Pygame으로 게임만들기 3

5. 총알 발사하기

```python
import pygame
import sys
import random
import time
from pygame.locals import *

# 상수영역
# 초당 프레임 수
FPS = 30
# 윈도우 크기, 비율 일정하게 만듦
WINDOWWIDTH = 1080
WINDOWHEIGHT = int(WINDOWWIDTH / 2)
# 배경 최대 크기
ORIGINBACKGROUNDWIDTH = 1280
ORIGINBACKGROUNDHEIGHT = 640
# 스프라이트 속도
BACKGROUNDSPEED = 2
BATSPEED = 7
FIREBALLSPEED = 15
# 박쥐 재시작 시간
BATTIME = 3
# 색
WHITE = (255, 255, 255)
```

```python
class AirplaneBullet(pygame.sprite.Sprite):
    """ ... """
    BULLETSPEED = 15

    def __init__(self, airplane_x, airplane_y):
        """ ... """
        global IMAGESDICT
        pygame.sprite.Sprite.__init__(self)
        self.image = IMAGESDICT["bullet"]
        self.rect = self.image.get_rect()
        self.rect.left = airplane_x + IMAGESDICT["airplane"].get_width()
        self.rect.top = airplane_y + IMAGESDICT["airplane"].get_height() / 2

    def update(self):
        """ ... """
        self.rect = self.rect.move(self.BULLETSPEED, 0)
        if self.rect.left > WINDOWWIDTH:
            self.kill()
```

```python
def init_enemy_pos(image):
    """ ... """
    x = WINDOWWIDTH
    y = random.randrange(0, WINDOWHEIGHT - image.get_height())
    return x, y


def draw_object(image, x, y):
    """ ... """
    global DISPLAYSURF
    DISPLAYSURF.blit(image, (x, y))
```

5. 총알 발사하기

```
83  def main():
84      global FPSCLOCK, DISPLAYSURF
85      global IMAGESDICT
86
87      # 비행기 왼쪽 초기 위치
88      airplane_x = WINDOWWIDTH * 0.05
89      airplane_y = WINDOWHEIGHT * 0.8
90      airplane_y_change = 0
91      airplane_x_change = 0
92
93      # 비행기 크기
94      AIRPLANEWIDTH = IMAGESDICT["airplane"].get_width()
95      AIRPLANEHEIGHT = IMAGESDICT["airplane"].get_height()
96
97      # 윈도우 변경에 따른 배경크기 변경
98      BACKGROUNDWIDTH = IMAGESDICT["background"].get_width()
99
100     # 배경 초기 위치
101     background_x = 0
102     other_background_x = BACKGROUNDWIDTH
103
104     # 박쥐 초기 위치
105     bat_x, bat_y = init_enemy_pos(IMAGESDICT["bat"])
106
107     # 박쥐 초기화 시간변수
108     bat_remove_time = 0
109
110     # 파이어볼 초기화 및 초기 위치
111     # 2/7확률로 fireball이 날아간다.
112     fireball_choice = random.randint(1, 7)
113     if fireball_choice == 1 or fireball_choice == 2:
114         fireball_x, fireball_y = init_enemy_pos(IMAGESDICT["fireball%s" % fireball_choice])
115     else:
116         fireball_x, fireball_y = WINDOWWIDTH, 0
```

5. 총알 발사하기

```
118     # 총알 sprite group
119     bullet_group = pygame.sprite.Group()
120
121     # game loop
122     while True:
123         # event handle
124         for event in pygame.event.get():
125             # 종료
126             if event.type == QUIT or (event.type == KEYUP and event.key == K_ESCAPE):
127                 pygame.quit()
128                 sys.exit()
129             if event.type == KEYDOWN:
130                 if event.key == K_UP:
131                     airplane_y_change = -5
132                 elif event.key == K_DOWN:
133                     airplane_y_change = 5
134                 if event.key == K_RIGHT:
135                     airplane_x_change = 5
136                 elif event.key == K_LEFT:
137                     airplane_x_change = -5
138                 if event.key == K_LCTRL:
139                     # 총알을 추가한다.
140                     bullet_group.add(AirplaneBullet(airplane_x, airplane_y))
141             if event.type == KEYUP:
142                 if event.key == K_UP or event.key == K_DOWN:
143                     airplane_y_change = 0
144                 elif event.key == K_RIGHT or event.key == K_LEFT:
145                     airplane_x_change = 0
146
147         # event에 따른 비행기 위치 변경 및 제한
148         airplane_y += airplane_y_change
149         if airplane_y < 0:
150             airplane_y = 0
151         elif airplane_y > WINDOWHEIGHT - AIRPLANEHEIGHT:
152             airplane_y = WINDOWHEIGHT - AIRPLANEHEIGHT
```

