20192210 김민기

Project : 총알 피하기 UPGRADE

미션 [1]	O	미션 [2]	O
미션 [3]	O	미션 [4]	O
미션 [5]	O	미션 [6]	O
미션 [7]	O	미션 [8]	O
미션 [9]	O	미션 [10]	O
미션 [11]	O		

미션 [1] : 비행기가 총알에 맞았을 때 효과음 발생시키기

```
115     for b in bullets:
116         if collision(player, b) and player.invinciblemode != True:
117             pygame.mixer.Sound('gun.mp3').play()    # (1) 총에 맞았을 때 효과음
118             player.explosion(screen)
```

충돌이 발생하고 무적모드가 아닐때 gun.mp3 파일이 재생되게 만들었다.

미션 [2] : 비행기가 총알에 맞았을 때 터지는 그림 효과 나타내기

```
# player.py
class Player:
    self.explosion_image = pygame.image.load('explosion.png')
    self.explosion_image = pygame.transform.scale(self.explosion_image, (64, 64))
    ...
    def explosion(self, screen):
        explosion_width = self.explosion_image.get_width()
        explosion_height = self.explosion_image.get_height()
        screen.blit(self.explosion_image, (self.pos[0] - explosion_width / 2,
                                           self.pos[1] - explosion_height / 2))
```

충돌이 발생하고 무적모드가 아닐때 explosion.png 이미지가 화면에 나타나게 함수를 만들어주었다.

미션 [3] : 배경그림이 비행기의 움직임에 반응하여 같이 움직이기

player가 움직이는 것 처럼 배경화면도 같이 움직이게 Background 클래스를 따로 구현하여 움직임 처리를 해주었다. Player 클래스와 마찬가지로 update 메소드에서 pos 값을 조정해서 화면 밖으로 나가지 않도록 경계를 설정했다. Game Loop 안에서 player 인스턴스의 움직임을 그대로 따라하였다. 이때 goto 메소드의 인자를 player와 반대로 주었는데, 같은 방향으로 움직여보니 어지러운 느낌이 들어서 반대로 움직이도록 하였다.

```
# background.py
import pygame

class Background:
    def __init__(self, x, y):
```

```
        self.image = pygame.image.load('bg.jpg')
        self.pos = [x, y]  # 가로, 세로
        self.to = [0, 0]

    def draw(self, screen):
        screen.blit(self.image, (self.pos[0], self.pos[1]))

    def goto(self, x, y):
        self.to[0] += x
        self.to[1] += y

    def update(self, dt, screen):
        width, height = screen.get_size()
        self.pos[0] = self.pos[0] + dt * self.to[0]
        self.pos[1] = self.pos[1] + dt * self.to[1]
        self.pos[0] = min(max(self.pos[0], width - self.image.get_width()), 0)
        self.pos[1] = min(max(self.pos[1], height - self.image.get_height()), 0)
```

미션 [4] : 비행기가 총알과 여러번 충돌해야 게임종료

생명력과 무적 개념을 도입했다. 우선 생명력은 메인 파일에서 life 변수를 선언하여 collision이 발생했을때 처리해주었다. 만약 생명력이 0 이하가 되었다면, gameover flag를 True로 설정하여 게임을 종료하였다.

미션 [5] : 무적시간동안 비행기가 반짝거리게 하기

무적은 Player class에 구현했다. invinciblemode flag를 사용하여 무적상태인지 판별하고 be_invincible 메소드로 무적상태를 설정했다. 이때 무적상태시간은 30ms로 설정하고 draw 메소드에서 무적상태시간을 줄이면서 3틱마다 나타나게 했다. 이렇게하면 player가 나타나기와 없어지기를 반복하며 반짝이는 효과를 줄 수 있다. 무적상태시간이 0보다 작게된다면 invinciblemode flag를 False로 설정해주었다. Game Loop에서의 collision시에도 무적상태인지 아닌지를 판별하여 생명력 감소에 조건을 주었다.

```
# Player.py
class Player
    def draw(self, screen):
        ...
        if self.invinciblemode == False:
           screen.blit(rotated, calib_pos)
        else:   # 무적 상태라면
            self.invincibletime -= 1
            if self.invincibletime < 0:
                self.invinciblemode = False
            if self.invincibletime % 3 == 0:    # 3틱 마다 나타나기 --> 깜빡거리기
                screen.blit(rotated, calib_pos)

    def be_invincible(self):
        self.invincibletime = 30
        self.invinciblemode = True
```

미션 [6] : 남은 생명력을 막대기와 숫자로 표시하기

기존의 draw_text를 그대로 사용하여 화면 위에 Time과 Bullets 옆에 남은 생명력을 출력하였다. 이때 막대는 '#'을 연달아 출력하여 막대처럼 보이게 만들었다.

```
# main.py
score = time.time() - start_time
draw_text(f"Time: {time.time() - start_time:.2f}, Bullets: {len(bullets)},
                    Life: {life}  {'#' * life}", 16, (10,10), (255,255,255))
                    # Life 부터 코드 추가함
```

