HW1: drawing some moving image rectangles

20171166 정성환

1. Task Requirements

- Create 101 images using Python class
- The generated image moves around the screen and changes direction when it hits the wall.
- A minimum of 10 random images shall be used.
- The sound shall be reproduced in the event of a collision
- You need to add a player image that can be controlled by a keyboard (use keyboard.py)

2. Code

```
#assets 경로 설정
current_path = os.path.dirname(__file__)
assets_path = os.path.join(current_path, 'assets')
image_path = os.path.join(assets_path, 'sogangimage')
```

-> Added a new image path to separate task-specific reference image files into folders.

```
student_image = pygame.image.load(os.path.join(assets_path, 'student.png'))
```

-> The player's image was set up using the student image.

```
#소리 경로 설정
myBeepSound = pygame.mixer.Sound( os.path.join(assets_path, "20171166_beep.wav" ))
myBeepSound.set_volume(0.01)
```

-> To implement sound when hitting a wall, we added a sound file path to the asset folder and added a volume adjustment function to prevent excessive volume. I set it as small as possible because different volumes may come out depending on the environment.

```
#10가지 이미지 딕셔너리 생성
image = {"1" : "sogang1.png", "2": "sogang2.png", "3" : "sogang3.png", "4" : "sogang4.png", "5":"sogang5.png",
        "6":"sogang6.png",  "7":"sogang7.png",  "8":"sogang8.png",  "9":"sogang9.png", "10":"sogang10.png"}
```

->Python does not have a switch statement. To generate an arbitrary image class, we used a dictionary structure to generate random numbers from 1 to 10 and to output images that fit them. Outputs different images depending on key values from 1 to 10.

```
class sogangImage:
    def __init__(self,):
        self.x = np.random.randint(low=100, high=200)
        self.y = np.random.randint(low=100, high=200)
        self.dx = np.random.randint(-9, 10)
        self.dy = np.random.randint(-10, 11)

        self.time = pygame.time.get_ticks()
        self.period = np.random.uniform(1000, 6000)

        self.imageNum = np.random.randint(low=1, high=10)

        self.image = pygame.image.load(os.path.join(image_path, image[str(self.imageNum)]))

        self.width = self.image.get_width()
        self.height = self.image.get_height()
```

-> The constructor portion of the class. Self, which randomly sets the initial position value and speed and sets the random number of which image to enter.imageNum and self to import appropriate images based on random numbers.Image successfully added.

Use get_width() and get_height() to get and store the height and width of the image.

```
def update(self,):
        update_flag = 0
        if 0:
            current_time = pygame.time.get_ticks()
            if current_time - self.time > self.period: # milisecond
```

```
                self.dx = np.random.randint(-19, 20)
                self.dy = np.random.randint(-20, 21)
                self.time = current_time
                pass

        self.x += self.dx
        self.y += self.dy

        if self.x + self.width > WINDOW_WIDTH:   # right side bounce
            self.dx *= -1
            update_flag = 1

        if self.x < 0: # left side bounce
            self.dx *= -1
            update_flag = 1



        if self.y + self.height > WINDOW_HEIGHT: # bootom side bounce
            self.dy *= -1
            update_flag = 1

        if self.y < 0: # top side bounce
            self.dy *= -1
            update_flag = 1

        return update_flag
```

-> SogangImage.update() is a method of continuously updating the position of an image. At this time, the width and width of the image were used to allow collisions with the wall. And by returning the update_flag, we can return 1 if there is a collision with the wall while moving, or 0 if there is a collision.

```
def draw(self, screen):
        screen.blit(self.image, [self.x, self.y])
```

-> Method of drawing an image on a screen.

```
sogang = sogangImage()
print(sogang)
listOfSogangImages = []
for i in range(101):
    sogang = sogangImage()
    listOfSogangImages.append(sogang)
```

-> Create an instance of the river image class and add to the list.As repeatedly, you can create a list of image instance list of 101.

```
# 게임 반복 구간
while not done:
    # 이벤트 반복 구간
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
        # 키가 눌릴 경우
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                student_dx = -3
            elif event.key == pygame.K_RIGHT:
                student_dx = 3
            elif event.key == pygame.K_UP:
                student_dy = -3
            elif event.key == pygame.K_DOWN:
                student_dy = 3
        # 키가 놓일 경우
        elif event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT or event.key == pygame.K_RIGHT:
                student_dx = 0
            elif event.key == pygame.K_UP or event.key == pygame.K_DOWN:
                student_dy = 0
```

-> The part that controls the player image through the keyboard. Use pygame.event.get(): to activate the for statement while the event is repeated. If the QUIT key is pressed, the event ends and you can change the direction of the player's movement by pressing the arrow key. Make dx, dy zero so that it does not move while the key is not pressed.

```
# 게임 로직 구간
    # 속도에 따라 원형 위치 변경: state update / logic update / parameter update

    student_x += student_dx
    student_y += student_dy

    # ----------------------------------
    # update ball 2
    # your update code here

    for i in range(len(listOfSogangImages)):
        sogang = listOfSogangImages[i]
        soundFlag= sogang.update()
        print(soundFlag)
        if soundFlag == 1:
            myBeepSound.play(0, 1,1)
            myBeepSound.stop
```

-> Repeated interval for updating student image coordinates and instance coordinates in the image list. Since the update() method has been changed to return 1 if it crashes into the wall during the update process, soundFlag receives it and if it is 1, it makes a sound in the if statement.

```
# ----------------------------------
    # 윈도우 화면 채우기
    screen.fill(WHITE)

    # 화면 그리기 구간



    screen.blit(student_image, [student_x, student_y])

    for i in range(len(listOfSogangImages)):
        sogang = listOfSogangImages[i]
        sogang.draw(screen)
```

-> It's a code that paints the screen white and draws images and players. The image is listed through a repeat statement.

3. conclusion

We defined classes, output and moved 101 images to the screen, and performed all the requirements for making sounds when colliding with the wall. The difficult point was whether to include a sound-producing function within the image class. In the end, it was performed externally without doing so, but it is necessary to confirm which structure is more efficient. In addition, there may be overlapping sounds or delays, so correction is required. This is because several images are likely to collide with the wall at the same time.