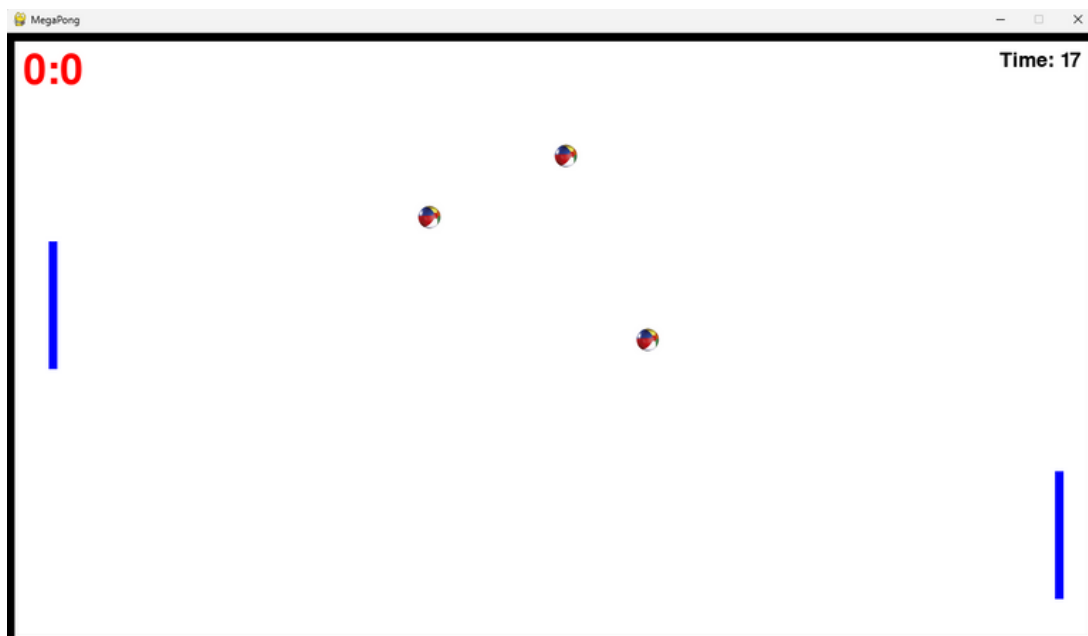
1. define an paddle length update function
2. update paddle length in every game loop

4. If the game continues past a certain time, then the number of balls increases to two, and finally to three.

```
def spawnBall(self):
    global numBalls
    if numBalls < 3:
        newBall = BallSprite('smallBall.png')
        newBall.rect.center = [scrWidth/2, scrHeight/2]
        newBall.xStep, newBall.yStep = self.randomSteps()
        sprites.add(newBall)
        numBalls += 1
```

```
# update game
if not gameOver:
    leftPaddle.move(leftStep)
    rightPaddle.move(rightStep)
    ball.update()
    sprites.update()
```

```
self.time_counter += 1
if self.time_counter % 500 == 0: self.accStep()
if self.time_counter % 500 == 0 and numBalls < 3:
    self.spawnBall()
```



**EXPLAIN:**

1. define an ball spawner function
2. update sprites in every game loop
3. make a spawn condition

```

1 # MegaPong.py
2
3 import pygame, random
4 from pygame.locals import *
5 from pygame.font import *
6
7 # some colors
8 BLACK = ( 0, 0, 0)
9 WHITE = ( 255, 255, 255)
10 RED = ( 255, 0, 0)
11 GREEN = ( 0, 255, 0)
12 BLUE = ( 0, 0, 255)
13
14 WALL_SIZE = 10
15 STEP = 4
16
17 PADDLE_STEP = 10
18 LEFT = 0
19 RIGHT = 1
20
21 WINNING_SCORE = 10
22
23
24
25 class BlockSprite(pygame.sprite.Sprite):
26
27     def __init__(self, x, y, width, height, color=BLACK):
28         super().__init__()
29         self.image = pygame.Surface((width, height))
30         self.image.fill(color)
31         self.rect = self.image.get_rect()
32         self.rect.topleft = (x, y)
33
34
35 # -----
36
37 class Paddle(BlockSprite):
38
39     def __init__(self, x, y, length):
40         super().__init__(x, y-length/2, 10, length, BLUE) # paddle width & height
41         self.size = length
42         self.length = length
43
44
45     def update_length(self, score):
46         self.size = max(self.length - score * 20, 50)
47         self.image = pygame.Surface((10, self.size))
48         self.image.fill(BLUE)
49         self.rect = self.image.get_rect(center=self.rect.center)
50
51
52     def move(self, step):
53         if pygame.sprite.collide_rect(self, top) and (step < 0): # at top & going up
54             step = 0
55         elif pygame.sprite.collide_rect(self, bottom) and (step > 0):
56             step = 0
57         self.rect.y += step
58
59 # -----
60
61 class BallSprite(pygame.sprite.Sprite):
62
63     def __init__(self, fnm):
64         super().__init__()
65         self.image = pygame.image.load(fnm).convert_alpha()
66         self.rect = self.image.get_rect()
67         self.rect.center = (scrWidth/2, scrHeight/2)
68         # start position of the ball in center of window

