2022 년 2 학기 게임 프로그래밍 입문 (실습 2)

이름: 정은성 학과: 원자력 공학과 학번: 2021103751

1. 공을 ball2.png로 변경

- 방법1) Ball.get_position을 overriding

Description]

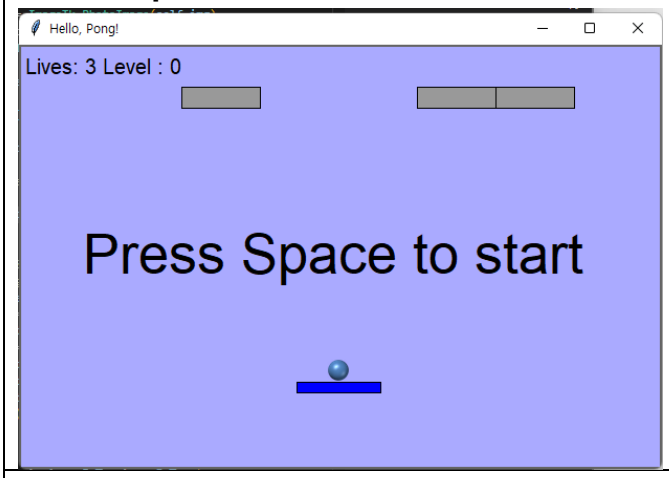
방법 2를 채택함

Ball은 이미지를 집어넣었다. 때문에 이를 get_position 함수를 받게 된다면, 원의 중심의 x와 y좌표를 받게 됨 이는 다른 함수에서 받아 사용할 때 문제가 됨 때문에 ball 안에 get_position의 함수를 오버라이딩해서 coords를 이용해 값을 받는 것을 바꾸어서 왼쪽 위의 x값, y값 그리고 오른쪽 아래의 x값, y값을 반환한다.

Code]

```
class Ball(GameObject):
    def __init__(self, canvas, x, y):
        self.radius = 10
        self.direction = [0.707, 0.707]
        self.speed = 10
        #사진의 저장 위치
        self.img =
Image.open("C:\\KyungHee\\kyunghee\\게임프로그래밍입문\\Practice\\ball.png")
        self.img = self.img.resize((20, 20), Image.ANTIALIAS)
        self.ball_img = ImageTk.PhotoImage(self.img)
        item = canvas.create_image(x, y, image = self.ball_img)
        super(Ball, self).__init__(canvas, item)
    # 함수 오버라이딩 구현
    def get_position(self):
        self.coords_xy = self.canvas.coords(self.item)
        self.coords = [self.coords_xy[0] - self.radius, self.coords_xy[1] - self.radius,\
                       self.coords_xy[0] + self.radius, self.coords_xy[1] + self.radius]
        return self.coords
```

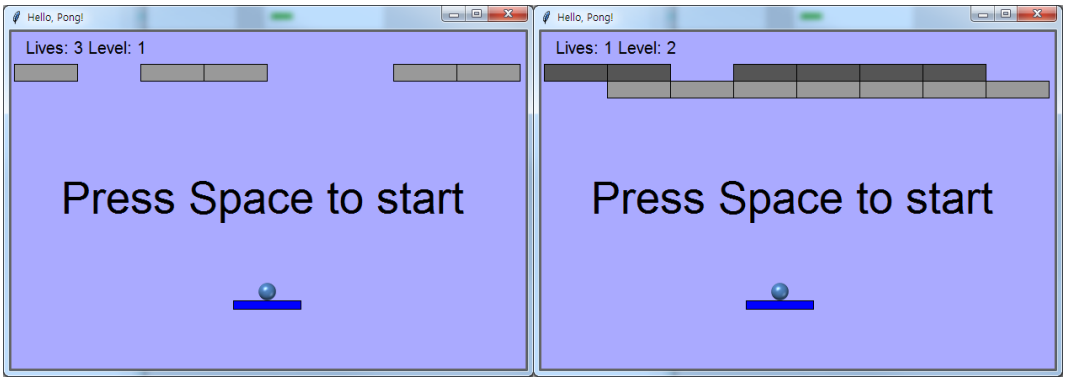
Game Shot]



2022 년 2 학기 게임 프로그래밍 입문 (실습 2)

이름: 정은성 학과: 원자력 공학과 학번: 2021103751

2. Level 추가, 벽돌 층수 변경, 임의의 벽돌만 생성



- Game.__int__에서 self.level = 1로 초기화
- Game.game_loop에서 num_bricks == 0인 경우에 level 증가, <space> binding 등의 추가 필요
- Game.update_lives_text에 level 표시: '%d %d' % (x, y)
- Level 시작할 때, 벽돌 생성을 함수로 변경

  ex) Game.setup_level(self)

  . Game.level에 따른 벽돌 층수 변경 및 임의의 벽돌만 생성

  . random.randint(0, 9) 사용

Description]

Self.level을 통해 지금 몇 단계인지 확인할 수 있는 변수를 만들어 준다.

