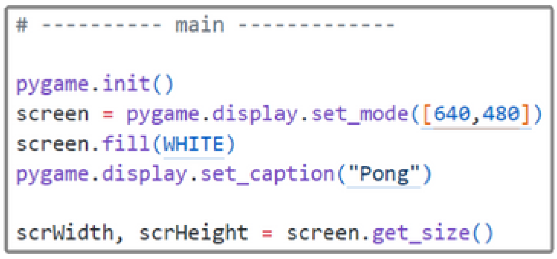
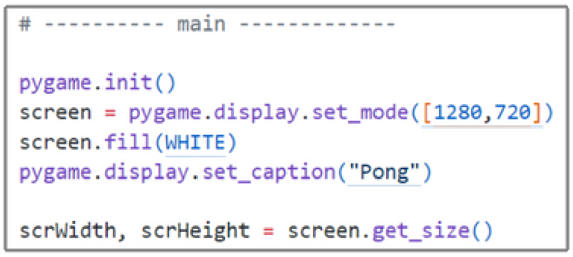
Nakrit Kreechaichana 6510110212

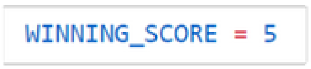
1. Increase the size of the game area, and make the winning score higher.



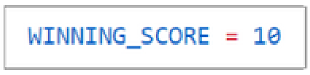
```
# ---------- main -------------

pygame.init()
screen = pygame.display.set_mode([640,480])
screen.fill(WHITE)
pygame.display.set_caption("Pong")

scrWidth, scrHeight = screen.get_size()
```



```
# ---------- main -------------

pygame.init()
screen = pygame.display.set_mode([1280,720])
screen.fill(WHITE)
pygame.display.set_caption("Pong")

scrWidth, scrHeight = screen.get_size()
```

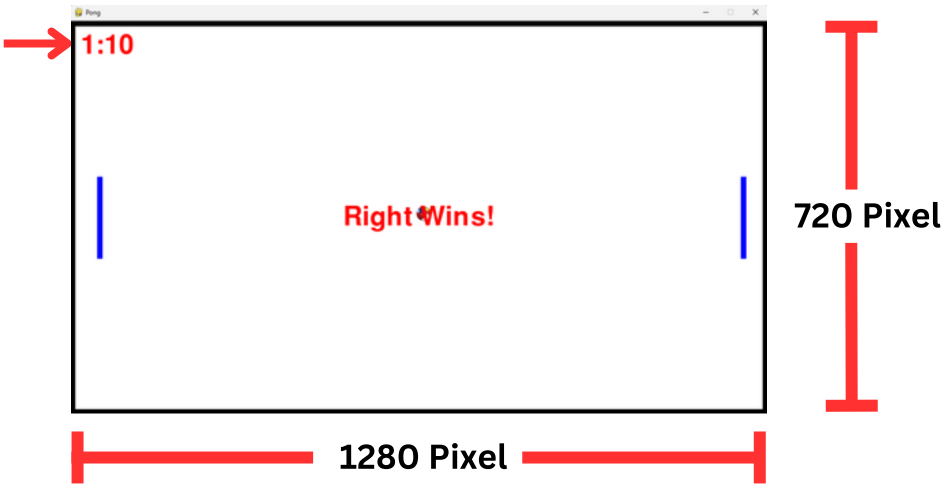


```
WINNING_SCORE = 5
```



```
WINNING_SCORE = 10
```

BEFORE 1:5
AFTER 1:10

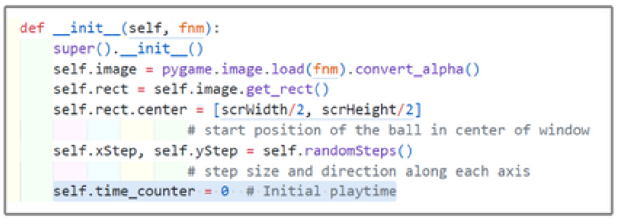


EXPLAIN:
- Change a screen size from [640, 480] to [1280, 720]
- Change winning score from 5 to 10

Nakrit Kreechaichana 6510110212

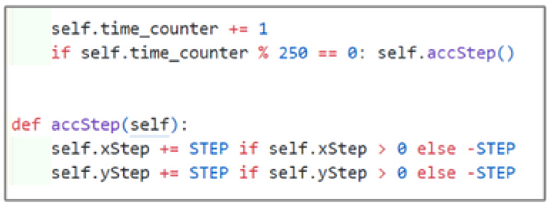
2. Make the ball gradually move faster as the game playing time increases.