5. 총알 발사하기

```
        # 배경 위치 설정
        background_x -= BACKGROUNDSPEED
        if background_x == -BACKGROUNDWIDTH:
            background_x = BACKGROUNDWIDTH
        draw_object(IMAGESDICT["background"], background_x, 0)

        other_background_x -= BACKGROUNDSPEED
        if other_background_x == -BACKGROUNDWIDTH:
            other_background_x = BACKGROUNDWIDTH
        draw_object(IMAGESDICT["background"], other_background_x, 0)

        # 박쥐 위치 설정
        if BATTIME <= time.time()-bat_remove_time:
            bat_x -= BATSPEED
        if bat_x <= 0:
            bat_remove_time = time.time()
            bat_x, bat_y = init_enemy_pos(IMAGESDICT["bat"])

        # fireball 위치 설정
        if fireball_choice == 1 or fireball_choice == 2:
            fireball_x -= FIREBALLSPEED
        else:
            fireball_x -= 2 * FIREBALLSPEED

        if fireball_x <= 0:
            fireball_choice = random.randint(1, 7)
            if fireball_choice == 1 or fireball_choice == 2:
                fireball_x, fireball_y = init_enemy_pos(IMAGESDICT["fireball%s" % fireball_choice])
            else:
                fireball_x, fireball_y = WINDOWWIDTH, 0
```

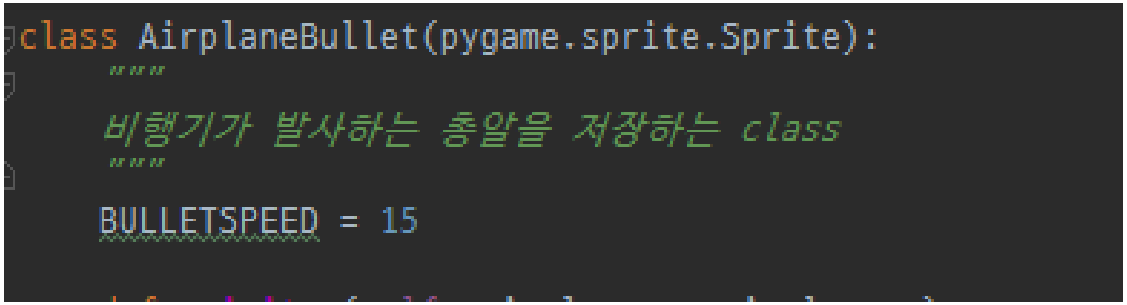
5. 총알 발사하기

```
191         # bullet이 저장된 group에 있는 모든 sprite의 update함수를 실행한다.
192         bullet_group.update()
193
194         # 다른 스프라이트 그리기
195         draw_object(IMAGESDICT["airplane"], airplane_x, airplane_y)
196         draw_object(IMAGESDICT["bat"], bat_x, bat_y)
197         if fireball_choice == 1 or fireball_choice == 2:
198             draw_object(IMAGESDICT["fireball%s" % fireball_choice], fireball_x, fireball_y)
199         bullet_group.draw(DISPLAYSURF)
200
201         pygame.display.update()
202         FPSCLOCK.tick(FPS)
```

5. 총알 발사하기

```
205  def game_init():
206      """
207      게임에 필요한 각종 값을 초기화 한다.
208      :return: None
209      """
210      global FPSCLOCK, DISPLAYSURF
211      global IMAGESDICT
212      FPSCLOCK = pygame.time.Clock()
213      pygame.init()
214
215      # DISPLAY Surface 설정하기
216      DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
217      pygame.display.set_caption('PyFlying')
218
219      # 이미지 받아오기
220      IMAGESDICT = {"airplane": pygame.image.load('images/plane.png'),
221                    "background": pygame.image.load('images/background.png'),
222                    "bat": pygame.image.load('images/bat.png'),
223                    "fireball1": pygame.image.load('images/fireball.png'),
224                    "fireball2": pygame.image.load('images/fireball2.png'),
225                    "bullet": pygame.image.load('images/bullet.png')}
226
227      # 배경 이미지 게임 윈도우 크기에 맞추기
228      assert WINDOWWIDTH <= ORIGINBACKGROUNDWIDTH or WINDOWHEIGHT <= ORIGINBACKGROUNDHEIGHT,\
229          '게임 윈도우 크기가 너무 큽니다.'
230      IMAGESDICT["background"] = pygame.transform.scale(IMAGESDICT["background"], (WINDOWWIDTH, WINDOWHEIGHT))
231      main()
232
233
234  if __name__ == '__main__':
235      game_init()
```

5. 총알 발사하기



- class
  - 무엇인가를 계속해서 만들 수 있는 설계도
  - 이번에는 AirplaneBullet을 통해서 class를 만들었다.

