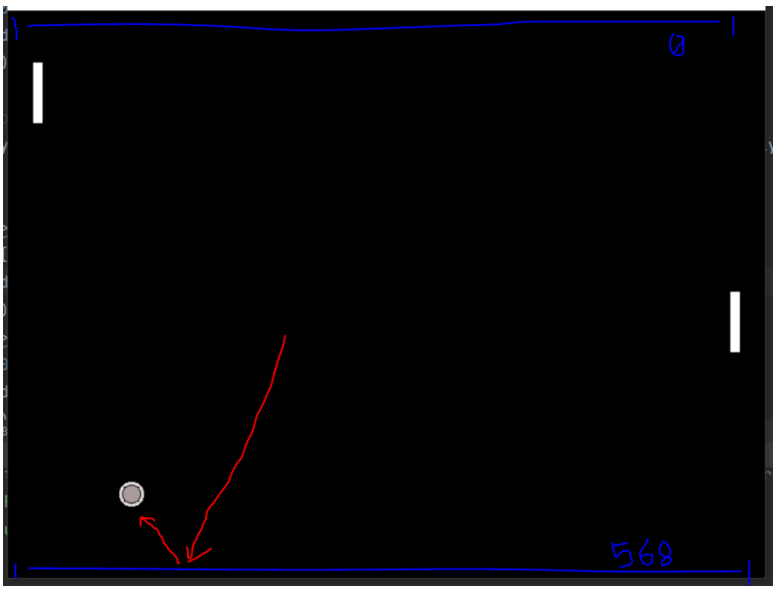
Coding documentation

- **day 1**

```
32  def corner(y):
33      global ballyd
34      if y <= 0 or y >= 568:
35          ballyd *= -1
36      return ballyd
```

```
93      # display ball and its changed position #
94      ballx += ballxd
95      bally += ballyd
96      corner(bally)
97      bounce()
98      ball(ballx, bally)
```



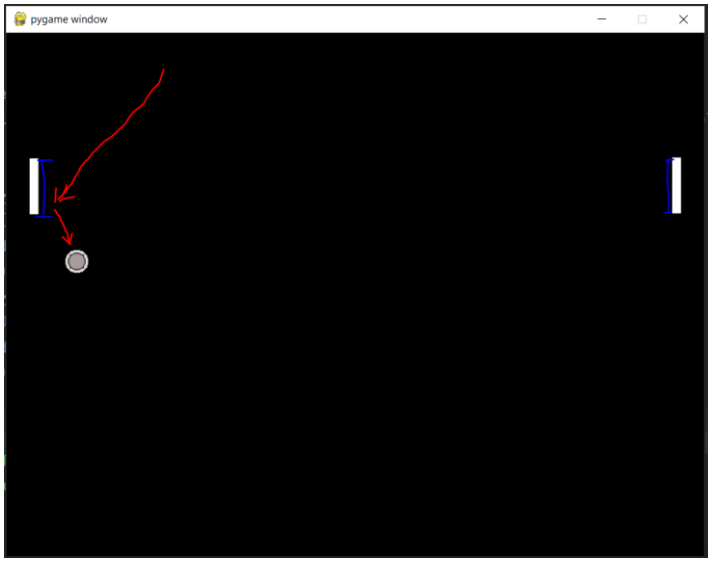
(Figure 1)