1.



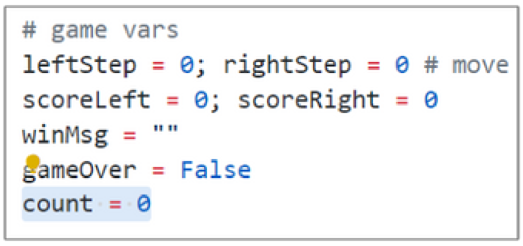
```
def __init__(self, fnm):
    super().__init__()
    self.image = pygame.image.load(fnm).convert_alpha()
    self.rect = self.image.get_rect()
    self.rect.center = [scrWidth/2, scrHeight/2]
                    # start position of the ball in center of window
    self.xStep, self.yStep = self.randomSteps()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

2.



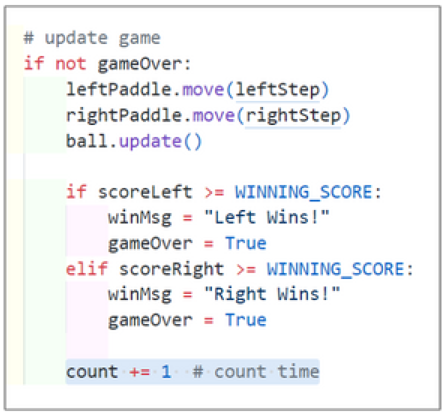
```
    self.time_counter += 1
    if self.time_counter % 250 == 0: self.accStep()


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

3.



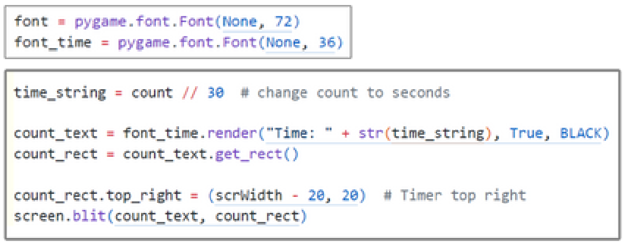
```
# game vars
leftStep = 0; rightStep = 0 # move
scoreLeft = 0; scoreRight = 0
winMsg = ""
gameOver = False
count = 0
```

4.



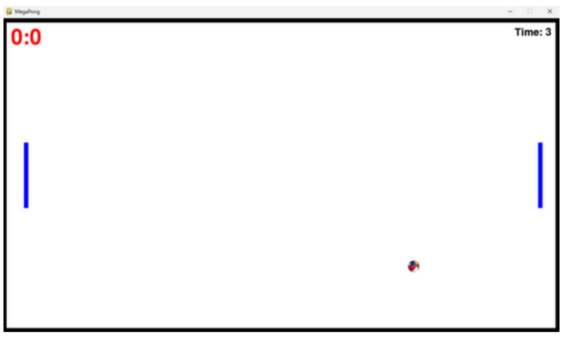
```
# update game
if not gameOver:
    leftPaddle.move(leftStep)
    rightPaddle.move(rightStep)
    ball.update()

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

    count += 1  # count time
```

5.