Setup_game에 level을 넣도록 수정해서 레벨에 따라서 어떻게 벽돌을 생성할지 구분함.

Game.update_live_text에 "Lives: %s Level : %s" % (self.lives, self.level)이라고 적어 level을 추가함.

또한 벽돌 생성을 random 함수로 구현해 어떻게 생성될지에 대한 여부를 남겨둠

Code]

<벽돌이 0개이면 레벨 증가 및 다시 시작>

```
def game_loop(self):
    self.check_collisions()
    num_bricks = len(self.canvas.find_withtag('brick'))
    if num_bricks == 0:
        self.level += 1
        self.ball.speed = None
        # 코드 수정부
        self.ball.update()
        self.reset = 1
        self.after(1000, self.setup_game(self.level, self.reset))
```

```
def setup_game(self, level, reset):
    self.add_ball()
    if (reset == 1):
        self.make_brick(level)
```

이름: 정은성 학과: 원자력 공학과 학번: 2021103751

```
    self.update_lives_text()
    self.text = self.draw_text(300, 200, 'Press Space to start')
    self.canvas.bind('<space>', lambda _: self.start_game())
```

<목숨과 레벨을 순서대로 표시 하기 위한 코드>

```
def update_lives_text(self):
    text = "Lives: %s Level : %s" % (self.lives, self.level)
    if self.hud is None:
        self.hud = self.draw_text(80, 20, text, 15)
    else:
        self.canvas.itemconfig(self.hud, text=text)
```

<레벨 시작할 때, 벽돌 생성을 함수로 변경>

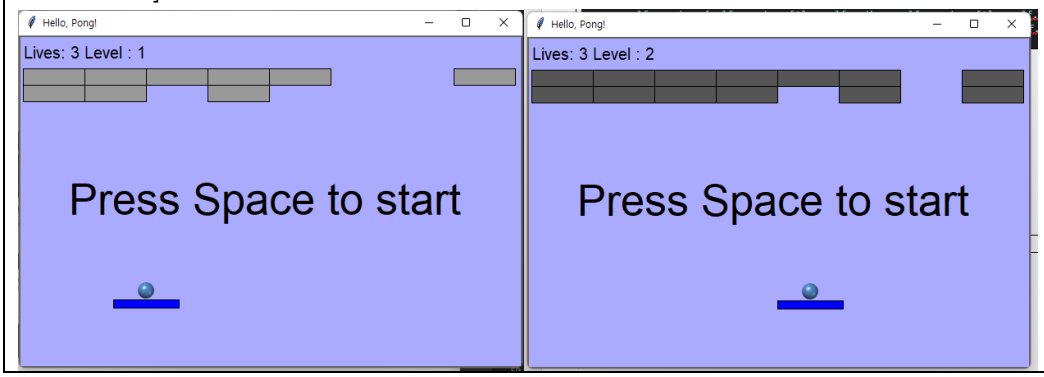
```
def make_brick(self, level):
    if (level == 1):
        for x in range(5, self.width - 5, 75):
            rand_num = random.randint(0, 9)
            if (rand_num % 2 == 0):
                pass
            else:
                self.add_brick(x + 37.5, 50, 1)
    elif (level == 2):
        for x in range(5, self.width - 5, 75):
            rand_num = random.randint(0, 9)
            if (rand_num % 4 == 0):
                pass
            else:
                self.add_brick(x + 37.5, 50, 1)
            if (rand_num % 2 == 0):
                pass
            else:
                self.add_brick(x + 37.5, 70, 1)
    elif (level == 3):
        for x in range(5, self.width - 5, 75):
            rand_num = random.randint(0, 9)
            if (rand_num % 4 == 0):
                pass
            else:
                self.add_brick(x + 37.5, 50, 2)
            if (rand_num % 2 == 0):
                pass
            else:
                self.add_brick(x + 37.5, 70, 2)
    elif (level == 4):
        for x in range(5, self.width - 5, 75):
            rand_num = random.randint(0, 9)
            if (rand_num % 4 == 0):
```