- The function “corner” is used to bounce ball between corners.

```
37  def bounce():
38      global player1x,player1y,player2x,player2y,ballx,bally,ballxd,ballyd
39      if ((ballx <= player1x + 37)and(ballx >= player1x + 28)) and ((bally+28 >= player1y)and(bally <= player1y + 64)):
40          ballx = 37
41          ballxd *= -1
42          if ((bally + 28 >= player1y + 24) and (bally <= player1y + 40)):
43              anglenear = [0.03,-0.03,0.05,-0.05]
44              ballyd = random.choice(anglenear)
45              print(ballyd)
46          elif ((bally+28 >= player1y)and(bally <= player1y + 64)):
47              anglefar = [0.1,-0.1,0.15,-0.15]
48              ballyd = random.choice(anglefar)
49              print(ballyd)
50
51      br = 28 # br means "bottom right"
52      if ((ballx+br <= player2x + 37)and(ballx+br >= player2x + 28)) and ((bally+28 >= player2y)and(bally <= player2y + 64)):
53          ballx = 735
54          ballxd *= -1
55          if ((bally + 28 >= player2y + 24) and (bally <= player2y + 40)):
56              anglenear = [0.03, -0.03, 0.05, -0.05]
57              ballyd = random.choice(anglenear)
58              print(ballyd)
59          elif ((bally+28 >= player2y)and(bally <= player2y + 64)):
60              anglefar = [0.1, -0.1, 0.15, -0.15]
61              ballyd = random.choice(anglefar)
62              print(ballyd)
63
64      return ballyd,ballxd
```

Player 1

Player 2

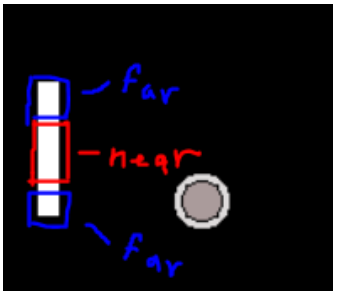



```
93      # display ball and its changed position #
94      ballx += ballxd
95      bally += ballyd
96      corner(bally)
97      bounce()
98      ball(ballx, bally)
```

(Figure 2)

- The function “bounce” is used to bounce ball from both players.

```
42          if ((bally + 28 >= player1y + 24) and (bally <= player1y + 40)):
43  near        anglenear = [0.03,-0.03,0.05,-0.05]
44              ballyd = random.choice(anglenear)
45              print(ballyd)
46          elif ((bally+28 >= player1y)and(bally <= player1y + 64)):
47  far         anglefar = [0.1,-0.1,0.15,-0.15]
48              ballyd = random.choice(anglefar)
49              print(ballyd)
```

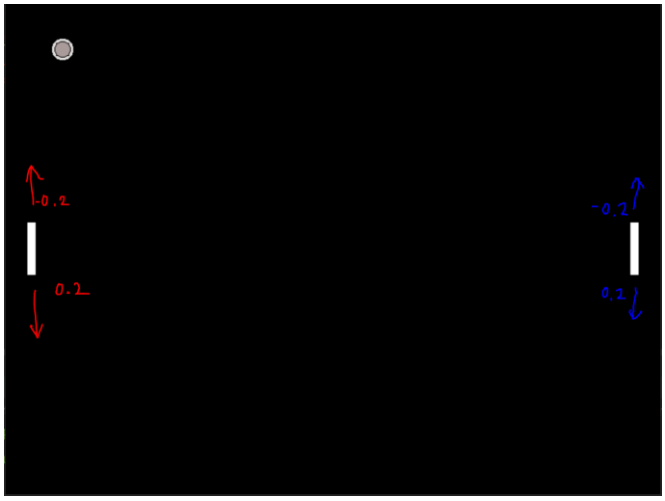


(Figure 3)

- The if and elif conditions inside the “bounce” function will change the ball movement as it hits the player’s paddle.
- The “near” part of paddle will give the ball y direction to either 0.03, -0.03, 0.05 and -0.05 randomly
- The “far” part of paddle will give the ball y direction to either 0.1, -0.1, 0.15 and -0.15 randomly
- ***Update for day 2: (now changed to 0.3, -0.3, 0.5, -0.5 for near; 1.0 -1.0, 1.5, -1.5 for far)***

```
67  while gamerun == True:
68      for event in pygame.event.get():
69          if event.type == pygame.QUIT:
70              gamerun = False
71
72          # ======== pressed button ======== #
73          if event.type == pygame.KEYDOWN:
74              if event.key == pygame.K_w:
75                  player1yd = -0.2
76              if event.key == pygame.K_UP:
77                  player2yd = -0.2
78  Player 1 movement
79              if event.key == pygame.K_s:
80                  player1yd = 0.2
81              if event.key == pygame.K_DOWN:
82                  player2yd = 0.2
83
84          # ======== released button ======== #
85          if event.type == pygame.KEYUP:
86              if event.key == pygame.K_w or event.key == pygame.K_s:
87                  player1yd = 0
88              if event.key == pygame.K_UP or event.key == pygame.K_DOWN:
89                  player2yd = 0
90
```