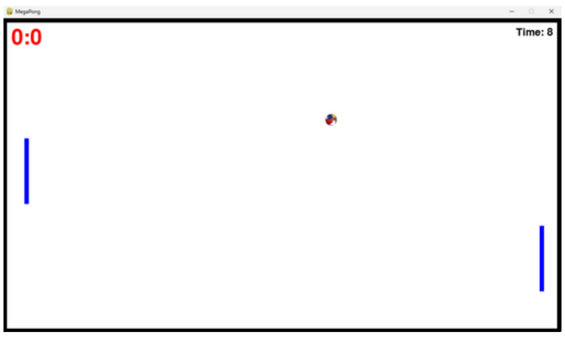


```
font = pygame.font.Font(None, 72)
font_time = pygame.font.Font(None, 36)
```

```
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

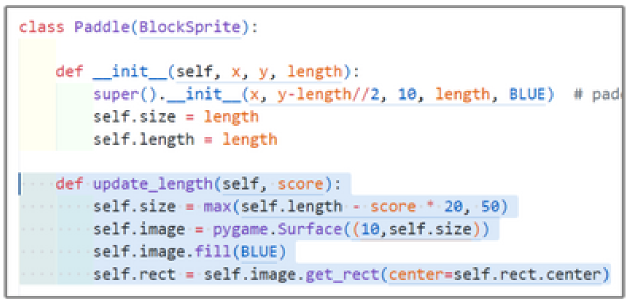


Play Time = 8
Step = 16

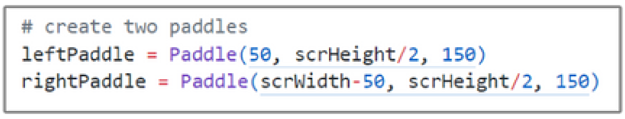
EXPLAIN:
1. Initialize play time for each round
2. defnie an acceleration function and use it when time_counter % 250 = 0
3. Initialize game time
4. Update game time
5. Display game time

Nakrit Kreechaichana 6510110212

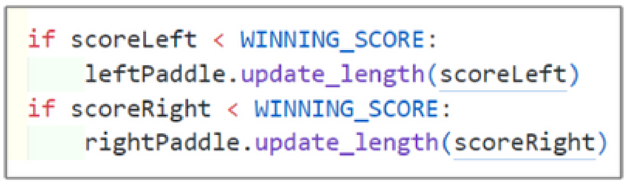
3. As a player's score increases, the length of their paddle decreases.



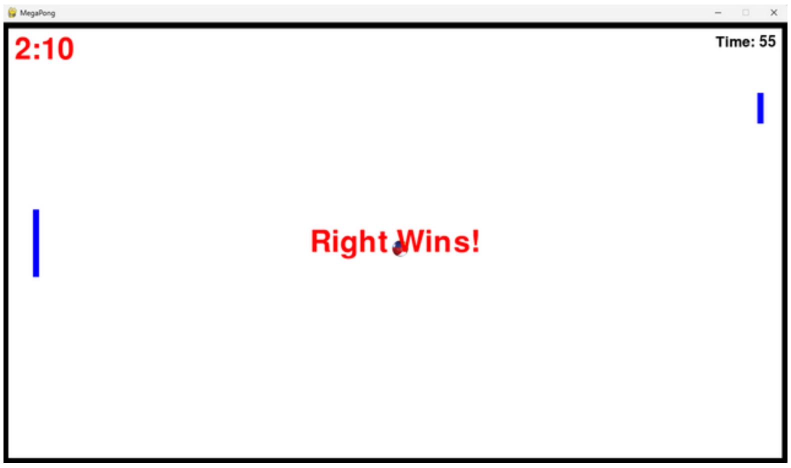
```
class Paddle(BlockSprite):

    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)
```

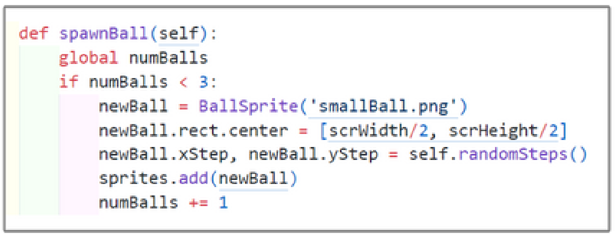