미션 [7] : 총알을 여러 종류로 만들고, 종류별로 크기와 색깔 다르게하기

Bullet class에서 radius와 color 리스트를 추가해주었다. self.radius와 self.color는 리스트에서 랜덤 값을 가지게 만들어서 총알을 여러 종류로 만들었다.

```
# bullet.py
class Bullet:
    def __init__(self, x, y, to_x, to_y):
        color_list = [(255,0,0), (0,255,0), (0,0,254)]  # red, green, blue
        radius_list = [4, 8, 12]
        self.pos = [x, y]
        self.to = [to_x, to_y]
```

```
        self.radius = radius_list[random.randint(0,2)]
        self.color = color_list[random.randint(0,2)]
```

미션 [8] : 총알의 종류마다 차감되는 생명력 다르게 하기

Game Loop에서 collision이 발생했을 때 bullet의 radius에 따라 생명력 감소를 다르게 하였다.

```
# main.py
for b in bullets:
    if collision(player, b) and player.invinciblemode != True:
        ...
        if b.radius == 12:
            life -= 3
        elif b.radius == 8:
            life -= 2
        else:   # b.radius == 4
            life -= 1

        if life <= 0:
            gameover = True
```

미션 [9] : 사용자의 가장 오래 버틴 생존 시간 10개 파일에 기록하기

충돌이 발생하고 생명력이 0보다 작아지면 gameover flag를 True로 설정하여 점수를 score.txt 파일에 쓰는 과정을 진행했다. 파일 입출력에서 모드는 ‘a’로 설정하여 기존 파일에 추가로 입력하게 만들었다. 입력을 마친 후 running flag를 False로 설정하여 GameOver Loop로 이동하게 만들었다.

```
# main.py
if gameover == True:
        f = open('score.txt', 'a')
        msg = f"{score:.3f}\n"
        f.write(msg)
        f.close()
        running = False     # goto GameOver Loop
```

미션 [10] : 게임오버시에, 파일에 기록된 생존시간 화면에 출력

GameOver Loop 전에 파일에서 저장한 score들을 읽어서 score_list 리스트에 저장해주었다. 이때 읽은 score에는 개행문자가 포함되어 있으므로 제거하고 리스트 정렬을 사용하기 위해 float 형으로 저장해 주었다. 읽기만 진행하기 때문에 ‘r’ 옵션을 주었다.

```
# main.py
f = open('score.txt', 'r')
score_list = f.readlines()  # score_list에 score 저장
for i in range(len(score_list)):
    score_list[i] = float(score_list[i].strip()) # 개행문자 제거, float로 저장
score_list.sort(reverse = True)    # 내림차순 정렬
# print(score_list)
f.close()
```

이후에 GameOver Loop에서 score를 출력해 주었다. 이때 엔터나 창 닫기 버튼을 누르면 프로그램이 종료되도록 하였다. sys.exit()이 없으면 pygame.quit()와 pygame.display.update()가 충돌하여 오류가 발생하여서 아예 프로그램을 종료시켰다.

```
# main.py
# GameOver Loop
while running != True:
    screen.fill((0,0,0))    # 화면에 검은색 채우기
    draw_text("GAME OVER", 100, (WIDTH/5, 0), (255,255,255))
    draw_text("PRESS ENTER TO QUIT", 50, (WIDTH/5 + 20, 100), (255,150,255))
    for event in pygame.event.get():
        if event.type == pygame.QUIT:   # 창 닫기 버튼을 눌렀을때
            pygame.quit()
            sys.exit()
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_RETURN:    # 엔터 누르면 게임 종료
                pygame.quit()
                sys.exit()
    for i in range(min(10, len(score_list))):
        if str(score_list[i]) == f"{score:.3f}":
            draw_text(f"{i+1:<3} : {score_list[i]:.3f}", 35,
                      (WIDTH/3 + 40, 150 + 40*i), (0,255,255))
                               # 기록이 10개 안에 있으면 하늘색으로 출력
```

```
        else:
            draw_text(f"{i+1:<3} : {score_list[i]:.3f}", 35,
                      (WIDTH/3 + 40, 150 + 40*i), (255,255,255))
                                    # 기록이 10개 안에 없으면 그냥 출력
    pygame.display.update()
```

미션 [11] : 10개의 기록 중 현재의 기록이 있을 경우 강조하여 표시하기

10개를 출력할때 이번 게임에서 저장한 score와 리스트에 있는 값들을 비교해주어 같으면 하늘색으로 출력하였다. “GAME OVER”, “PRESS ENTER TO QUIT”, 점수 출력 모두 가운데에 출력되도록 위치 정보를 맞춰 draw_text()에 인자로 전달해 주었다.