Player 2 movement

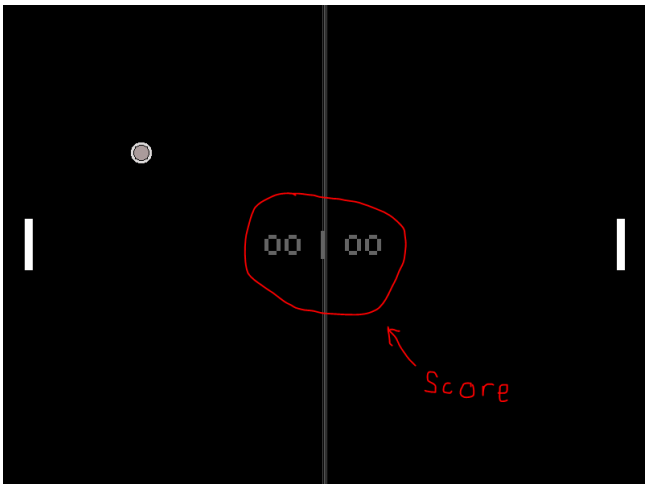


(Figure 4)

- The keys “W”, ”S”, ”Up arrow” and “Down arrow” is assigned to control the player.

- **Day 2**

```
22  def scorenumbers():
23      global ballxd,ballx,bally,p1score,p2score,scoredisplay
24      if ballx <= 0:
25          p2score += 1
26          ballxd = -2.0 # resets ball speed
27          ballx = 386
28          bally = 286
29          return p1score
30      elif ballx+28 >= 800:
31          p1score += 1
32          ballxd = 2.0 # resets ball speed
33          ballx = 386
34          bally = 286
35          return p2score
```




```
202         # display score system #
203         scoredisplay = "{} | {}".format(str(p1score).zfill(2), str(p2score).zfill(2))
204         gameover(p1score, p2score)
205
206         scoretext = scorefont.render(scoredisplay, False, (100, 100, 100))
207         centerposition = scoretext.get_rect(center=(400, 295))
208         scorenumbers()
209         screen.blit(scoretext, centerposition)
```

(Figure 5)

- The function “scorenumber” is used to update scores for p1score and p2score…
- After, the “screen.blit” in line 209 will display a score on screen.

```
37  def gameover(a,b):
38      global scoredisplay,ballx,ballxd,ballyd,isplayer1won,isplayer2won
39      winscore = 10
40      if a == winscore:
41          scoredisplay = "Player 1 won!!!"
42          ballx = 386
43          ballxd = 0
44          ballyd = 0
45          ballpng.set_alpha(0)
46          isplayer1won = True
47      elif b == winscore:
48          scoredisplay = "Player 2 won!!!"
49          ballx = 386
50          ballxd = 0
51          ballyd = 0
52          ballpng.set_alpha(0)
53          isplayer2won = True
54      return scoredisplay,isplayer1won,isplayer2won
```


```
202         # display score system #
203         scoredisplay = "{} | {}".format(str(p1score).zfill(2), str(p2score).zfill(2))
204         gameover(p1score, p2score)
```

(Figure 6)