2022 년 2 학기 게임 프로그래밍 입문 (실습 2)

이름: 정은성 학과: 원자력 공학과 학번: 2021103751

```
                pass
            else:
                self.add_brick(x + 37.5, 50, 3)
            if (rand_num % 2 == 0):
                pass
            else:
                self.add_brick(x + 37.5, 70, 2)
            if (rand_num % 2 == 0):
                pass
            else:
                self.add_brick(x + 37.5, 90, 1)
    elif (level == 5):
        for x in range(5, self.width - 5, 75):
            rand_num = random.randint(0, 9)
            if (rand_num % 4 == 0):
                pass
            else:
                self.add_brick(x + 37.5, 50, 3)
            if (rand_num % 2 == 0):
                pass
            else:
                self.add_brick(x + 37.5, 70, 2)
            if (rand_num % 2 == 0):
                pass
            else:
                self.add_brick(x + 37.5, 90, 2)
```

Game Shot]

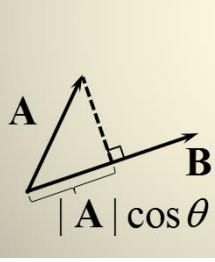


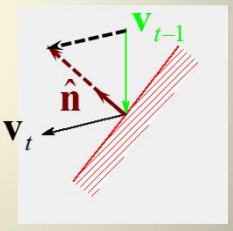
2022 년 2 학기 게임 프로그래밍 입문 (실습 2)

이름: 정은성 학과: 원자력 공학과 학번: 2021103751

3. 모서리에 충돌한 공의 충돌 반응 처리

$$\mathbf{v}_t = 2(-\mathbf{v}_{t-1}\cdot\hat{\mathbf{n}})\hat{\mathbf{n}} + \mathbf{v}_{t-1}$$

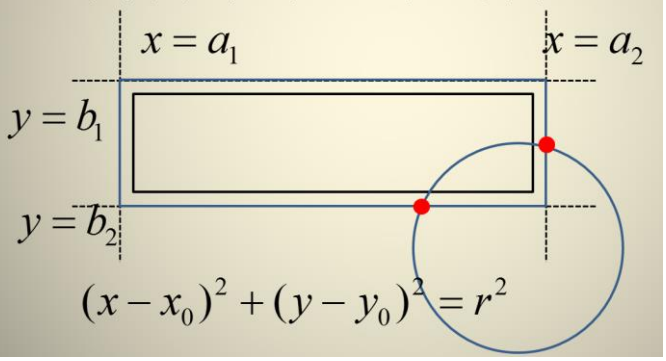




◈ Ball

❑ 생성자에서 방향 벡터의 크기를 1로 수정

❑ 네 개의 직선과 원의 교점을 리스트에 추가



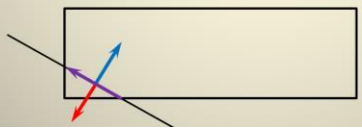
❑ 반응 계산

➢ 교점의 2개이면 아래의 수식을 이용하여 방향 벡터 수정

$$\mathbf{v}_t = 2(-\mathbf{v}_{t-1}\cdot\hat{\mathbf{n}})\hat{\mathbf{n}} + \mathbf{v}_{t-1}$$

➢ 교점을 잇는 벡터를 +90/-90도로 회전하여 두 개의 법선 벡터 생성

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$



❑ 교점 판별

➢ 교점의 x 또는 y를 계산하기 위하여 이차 방정식 생성

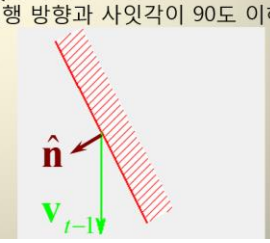
➢ 이차 방정식의 판별식이 0보다 작으면 교점이 없음

➢ 교점의 x 또는 y의 좌표가 brick이나 paddle와 겹침 확인
(예, $y=b_2$와의 교점의 x좌표가 a_1과 a_2 사이에 있는지 확인)

➢ 위의 조건을 만족하면 교점을 리스트에 추가

➢ 교점의 중심에서 물체의 중심을 잇는 벡터와의 각도가 크면 바깥쪽으로 향하는 법선 벡터

➢ 법선 벡터가 충돌하기 전의 진행 방향과 사잇각이 90도 이하이면 충돌 반응하지 않음



2022 년 2 학기 게임 프로그래밍 입문 (실습 2)

이름: 정은성 학과: 원자력 공학과 학번: 2021103751

Description]

<1. 충돌 메인 함수>

1-1. 우선적으로 충돌 메인 함수에서 충돌된 오브젝트의 4개의 꼭짓점의 위치를 구할 수 있다.