5. 총알 발사하기

```
if event.key == K_LCTRL:
    # 총알을 추가한다.
    bullet_group.add(AirplaneBullet(airplane_x, airplane_y))
```

- instance
  - class를 통해 소프트웨어 내부에 구현된 실체
  - AirplaneBullet(airplane_x, airplane_y)를 통해서 instance를 생성했다.
- 객체
  - 현실의 대상(Object)과 비슷한 상태나 행동을 가진다.
  - instance를 객체라 불러도 틀리지는 않다. (객체에 instance가 포함된다.)

5. 총알 발사하기

- class와 instance의 차이
  - AirplaneBullet class에는 총알의 모양과, 이동속도만이 초기값으로 저장되어 있다.
  - 이 클래스가 가진 생성자를 통해 만든 instance들은 각각의 위치를 저장하고 있다.
  - 따라서 각 만들어진 총알(instance)는 서로 다르게 움직일 수 있게 된다.

5. 총알 발사하기

```
29  class AirplaneBullet(pygame.sprite.Sprite):
30      """
31      비행기가 발사하는 총알을 저장하는 class
32      """
33      BULLETSPEED = 15
34
35      def __init__(self, airplane_x, airplane_y):
36          """
37          생성자, 이미지는 global로 받아옴으로, 받아올 수 없은 airplane의 x, y위치만 받아온다.
38          :param airplane_x: 비행기 x위치
39          :param airplane_y: 비행기 y위치
40          """
41          global IMAGESDICT
42          pygame.sprite.Sprite.__init__(self)
43          self.image = IMAGESDICT["bullet"]
44          self.rect = self.image.get_rect()
45          self.rect.left = airplane_x + IMAGESDICT["airplane"].get_width()
46          self.rect.top = airplane_y + IMAGESDICT["airplane"].get_height() / 2
```

- class AirplaneBullet(pygame.sprite.Sprite)는 pygame.sprite.Sprite를 상속받는 것을 이야기 한다.
- 상속이란, 다른 class의 기능을 물려받는 것을 말한다.

5. 총알 발사하기

- 상속은 기존 클래스를 바꾸지 않고, 기존 클래스를 추가할 때 사용하게 된다.
- 객체지향적 프로그래밍을 할 때, 상속을 사용하게 된다.
- 예를 들어 고양이, 개, 말, 소를 class를 통해 만든다고 하자. 이 때, 네 발 동물이 가지고 있는 특성을 묶어서 네 발 동물이라는 class를 만들고 고양이나 개가 네 발 동물을 상속받게 한다면 유지 보수 측면에서 편해진다.
- ex) 갑자기 꼬리를 흔드는 함수를 넣을 때 4개의 class에 각각 넣는 것은 유지보수상 어렵다.

5. 총알 발사하기

```
35      def __init__(self, airplane_x, airplane_y):
36          """..."""
41          global IMAGESDICT
42          pygame.sprite.Sprite.__init__(self)
43          self.image = IMAGESDICT["bullet"]
44          self.rect = self.image.get_rect()
45          self.rect.left = airplane_x + IMAGESDICT["airplane"].get_width()
46          self.rect.top = airplane_y + IMAGESDICT["airplane"].get_height() / 2
```

- 생성자 부분이다.
- 이는 instance를 만들 때, 필요한 값을 무조건 받도록 만든다.
- pygame.sprite.Sprite.__init__(self)는 상속한 class의 생성자를 받아오는 부분이다.
- Sprite class에도 pygame에서 sprite를 사용하기 위한 필요한 생성자들이 있는데, 그것을 불러온다.

5. 총알 발사하기

```
35      def __init__(self, airplane_x, airplane_y):
36          """..."""
41          global IMAGESDICT
42          pygame.sprite.Sprite.__init__(self)
43          self.image = IMAGESDICT["bullet"]
44          self.rect = self.image.get_rect()
45          self.rect.left = airplane_x + IMAGESDICT["airplane"].get_width()
46          self.rect.top = airplane_y + IMAGESDICT["airplane"].get_height() / 2
```

- self는 객체 자신을 받아오는 부분이다.
  - 만들어진 객체들은 각각 다른 메모리 부분에 저장된다. 따라서 각 객체가 가진 변수를 불러오기 위해선 그 객체가 어느 것인지 알아야 된다. 이를 위해 사용하는 이름이 self이다. class에 있는 메소드의 첫번째 매개변수는 self로 하는 것으로 합의되어 있다.
- sprite는 image를 가지고, 그 둘레를 싸고 있는 사각형으로 위치를 확인한다.

