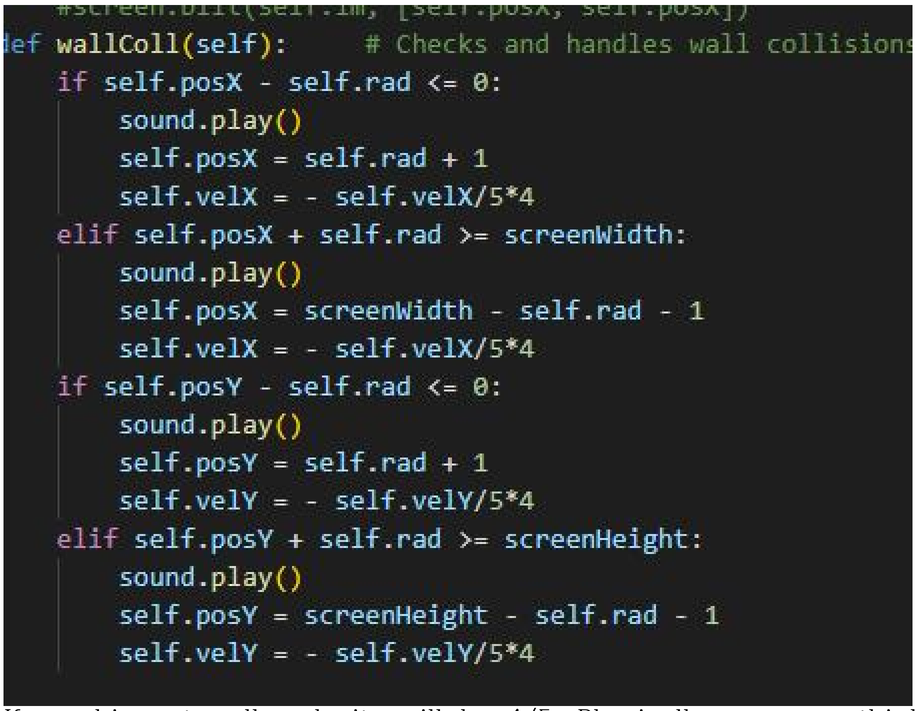
extra_hw1

20171300 지용환

We have to calculate velocity and position changes in the extra. But I don’t know about physics(conservation of momentum,momentum energy, ..ect). So I calculate in my way.

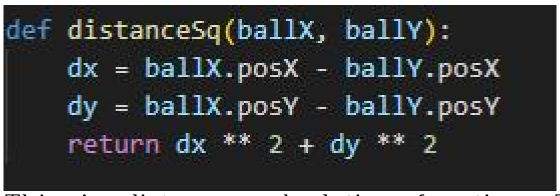
First in the cases of ball crashing at walls, most of them have already been written.



```
    #screen.blit(self.im, [self.posX, self.posX])
def wallColl(self):     # Checks and handles wall collisions
    if self.posX - self.rad <= 0:
        sound.play()
        self.posX = self.rad + 1
        self.velX = - self.velX/5*4
    elif self.posX + self.rad >= screenWidth:
        sound.play()
        self.posX = screenWidth - self.rad - 1
        self.velX = - self.velX/5*4
    if self.posY - self.rad <= 0:
        sound.play()
        self.posY = self.rad + 1
        self.velY = - self.velY/5*4
    elif self.posY + self.rad >= screenHeight:
        sound.play()
        self.posY = screenHeight - self.rad - 1
        self.velY = - self.velY/5*4
```

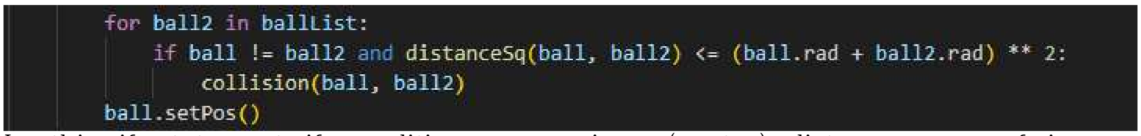
If crashing at wall, velocity will be 4/5. Physically, you can think of the wall as consuming 1/5 of the kinetic energy. Think of it as such a wall. (Is that correct?). Anyway, if crashig at wall, velocity of ball reduced. And can be stopped by crashing the wall.

Next is ball crashing.