```
if scoreLeft < WINNING_SCORE:
    leftPaddle.update_length(scoreLeft)
if scoreRight < WINNING_SCORE:
    rightPaddle.update_length(scoreRight)
```

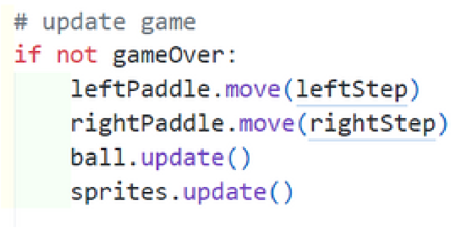


EXPLAIN:
1. defnie an paddle length update function
2. update paddle length in every game loop

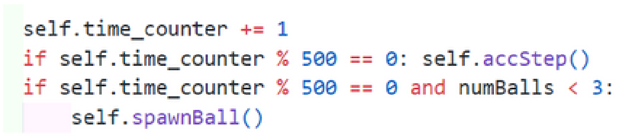
4. If the game continues past a certain time, then the number of balls increases to two, and finally to three.



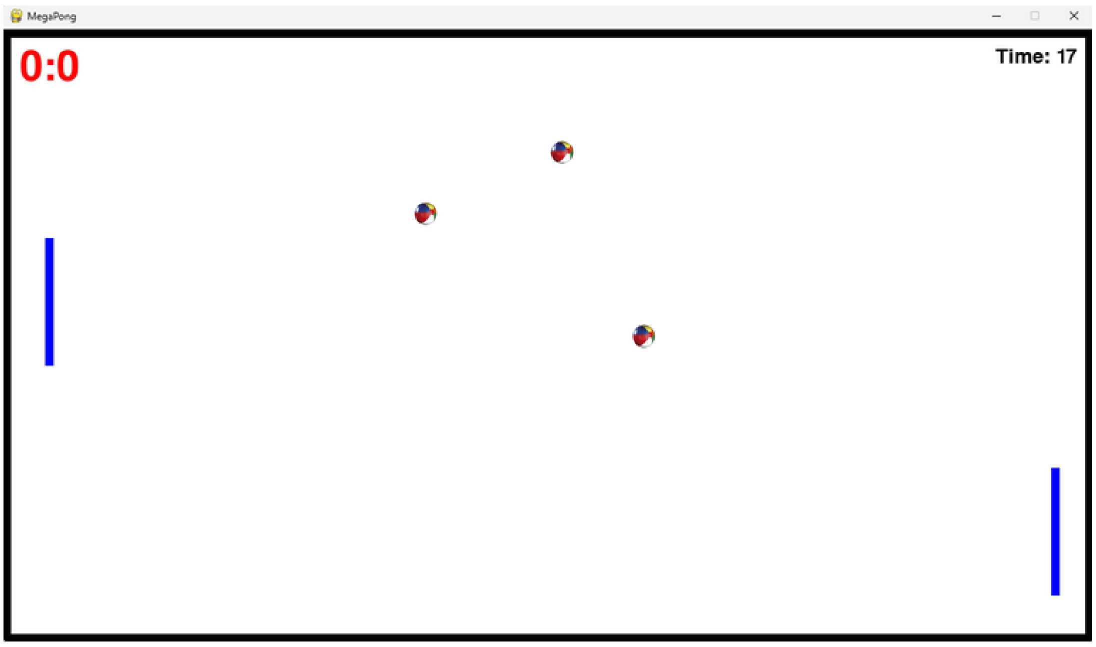
```python
def spawnBall(self):
    global numBalls
    if numBalls < 3:
        newBall = BallSprite('smallBall.png')
        newBall.rect.center = [scrWidth/2, scrHeight/2]
        newBall.xStep, newBall.yStep = self.randomSteps()
        sprites.add(newBall)
        numBalls += 1
```



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



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



<u>EXPLAIN</u>:
1. define an ball spawner function
2. update sprites in every game loop
3. make a spawn condition

```python
# MegaPong.py

import pygame, random
from pygame.locals import *
from pygame.font import *

# some colors
BLACK = (   0,   0,   0)
WHITE = ( 255, 255, 255)
RED   = ( 255,   0,   0)
GREEN = (   0, 255,   0)
BLUE  = ( 0,   0,   255)

WALL_SIZE = 10
STEP = 4

PADDLE_STEP = 10
LEFT  = 0
RIGHT = 1

WINNING_SCORE = 10



class BlockSprite(pygame.sprite.Sprite):

    def __init__(self, x, y, width, height, color=BLACK):
        super().__init__()
        self.image = pygame.Surface((width, height))
        self.image.fill(color)
        self.rect = self.image.get_rect()
        self.rect.topleft = (x, y)


# -----------------------------------------------------------

class Paddle(BlockSprite):

    def __init__(self, x, y, length):
        super().__init__(x, y-length//2, 10, length, BLUE)  # paddle width & height
        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)

    def move(self, step):
        if pygame.sprite.collide_rect(self, top) and (step < 0):  # at top & going up
            step = 0
        elif pygame.sprite.collide_rect(self, bottom) and (step > 0):
            # at bottom and going down
            step = 0
        self.rect.y += step