5. 총알 발사하기

```
35      def __init__(self, airplane_x, airplane_y):
36          """..."""
41          global IMAGESDICT
42          pygame.sprite.Sprite.__init__(self)
43          self.image = IMAGESDICT["bullet"]
44          self.rect = self.image.get_rect()
45          self.rect.left = airplane_x + IMAGESDICT["airplane"].get_width()
46          self.rect.top = airplane_y + IMAGESDICT["airplane"].get_height() / 2
```

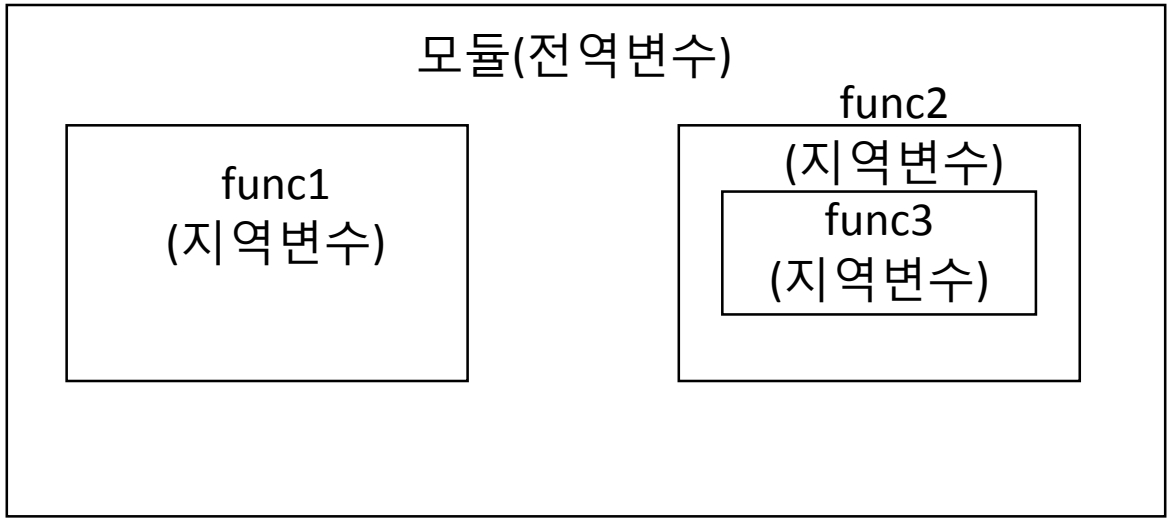
- self.image, self.rect는 만들어진 instance가 가지고 있는 image, rect라는 instance변수이다.
- self.rect = self.image.get_rect()는 surface의 사각형을 만들어준다.
- rect.left와 rect.top은 rect가 가지고 있는 변수로, 왼쪽 x좌표와 위쪽 y좌표를 저장하게 만든다. 이렇게 left나 top을 변경하면 나머지 좌표도 자동적으로 바뀌게 된다.

5. 총알 발사하기

- 네임스페이스(name space, 이름공간)
  - 특정한 변수나 함수를 이름에 따라 구분할 수 있는 범위를 말한다.
  - 프로그래밍언어에서는 이 개념을 통해 하나의 이름이 통용될 수 있는 범위를 제한하여 네임스페이스가 다르면 같은 이름이 다른 개체를 가리키도록 할 수 있다.
  - 파이썬의 네임스페이스는 크게 3가지가 있다.
    - 전역 네임 스페이스: 모듈별로 존재, 모듈 전체에 통용
    - 지역 네임 스페이스 : 함수 및 메소드 별로 존재, 함수 내 지역변수
    - 빌트인 네임 스페이스 : 기본 내장 함수 및 기본 예외들

5. 총알 발사하기

- 변수의 스코프(scope)
  - 이름으로 변수를 찾을 수 있는 영역의 범위
  - 파이썬에서는 지역 변수와 전역 변수만 존재한다.



5. 총알 발사하기

- 쉐도우잉(shadowing; name masking)
  - 특정한 스코프 내에서 선언된 이름이 외부 스코프와 중첩되는 것을 말한다.
  - 지역 name space가 우선적으로 참조되고, 존재할 경우 그 상위의 name space는 참조하지 않는다.
  - 변수명을 참조할 때 거치는 방법
    - 현재 함수의 네임 스페이스에서 지역변수명을 찾는다. 존재하면 이를 사용한다.
    - 찾지 못하면 상위 네임 스페이스로 계속 올라가고, 마지막에는 전역 네임 스페이스를 확인한다.
    - 발견되지 않으면 빌트인(내장) 네임 스페이스를 확인한다.
    - 그래도 없으면 NameError를 발생시킨다.