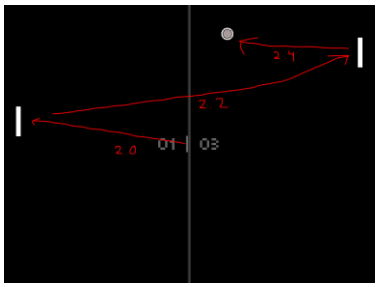
- The function “gameover” is used to stop game if one of the player is scored 10.

```python
def bounce():
    global player1x,player1y,player2x,player2y,ballx,bally,ballxd,ballyd
    if ((ballx <= player1x + 37)and(ballx >= player1x + 28)) and ((bally+28 >= player1y)and(bally <= player1y + 64)):
        ballx = 37
        ballxd *= -1
        if ((bally + 28 >= player1y + 24) and (bally <= player1y + 40)):
            anglenear = [0.3,-0.3,0.5,-0.5]
            ballyd = random.choice(anglenear)
            print(ballyd)
        elif ((bally+28 >= player1y)and(bally <= player1y + 64)):
            anglefar = [1.0,-1.0,1.5,-1.5]
            ballyd = random.choice(anglefar)
            print(ballyd)
        ballxd += 0.2  # increase speed of ball movement

    br = 28 # br means "bottom right"
    if ((ballx+br <= player2x + 37)and(ballx+br >= player2x + 28)) and ((bally+br >= player2y)and(bally <= player2y + 64)):
        ballx = 735
        ballxd *= -1
        if ((bally + 28 >= player2y + 24) and (bally <= player2y + 40)):
            anglenear = [0.3,-0.3,0.5,-0.5]
            ballyd = random.choice(anglenear)
            print(ballyd)
        elif ((bally+28 >= player2y)and(bally <= player2y + 64)):
            anglefar = [1.0,-1.0,1.5,-1.5]
            ballyd = random.choice(anglefar)
            print(ballyd)
        ballxd -= 0.2  # increase speed of ball movement

    return ballyd,ballxd
```

```python
def scorenumbers():
    global ballxd,ballx,bally,p1score,p2score,scoredisplay
    if ballx <= 0:
        p2score += 1
        ballxd = -2.0 # resets ball speed
        ballx = 386
        bally = 286
        return p1score
    elif ballx+28 >= 800:
        p1score += 1
        ballxd = 2.0 # resets ball speed
        ballx = 386
        bally = 286
        return p2score
```



(Figure 7)

- Every time a ball hits to the player's paddle, the ball movement increases by 0.2 and so on.
- If the ball hits to the player's goal. The ball movement resets to 2.0

```python
speed = pygame.time.Clock()
gamerun = True  # to run whole game window
gameresume = True  # to toggle pause and resume
```

```python
def pause(state):
    global gameresume
    if state == True:
        print("game paused")
        gameresume = False
    else:
        print("game resume")
        gameresume = True
```

```python
        # ======== pressed button ======== #
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_w:
                player1yd = -4.0
            if event.key == pygame.K_s:
                player1yd = 4.0
            if isplayer1won == True:
                if event.key == pygame.K_a:
                    player1xd = -4.0
                if event.key == pygame.K_d:
                    player1xd = 4.0

            if event.key == pygame.K_UP:
                player2yd = -4.0
            if event.key == pygame.K_DOWN:
                player2yd = 4.0
            if isplayer2won == True:
                if event.key == pygame.K_LEFT:
                    player2xd = -4.0
                if event.key == pygame.K_RIGHT:
                    player2xd = 4.0

            # if pressed space then pause #
            if event.key == pygame.K_SPACE:
                pause(gameresume)
```

```python
    if gameresume == False:
        player1x = player1x
        player1xd = player1xd
        player1y = player1y
        player1yd = player1yd
        player2x = player2x
        player2xd = player2xd
        player2y = player2y
        player2yd = player2yd
        pygame.draw.rect(screen, (100, 0, 0), pygame.Rect(0,0,800,600), 10)
        screen.blit(blackimage,(0,0))
        screen.blit(pausetext, (pausetext.get_rect(center=(400,40))))
        pygame.display.update()
```

```python
    else:
        # ================ this line and below is to blit images, (from back to front) ================== #
        screen.fill((0, 0, 0))
        screen.blit(bgpng, (0, 0))
```

(Figure 8)

- Adds the “pause” function to pause the game by pressing space bar keys.