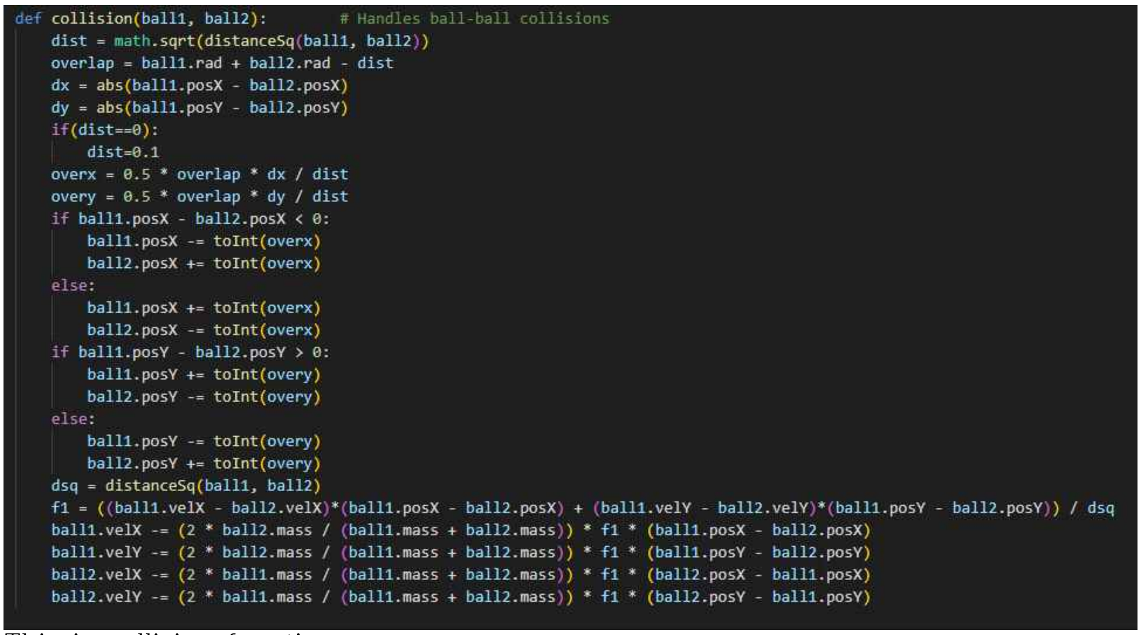
```
def distanceSq(ballX, ballY):
    dx = ballX.posX - ballY.posX
    dy = ballX.posY - ballY.posY
    return dx ** 2 + dy ** 2
```

This is distance calculating function. This returns the square of the distance between images. And using for loop in main() funciont(already explain in explain_hw1), all ball check below if statement, and then goes to collision.



```python
        for ball2 in ballList:
            if ball != ball2 and distanceSq(ball, ball2) <= (ball.rad + ball2.rad) ** 2:
                collision(ball, ball2)
        ball.setPos()
```

In this if statement if condition means image(center) distance<=sum of images radius. I,e this means the two balls collide.



```python
def collision(ball1, ball2):        # Handles ball-ball collisions
    dist = math.sqrt(distanceSq(ball1, ball2))
    overlap = ball1.rad + ball2.rad - dist
    dx = abs(ball1.posX - ball2.posX)
    dy = abs(ball1.posY - ball2.posY)
    if(dist==0):
        dist=0.1
    overx = 0.5 * overlap * dx / dist
    overy = 0.5 * overlap * dy / dist
    if ball1.posX - ball2.posX < 0:
        ball1.posX -= toInt(overx)
        ball2.posX += toInt(overx)
    else:
        ball1.posX += toInt(overx)
        ball2.posX -= toInt(overx)
    if ball1.posY - ball2.posY > 0:
        ball1.posY += toInt(overy)
        ball2.posY -= toInt(overy)
    else:
        ball1.posY -= toInt(overy)
        ball2.posY += toInt(overy)
    dsq = distanceSq(ball1, ball2)
    f1 = ((ball1.velX - ball2.velX)*(ball1.posX - ball2.posX) + (ball1.velY - ball2.velY)*(ball1.posY - ball2.posY)) / dsq
    ball1.velX -= (2 * ball2.mass / (ball1.mass + ball2.mass)) * f1 * (ball1.posX - ball2.posX)
    ball1.velY -= (2 * ball2.mass / (ball1.mass + ball2.mass)) * f1 * (ball1.posY - ball2.posY)
    ball2.velX -= (2 * ball1.mass / (ball1.mass + ball2.mass)) * f1 * (ball2.posX - ball1.posX)
    ball2.velY -= (2 * ball1.mass / (ball1.mass + ball2.mass)) * f1 * (ball2.posY - ball1.posY)
```

This is collision function.

First, dist is distnace of center of image. And overlap is overlap distance. Thses are varailbe, created to prevent errors that occur when the location is completely overlapped when first created and when the collision happens too fast and the balls overlap. Since dx/dist mean cos, overx mean half of the overlapping length along the x-axis simiarly, dy/dist mean y-axis. The two balls whose positions overlap are reassigned using these variables so that the positions do not overlap according to each case (ball1 is the ball on the left, ball is the ball on the right, etc.). The important is change of velocity. The below 6 lines are about velocity. dsp means the square of the distance. And note that (ba11.posX-ball2.posX)^2/dsp mean cos^2 and similarly (ba11.posY-ball2.posY)^2/dsp is sin^2. I don't understand much about this formula, but let's explain it: the part calculated with mass means the force of velocity, and the latter part means the direction of velocity. In the end, this formula represents the rate of change of velocity, and subtracting it from the original velocity results in the later velocity. This way, you can actually achieve quite a bit of the direction in which the images bounces, and even the change in speed.