1-2. 원의 중심 좌표를 구한다. (이는 get_position_center() 함수로 구현한 것으로 이는 원의 중심 좌표가 나온다)

1-3. collide_where는 어디에서 충돌이 발생했는지 알려주며, 추가적으로 충돌하지 않으면 NULL을 받아 충돌 여부를 확인한다.

1-4. 만약 충돌이 발생하면 cal_reflection() 함수를 통해 반사 백터를 구현하고 충돌벡터의 움직임을 구현한다.

<2. 충돌 지점 처리>

주석으로 처리함 간단하게 다시 말하면 solution이라는 이차방정식 풀어주는 함수를 구현한 뒤에 각각 알 수 있는 값을 확인한 뒤 충돌한 x값과 y값을 구한다.

<3. 반사 백터 처리>

반사벡터를 구하기 위해 법선 백터와 진입 벡터를 더해서 반사벡터를 구해주고, 단위 백터로 계속해서 치환해주는 작업을 계속해준다.

<4. 충돌 90도 여부 처리>

사잇값 구하는 공식을 구현해 사잇값을 찾는다.

Code]

<1. 충돌 메인 함수 구현>

```
def collide(self, game_objects):
    coords = self.get_position()
    x = (coords[0] + coords[2]) * 0.5

    if len(game_objects) > 1:
        self.direction[1] *= -1
    elif len(game_objects) == 1:
        game_object = game_objects[0]

        # 사각형의 꼭짓점 위치
        coords = game_object.get_position()
        # 원의 중심 좌표 구하기
        ball_center = self.get_position_center()
        meet = self.collide_where(coords, ball_center)
        # 모서리 충돌이 발생함.
        if (meet):
            self.cal_reflection(meet)
        # 모서리 충돌이 발생하지 않음.
        else:
            if x > coords[2]:
                self.direction[0] *= 1
            elif x < coords[0]:
```

이름: 정은성 학과: 원자력 공학과 학번: 2021103751

```
                self.direction[0] *= -1
            else:
                self.direction[1] *= -1
```

<2. 충돌 지점 구하기>

```
# 충돌을 어디에서 발생했는지 리턴하는 함수
def collide_where(self, rectangle, circle):
    rectangle_xy = rectangle

    # 충돌 처리 영역을 확장하면 충돌 처리 직선을 바깥으로 그린것이 되고
    # 그것이 충돌 처리 위치의 기준이 됨
    rectangle_xy[0] -= 0.5
    rectangle_xy[1] -= 0.5
    rectangle_xy[2] += 0.5
    rectangle_xy[3] += 0.5

    # 원의 중심 x 좌표와 y 좌표를 구한다.
    circle_x = circle[0]
    circle_y = circle[1]

    # 충돌지점을 확인하기 위해 사각형의 중심이 되는 x 와 y 값을 구한다.
    check_x = (rectangle_xy[0] + rectangle_xy[2])/2
    check_y = (rectangle_xy[1] + rectangle_xy[3])/2

    # find_x 의 값은 초기에 none 으로 할당이 됨
    find_x = None
    find_y = None

    # 기준에 따라 충돌된 x 좌표와 y 좌표를 구할 수 있다.
    # 이는 사각형의 중심보다 위에 충돌이 되었다. 이러한 정보를 통해 특정되서 미리 알 수 있는 값은
    # 사각형의 아래 y 값, 원의 반지름, 원의 중심값
    # 이를 통해 알 수 있는 값은 사각형의 밑변의 충돌한 x 값이다.
    if (circle_y > check_y):
        # 근의 방정식에 의해 충돌된
        find_x = self.solution(-2*circle_x, circle_x**2-100+(rectangle_xy[3]-circle_y)**2,
rectangle_xy[0], rectangle_xy[2])
        and_y = rectangle_xy[3]
    # 여기서는 사각형의 윗변의 충돌한 x 값이다.
    else:
        find_x = self.solution(-2*circle_x, circle_x**2-100+(rectangle_xy[1]-circle_y)**2,
rectangle_xy[0], rectangle_xy[2])
        and_y = rectangle_xy[1]
    # 여기서는 사각형의 왼쪽 변의 충돌한 y 값이다.
    if (circle_x < check_x):
        find_y = self.solution(-2*circle_y, circle_y**2-100+(rectangle_xy[0]-circle_x)**2,
rectangle_xy[1], rectangle_xy[3])
```