5. 총알 발사하기

- 쉐도우가 발생했는지 알아차리기 어려운 경우
  - myvalue= 이라고 설정하는 순간에 지역변수처리가 된다.
  - 이럴 경우에는 global문으로 처리해서 접근할 수 있으나 전역변수는 프로그램이 커질 경우 문제가 발생할 수 있다.

```
myvalue = 3

def increase_my_value(step=1):
  myvalue = myvalue + step

increase_my_value()
print(myvalue)
```

5. 총알 발사하기

- 쉐도우가 발생했는지 알아차리기 어려운 경우
  - myvalue= 이라고 설정하는 순간에 지역변수처리가 된다.
  - 이럴 경우에는 global문으로 처리해서 접근할 수 있으나 전역변수는 프로그램이 커질 경우 문제가 발생할 수 있다.

```
myvalue = 3

def increase_my_value(step=1):
  myvalue = myvalue + step

increase_my_value()
print(myvalue)
```

5. 총알 발사하기

```
def sort_priority(values, group):
    found = False
    def helper(x):
        if x in group:
            #print(found)    # found를 출력해보려고 하면 에러 발생
            found = True
            return (0, x)
        return (1, x)
    values.sort(key=helper)
    return found
```

- UnboundLocalError가 발생한다.
  - UnboundLocalError: local variable ‘found’ refenced before assignment

5. 총알 발사하기

- 파이썬은 인터프리터 언어이지만, 실제 실행을 할 때는 컴파일 된 바이트코드로 번역된 상태가 된다.
- 따라서 첫번째 print가 사용될 때는 이미 found라는 값이 지역변수라는 것을 알고 있어 할당되지 않았다는 오류를 출력하는 것이다.
- 이러한 방식은 중간 불필요한 지역변수가 생성되어 상위 scope의 값을 제대로 불러오지 못하는 것을 방지한다.

5. 총알 발사하기

```
def __init__(self, airplane_x, airplane_y):
    """ ... """
    global IMAGESDICT
```

- 처음 global문을 사용하면, 자동적으로 전역변수가 생성된다. 따라서 global문이 없이도 IMAGESDICT을 사용할 수 있다.
- 하지만, 사용한 책에서는 global문을 명시적으로 적어 주었는데, 이는 이 함수에서 사용할 상수를 명시적으로 알려주기 위해서 사용한 것으로 보인다.

5. 총알 발사하기

```
class AirplaneBullet(pygame.sprite.Sprite):
    """ ... """
    BULLETSPEED = 15

    def __init__(self, airplane_x, airplane_y):
        """ ... """
        global IMAGESDICT
        pygame.sprite.Sprite.__init__(self)
        self.image = IMAGESDICT["bullet"]
        self.rect = self.image.get_rect()
```

- instance 변수와 class 변수
  - instance 변수는 속성, 객체변수, 멤버 변수라고도 불린다.
  - 각 객체가 가지고 있는 변수로 서로 공유 되지 않는다.
  - self를 통해 접근할 수 있다.

5. 총알 발사하기

```
class AirplaneBullet(pygame.sprite.Sprite):
    """ ... """
    BULLETSPEED = 15

    def __init__(self, airplane_x, airplane_y):
        """ ... """
        global IMAGESDICT
        pygame.sprite.Sprite.__init__(self)
        self.image = IMAGESDICT["bullet"]
        self.rect = self.image.get_rect()
```

- instance 변수와 class 변수
  - class변수는 모든 instance가 공유할 수 있는 변수.
  - 접근 방식은 class이름.변수명
  - 계좌의 총 수와 같이 instance의 전체에 관련된 것과 같은 곳에서 사용된다.

5. 총알 발사하기

```
class AirplaneBullet(pygame.sprite.Sprite):
    """ ... """
    BULLETSPEED = 15

    def __init__(self, airplane_x, airplane_y):
        """ ... """
        global IMAGESDICT
        pygame.sprite.Sprite.__init__(self)
        self.image = IMAGESDICT["bullet"]
        self.rect = self.image.get_rect()
```

- instance 변수와 class 변수
  - instance의 scope의 상위 scope는 class이기 때문에, instance에 없는 변수는 class부분의 네임 스페이스를 찾게 된다.

5. 총알 발사하기

```
    def update(self):
        """ ... """
        self.rect = self.rect.move(self.BULLETSPEED, 0)
        if self.rect.left > WINDOWWIDTH:
            self.kill()
```

- update는 메서드 오버라이딩(method overridng)을 한 부분이다.
- 메서드 오버라이딩은 부모 클래스 메서드와 같은 이름의 함수를 만드는 것이다.
- 이렇게 오버라이딩을 하고 함수를 호출하면 자식 클래스가 가진 메서드가 실행된다.