```

```

69 self.xStep, self.yStep = self.randomSteps()
70 # step size and direction along each axis
71 self.time_counter = 0 # Initial playtime
72
73
74
75 def update(self):
76     global scoreLeft, scoreRight, numBalls
77     if pygame.sprite.collide_rect(self, leftPaddle) and (self.xStep < 0):
78         # hit left paddle and going left
79         self.xStep = -self.xStep # change direction
80
81     if pygame.sprite.collide_rect(self, rightPaddle) and (self.xStep > 0):
82         # hit right paddle and going right
83         self.xStep = -self.xStep # change direction
84
85     if pygame.sprite.collideany(self, horizWalls):
86         # change y-step direction at top and bottom sides
87         self.yStep = -self.yStep
88
89     if pygame.sprite.collideany(self, vertWalls):
90         # ball has reached left or right sides
91         if pygame.sprite.collide_rect(self, right):
92             scoreLeft += 1
93             # left side
94             scoreRight += 1
95         # reset the ball
96         self.rect.center = (scrWidth/2, scrHeight/2)
97         self.xStep, self.yStep = self.randomSteps()
98
99     self.rect.x += self.xStep # move the ball horizontally
100     self.rect.y += self.yStep # and vertically
101
102     self.time_counter += 1
103     if self.time_counter % 500 == 0: self.accStep()
104     if self.time_counter % 500 == 0 and numBalls < 3:
105         self.spawnBall()
106
107
108     def accStep(self):
109         self.xStep += STEP if self.xStep > 0 else -STEP
110         self.yStep += STEP if self.yStep > 0 else -STEP
111
112
113     def randomSteps(self):
114         # create a random +/- STEP pair
115         x = STEP
116         if random.random() > 0.5:
117             x = -x
118         y = STEP
119         if random.random() > 0.5:
120             y = -y
121         return [x, y]
122
123
124     def spawnBall(self):
125         global numBalls
126         if numBalls < 3:
127             newBall = BallSprite('smallBall.png')
128             newBall.rect.center = (scrWidth/2, scrHeight/2)
129             newBall.xStep, newBall.yStep = self.randomSteps()
130             sprites.add(newBall)
131             numBalls += 1
132
133 # -----
134
135     def centerImage(screen, im):
136         x = (scrWidth - im.get_width())/2
137

```

```
138 y = (scrHeight - im.get_height())/2
139 screen.blit(im, (x,y))
140
141 # ----- main -----
142
143 pygame.init()
144 screen = pygame.display.set_mode([1280,720])
145 screen.fill(WHITE)
146 pygame.display.set_caption("MegaPong")
147
148 scrWidth, scrHeight = screen.get_size()
149
150 # create wall sprites
151 top = BlockSprite(0, 0, scrWidth, WALL_SIZE)
152 bottom = BlockSprite(0, scrHeight-WALL_SIZE, scrWidth, WALL_SIZE)
153 left = BlockSprite(0, 0, WALL_SIZE, scrHeight)
154 right = BlockSprite(scrWidth-WALL_SIZE, 0, WALL_SIZE, scrHeight)
155
156 horizWalls = pygame.sprite.Group(top, bottom)
157 vertWalls = pygame.sprite.Group(left, right)
158
159 # create two paddles
160 leftPaddle = Paddle(50, scrHeight/2, 150)
161 rightPaddle = Paddle(scrWidth-50, scrHeight/2, 150)
162
163 ball = BallSprite('smallBall.png')
164
165 sprites = pygame.sprite.OrderedUpdates(top, bottom, left, right, leftPaddle, rightPaddle, ball)
166
167 # game vars
168 leftStep = 0; rightStep = 0 # move step in pixels for paddles
169 scoreLeft = 0; scoreRight = 0
170 winMsg = ""
171 gameOver = False
172 count = 0
173 numBalls = 1
174
175 font = pygame.font.Font(None, 72)
176 font_time = pygame.font.Font(None, 36)
177
178 clock = pygame.time.Clock()
179
180 running = True
181 while running:
182     clock.tick(30)
183
184     # handle events
185     for event in pygame.event.get():
186         if event.type == QUIT:
187             running = False
188
189         if event.type == KEYDOWN:
190             if event.key == K_q: # left paddle
191                 leftStep = -PADDLE_STEP # up
192             elif event.key == K_s:
193                 leftStep = PADDLE_STEP # down
194
195             if event.key == K_p: # right paddle
196                 rightStep = -PADDLE_STEP # up
197             elif event.key == K_l:
198                 rightStep = PADDLE_STEP # down
199
200         elif event.type == KEYUP:
201             if event.key == K_q or event.key == K_s: # left paddle
202                 leftStep = 0
203             if event.key == K_p or event.key == K_l: # right paddle
204                 rightStep = 0
205
206
207 # update game
208 if not gameOver:
209     leftPaddle.move(leftStep)
210     rightPaddle.move(rightStep)
211     ball.update()
212     sprites.update()
213
214 if scoreLeft >= WINNING_SCORE:
215     winMsg = "Left Wins!"
216     gameOver = True
217 elif scoreRight >= WINNING_SCORE:
218     winMsg = "Right Wins!"
219     gameOver = True
220
221 count += 1 # count time
222
223 # redraw
224 screen.fill(WHITE)
225 sprites.draw(screen);
226
227 screen.blit( font.render(str(scoreLeft) + "." +
228                        str(scoreRight), True, RED), [20, 20])
229
230 time_string = count // 30 # change count to seconds
231
232 count_text = font_time.render("Time: " + str(time_string), True, BLACK)
233 count_rect = count_text.get_rect()
234
235 count_rect.topright = (scrWidth - 20, 20) # Timer top right
236 screen.blit(count_text, count_rect)
237
238 if scoreLeft < WINNING_SCORE:
239     leftPaddle.update_length(scoreLeft)
240 if scoreRight < WINNING_SCORE:
241     rightPaddle.update_length(scoreRight)
242
243 if gameOver:
244     centerImage(screen, font.render(winMsg, True, RED))
245
246 pygame.display.update()
247
248 pygame.quit()
```