이름: 정은성 학과: 원자력 공학과 학번: 2021103751

```
        and_x = rectangle_xy[0]
    # 여기서는 사각형의 오른쪽 변의 충돌한 y 값이다.
    else:
        find_y = self.solution(-2*circle_y, circle_y**2-100+(rectangle_xy[2]-circle_x)**2,
rectangle_xy[1], rectangle_xy[3])
        and_x = rectangle_xy[2]

    # 만약 모서리 충돌이 발생했다면 if 문 아래로 들어간다.
    # 또한 모서리 충돌이 아닌 변의 충돌이면 둘중에 하나가 none 이다.
    if (find_x != None and find_y != None):
        coords = [find_x, and_y, and_x, find_y]
        return coords
    else:
        return NULL
```

<2 sub 근의 공식 방정식 함수>

```
# 근의 공식 값 출력
def solution(self, b, c, range_one, range_two):
    D = (b**2) - (4*c)
    if D>0:
        r1= (-b + (b**2-4*c)**0.5)/(2)
        r2 = (-b - (b**2-4*c)**0.5)/(2)
        # 값이 출동 되는 위치에 있는 지 여부를 확인하고 return 값을 전달해줌
        if (r1 >= range_one and r1 <= range_two):
            return r1
        else:
            return r2
    elif D==0:
        x = -b / 2
        return x
    else:
        pass
```

<3. 반사 백터 구현>

```
# 반사되는 백터를 구하기 위한 함수
def cal_reflection(self, meet):
    # 사각형의 밑변에 부딛친 x 축쪽의 교점의 좌표
    x1 = meet[0]
    y1 = meet[1]
    # 사각형의 옆면에 부딛친 y 축쪽의 교점의 좌표
    x2 = meet[2]
    y2 = meet[3]

    # 백터의 방향 설정
    abs1 = 1
    abs2 = 1
    # 충돌하는 위치에 따라 생성되는 법선 백터의 부호의 값은 일정함.
    if (x1 > x2 and y1 > y2):
        abs1 *= -1
```

이름: 정은성 학과: 원자력 공학과 학번: 2021103751

```
    elif (x1 > x2 and y1 < y2):
        abs1 *= -1
        abs2 *= -1
    elif (x1 < x2 and y1 < y2):
        abs2 *= -1

    # 법선 백터
    # 둘의 거리를 절대값으로 구해주고 판정된 법선 벡터의 부호를 곱해준다.
    N_vector_x = abs1*abs(y1 - y2)
    N_vector_y = abs2*abs(x1 - x2)

    # 법선 백터를 단위 백터로 치환
    new_length = ((N_vector_x)**2 + (N_vector_y)**2)**(0.5)
    N_vector_x = N_vector_x/new_length
    N_vector_y = N_vector_y/new_length

    # 법선 백터가 충돌하기전에 사잇각이 90 도 이하인지 확인하고 이하이면 충돌 발생 X
    if (self.check_90degree(N_vector_x, N_vector_y)):
        # 반사 백터 구하기 : 2 개의 백터(법선 백터와 진입 백터를 더하면 반사백터를 구할 수 있다.)
        vec_X_reflection = self.direction[0] + N_vector_x
        vec_Y_reflection = self.direction[1] + N_vector_y

        # 반사 벡터를 구하고 이를 단위 백터로 구현해준다.
        R_length = ((vec_X_reflection)**2 + (vec_Y_reflection)**2)**(0.5)
        vec_X_reflection = vec_X_reflection/R_length
        vec_Y_reflection = vec_Y_reflection/R_length

        # 반사백터를 이동하는 방향 백터에 넣어준다.
        self.direction[0] = vec_X_reflection
        self.direction[1] = vec_Y_reflection
```

<4. 충돌 벡터 사잇값 90>

```
def check_90degree(self, x, y):
    # 2 개의 백터를 통해서 degree 구하는 공식 통해서 degree 를 구함.
    degree = math.acos(x*self.direction[1]+y*self.direction[0]) * (180/3.14)
    # 만약 90 이상이면, true 로 처리하며 충돌 처리 함.
    if (degree > 90):
        return True
    # 90 미만이면, False 로 처리하며 충돌 처리 안함.
    else:
        return False
```

Game Shot]

2022 년 2 학기 게임 프로그래밍 입문 (실습 2)

이름: 정은성 학과: 원자력 공학과 학번: 2021103751