5. 총알 발사하기

```
168     def update(self, *args):
169         """method to control sprite behavior
170
171         Sprite.update(*args):
172
173         The default implementation of this method does nothing; it's just a
174         convenient "hook" that you can override. This method is called by
175         Group.update() with whatever arguments you give it.
176
177         There is no need to use this method if not using the convenience
178         method by the same name in the Group class.
179
180         """
181         pass
182
```

- 부모 클래스인 sprit가 가진 update 메서드
- 비어있는 것을 알 수 있다.
- 전체적인 클래스 구조를 짤 때, 미리 만들 필요가 있는 메서드여서 비워두었다.

5. 총알 발사하기

```
# 총알 sprite group
bullet_group = pygame.sprite.Group()
```

```
if event.key == K_LCTRL:
    # 총알을 추가한다.
    bullet_group.add(AirplaneBullet(airplane_x, airplane_y))
```

- sprite가 가진 group class이다.
- 각 sprite를 묶어 group으로 한 번에 사용할 수 있게 만든다.

5. 총알 발사하기

```
# bullet의 저장된 group에 있는 모든 sprite의 update함수를 실행한다.
bullet_group.update()
```

- Group 클래스의 메서드
  - Group.add(*sprites): return None -그룹에 스프라이트를 추가
  - Group.remove(*sprites): return None - 그룹에서 스프라이트를 제거
  - Group.draw(Surface): return None - Surface에 포함된 스프라이트를 그리기
  - Group.update(*args): return None - 그룹 내 모든 스프라이트의 update() 메소드 호출
- update를 통해 내부에 있는 모든 스프라이트의 update를 한다.

6. 적 공격하기

빠진 부분이 꽤 있으니 주의해서 확인

```python
import pygame
import sys
import random
import time
from pygame.locals import *

# 상수영역
# 초당 프레임 수
FPS = 30
# 윈도우 크기, 비율 일정하게 만듦
WINDOWWIDTH = 1080
WINDOWHEIGHT = int(WINDOWWIDTH / 2)
# 배경 최대 크기
ORIGINBACKGROUNDWIDTH = 1280
ORIGINBACKGROUNDHEIGHT = 640
# 스프라이트 속도
BACKGROUNDSPEED = 2
FIREBALLSPEED = 15
# 색
WHITE = (255, 255, 255)


class Boom(pygame.sprite.Sprite):
    """
    박쥐가 죽었을 때, 폭발이미지
    """
    BOOMTIME = 5

    def __init__(self, x, y):
        global IMAGESDICT
        super().__init__()
        self.image = IMAGESDICT["boom"]
        self.rect = self.image.get_rect()
        self.rect.left = x
        self.rect.top = y
        self.time = 0
```

```python
    def update(self):
        self.time += 1
        if self.time >= self.BOOMTIME:
            self.kill()


class BatEnemy(pygame.sprite.Sprite):
    """..."""
    BATSPEED = 7
    BATTIME = 3
    bat_num = 0
    bat_remove_time = 0

    def __init__(self):
        """..."""
        global IMAGESDICT
        super().__init__()
        self.image = IMAGESDICT["bat"]
        self.rect = self.image.get_rect()
        self.rect.left, self.rect.top = init_enemy_pos(IMAGESDICT["bat"])
        BatEnemy.bat_num += 1

    def __del__(self):
        """..."""
        BatEnemy.bat_num -= 1
        BatEnemy.bat_remove_time = time.time()

    def update(self):
        self.rect = self.rect.move(-self.BATSPEED, 0)
        if self.rect.left < 0:
            self.kill()

    def position(self):
        return self.rect.left, self.rect.top
```

6. 적 공격하기

```
89  class AirplaneBullet(pygame.sprite.Sprite):
90      """ ... """
94      BULLETSPEED = 15
95
96      def __init__(self, airplane_x, airplane_y):
97          """ ... """
102         global IMAGESDICT
103         super().__init__()
104         self.image = IMAGESDICT["bullet"]
105         self.rect = self.image.get_rect()
106         self.rect.left = airplane_x + IMAGESDICT["airplane"].get_width()
107         self.rect.top = airplane_y + IMAGESDICT["airplane"].get_height() / 2
108
109     def update(self):
110         """ ... """
116         self.rect = self.rect.move(self.BULLETSPEED, 0)
117         if self.rect.left > WINDOWWIDTH:
118             self.kill()
119
120
121 def init_enemy_pos(image):
122     """ ... """
127     x = WINDOWWIDTH
128     y = random.randrange(0, WINDOWHEIGHT - image.get_height())
129     return x, y
130
131
132 def draw_object(image, x, y):
133     """ ... """
140     global DISPLAYSURF
141     DISPLAYSURF.blit(image, (x, y))
```