# -----------------------------------------------------------

class BallSprite(pygame.sprite.Sprite):

    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
```

```python
        self.xStep, self.yStep = self.randomSteps()
                    # step size and direction along each axis
        self.time_counter = 0  # Initial playtime


    def update(self):
        global scoreLeft, scoreRight, numBalls
        if pygame.sprite.collide_rect(self, leftPaddle) and (self.xStep < 0):
            # hit left paddle and going left
            self.xStep = -self.xStep    # change direction

        elif pygame.sprite.collide_rect(self, rightPaddle) and (self.xStep > 0):
            # hit right paddle and going right
            self.xStep = -self.xStep    # change direction

        if pygame.sprite.spritecollideany(self, horizWalls):
            # change y-step direction at top and bottom sides
            self.yStep = -self.yStep

        if pygame.sprite.spritecollideany(self, vertWalls):
            # ball has reached left or right sides
            if pygame.sprite.collide_rect(self, right):
                scoreLeft += 1
            else:   # left side
                scoreRight += 1

            # reset the ball
            self.rect.center = (scrWidth/2, scrHeight/2)
            self.xStep, self.yStep = self.randomSteps()

        self.rect.x += self.xStep   # move the ball horizontally
        self.rect.y += self.yStep   # and vertically

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


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


    def randomSteps(self):
        # create a random +/- STEP pair
        x = STEP
        if random.random() > 0.5:
            x = -x
        y = STEP
        if random.random() > 0.5:
            y = -y
        return [x,y]


    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


# ----------------------------------

def centerImage(screen, im):
    x = (scrWidth - im.get_width())/2
```

Nakrit Kreechaichana 6510110212

```python
    y = (scrHeight - im.get_height())/2
    screen.blit(im, (x,y))


# ---------- main -------------

pygame.init()
screen = pygame.display.set_mode([1280,720])
screen.fill(WHITE)
pygame.display.set_caption("MegaPong")

scrWidth, scrHeight = screen.get_size()

# create wall sprites
top    = BlockSprite(0, 0, scrWidth, WALL_SIZE)
bottom = BlockSprite(0, scrHeight-WALL_SIZE, scrWidth, WALL_SIZE)
left   = BlockSprite(0, 0, WALL_SIZE, scrHeight)
right  = BlockSprite(scrWidth-WALL_SIZE, 0, WALL_SIZE, scrHeight)

horizWalls = pygame.sprite.Group(top, bottom)
vertWalls = pygame.sprite.Group(left, right)

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

ball = BallSprite('smallBall.png')

sprites = pygame.sprite.OrderedUpdates(top, bottom, left, right, leftPaddle, rightPaddle, ball)

# game vars
leftStep = 0; rightStep = 0 # move step in pixels for paddles
scoreLeft = 0; scoreRight = 0
winMsg = ""
gameOver = False
count = 0
numBalls = 1

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

clock = pygame.time.Clock()

running = True
while running:
    clock.tick(30)

    # handle events
    for event in pygame.event.get():
        if event.type == QUIT:
            running = False

        if event.type == KEYDOWN:
            if event.key == K_q:   # left paddle
                leftStep = -PADDLE_STEP    # up
            elif event.key == K_s:
                leftStep = PADDLE_STEP     # down

            if event.key == K_p:   # right paddle
                rightStep = -PADDLE_STEP   # up
            elif event.key == K_l:
                rightStep = PADDLE_STEP    # down

        elif event.type == KEYUP:
            if event.key == K_q or event.key == K_s:   # left paddle
                leftStep = 0
            if event.key == K_p or event.key == K_l:   # right paddle
                rightStep = 0

```

```python

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

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

        count += 1  # count time


    # redraw
    screen.fill(WHITE)
    sprites.draw(screen);

    screen.blit( font.render(str(scoreLeft) + ":" +
                        str(scoreRight), True, RED), [20, 20])

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

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

    if gameOver:
        centerImage(screen, font.render(winMsg, True, RED))

    pygame.display.update()

pygame.quit()

```

Nakrit Kreechaichana 6510110212