6. 적 공격하기

```
144  def main():
145      global FPSCLOCK, DISPLAYSURF
146      global IMAGESDICT
147
148      # 비행기 왼쪽 초기 위치
149      airplane_x = WINDOWWIDTH * 0.05
150      airplane_y = WINDOWHEIGHT * 0.8
151      airplane_y_change = 0
152      airplane_x_change = 0
153
154      # 비행기 크기
155      AIRPLANEWIDTH = IMAGESDICT["airplane"].get_width()
156      AIRPLANEHEIGHT = IMAGESDICT["airplane"].get_height()
157
158      # 윈도우 변경에 따른 배경크기 변경
159      BACKGROUNDWIDTH = IMAGESDICT["background"].get_width()
160
161      # 배경 초기 위치
162      background_x = 0
163      other_background_x = BACKGROUNDWIDTH
164
165      # 파이어볼 초기화 및 초기 위치
166      # 2/7확률로 fireball이 날아간다.
167      fireball_choice = random.randint(1, 7)
168      if fireball_choice == 1 or fireball_choice == 2:
169          fireball_x, fireball_y = init_enemy_pos(IMAGESDICT["fireball%s" % fireball_choice])
170      else:
171          fireball_x, fireball_y = WINDOWWIDTH, 0
172
173      # 총알 sprite group
174      bullet_group = pygame.sprite.Group()
175      bat_group = pygame.sprite.Group()
176      boom_group = pygame.sprite.Group()
177      # 전체 sprite group
178      sprite_group = pygame.sprite.Group()
```

6.

```
    # game loop
    while True:
        # event handle
        for event in pygame.event.get():
            # 종료
            if event.type == QUIT or (event.type == KEYUP and event.key == K_ESCAPE):
                pygame.quit()
                sys.exit()
            if event.type == KEYDOWN:
                if event.key == K_UP:
                    airplane_y_change = -5
                elif event.key == K_DOWN:
                    airplane_y_change = 5
                if event.key == K_RIGHT:
                    airplane_x_change = 5
                elif event.key == K_LEFT:
                    airplane_x_change = -5
                if event.key == K_LCTRL:
                    # 총알을 추가한다.
                    bullet_group.add(AirplaneBullet(airplane_x, airplane_y))
                    sprite_group.add(bullet_group)
            if event.type == KEYUP:
                if event.key == K_UP or event.key == K_DOWN:
                    airplane_y_change = 0
                elif event.key == K_RIGHT or event.key == K_LEFT:
                    airplane_x_change = 0

        # event에 따른 비행기 위치 변경 및 제한
        airplane_y += airplane_y_change
        if airplane_y < 0:
            airplane_y = 0
        elif airplane_y > WINDOWHEIGHT - AIRPLANEHEIGHT:
            airplane_y = WINDOWHEIGHT - AIRPLANEHEIGHT

        airplane_x += airplane_x_change
        if airplane_x < 0:
            airplane_x = 0
        elif airplane_x > WINDOWWIDTH - AIRPLANEWIDTH:
            airplane_x = WINDOWWIDTH - AIRPLANEWIDTH
```

6. 적 공격하기

```
220         # 배경 위치 설정
221         background_x -= BACKGROUNDSPEED
222         if background_x == -BACKGROUNDWIDTH:
223             background_x = BACKGROUNDWIDTH
224         draw_object(IMAGESDICT["background"], background_x, 0)
225 
226         other_background_x -= BACKGROUNDSPEED
227         if other_background_x == -BACKGROUNDWIDTH:
228             other_background_x = BACKGROUNDWIDTH
229         draw_object(IMAGESDICT["background"], other_background_x, 0)
230 
231         # 박쥐가 죽으면 재시작 시간 이후 박쥐를 만든다.
232         if BatEnemy.BATTIME <= time.time() - BatEnemy.bat_remove_time \
233                 and BatEnemy.bat_num <= 0:
234             bat_group.add(BatEnemy())
235             sprite_group.add(bat_group)
236 
237         # fireball 위치 설정
238         if fireball_choice == 1 or fireball_choice == 2:
239             fireball_x -= FIREBALLSPEED
240         else:
241             fireball_x -= 2 * FIREBALLSPEED
242 
243         if fireball_x <= 0:
244             fireball_choice = random.randint(1, 7)
245             if fireball_choice == 1 or fireball_choice == 2:
246                 fireball_x, fireball_y = init_enemy_pos(IMAGESDICT["fireball%s" % fireball_choice])
247             else:
248                 fireball_x, fireball_y = WINDOWWIDTH, 0
249 
250         # bullet과 박쥐를 포함한 모든 sprite의 update함수를 실행한다.
251         sprite_group.update()
252 
```

6. 적 공격하기

```
253         # 충돌을 검사한다.
254         bat_collision_dict = pygame.sprite.groupcollide(bullet_group, bat_group, False, False)
255         if bat_collision_dict:
256             for bullet in bat_collision_dict.keys():
257                 bat_x, bat_y = bat_collision_dict[bullet][0].position()
258                 boom_group.add(Boom(bat_x, bat_y))
259             sprite_group.add(boom_group)
260             pygame.sprite.groupcollide(bullet_group, bat_group, True, True)
261 
262         # 다른 스프라이트 그리기
263         draw_object(IMAGESDICT["airplane"], airplane_x, airplane_y)
264         if fireball_choice == 1 or fireball_choice == 2:
265             draw_object(IMAGESDICT["fireball%s" % fireball_choice], fireball_x, fireball_y)
266         sprite_group.draw(DISPLAYSURF)
267 
268         pygame.display.update()
269         FPSCLOCK.tick(FPS)
```

6. 적 공격하기

```
272  def game_init():
273      """
274      게임에 필요한 각종 값을 초기화 한다.
275      :return: None
276      """
277      global FPSCLOCK, DISPLAYSURF
278      global IMAGESDICT
279      FPSCLOCK = pygame.time.Clock()
280      pygame.init()
281
282      # DISPLAY Surface 설정하기
283      DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
284      pygame.display.set_caption('PyFlying')
285
286      # 이미지 받아오기
287      IMAGESDICT = {"airplane": pygame.image.load('images/plane.png'),
288                    "background": pygame.image.load('images/background.png'),
289                    "bat": pygame.image.load('images/bat.png'),
290                    "fireball1": pygame.image.load('images/fireball.png'),
291                    "fireball2": pygame.image.load('images/fireball2.png'),
292                    "bullet": pygame.image.load('images/bullet.png'),
293                    "boom": pygame.image.load('images/boom.png')}
294
295      # 배경 이미지 게임 윈도우 크기에 맞추기
296      assert WINDOWWIDTH <= ORIGINBACKGROUNDWIDTH or WINDOWHEIGHT <= ORIGINBACKGROUNDHEIGHT,\
297          '게임 윈도우 크기가 너무 큽니다.'
298      IMAGESDICT["background"] = pygame.transform.scale(IMAGESDICT["background"], (WINDOWWIDTH, WINDOWHEIGHT))
299      main()
300
301
302  if __name__ == '__main__':
303      game_init()
304
```

6. 적 공격하기

```
26  class Boom(pygame.sprite.Sprite):
27      """
28      박쥐가 죽었을 때, 폭발이미지
29      """
30      BOOMTIME = 5
31
32      def __init__(self, x, y):
33          global IMAGESDICT
34          super().__init__()
35          self.image = IMAGESDICT["boom"]
36          self.rect = self.image.get_rect()
37          self.rect.left = x
38          self.rect.top = y
39          self.time = 0
40
41      def update(self):
42          self.time += 1
43          if self.time >= self.BOOMTIME:
44              self.kill()
```

- Boom class는 폭발이미지 sprit이다.
- 박쥐의 x와 y값을 받아, 그 부분에 폭발 이미지를 만든다.

6. 적 공격하기

```
    def update(self):
        self.time += 1
        if self.time >= self.BOOMTIME:
            self.kill()
```

- Class 변수인 BOOMTIME의 frame만큼 보이도록 한다.
- update가 한 번 호출될 때마다 time변수를 1씩 올리고, BOOMTIME보다 같거나 커지면 이 sprite를 삭제한다.

6. 적 공격하기

```
47  class BatEnemy(pygame.sprite.Sprite):
48      """..."""
55      BATSPEED = 7
56      BATTIME = 3
57      bat_num = 0
58      bat_remove_time = 0
59
60      def __init__(self):
61          """..."""
64          global IMAGESDICT
65          super().__init__()
66          self.image = IMAGESDICT["bat"]
67          self.rect = self.image.get_rect()
68          self.rect.left, self.rect.top = init_enemy_pos(IMAGESDICT["bat"])
69          BatEnemy.bat_num += 1
70
71      def __del__(self):
72          """..."""
76          BatEnemy.bat_num -= 1
77          BatEnemy.bat_remove_time = time.time()
78
79      def update(self):
80          self.rect = self.rect.move(-self.BATSPEED, 0)
81          if self.rect.left < 0:
82              self.kill()
83
84      def position(self):
85          return self.rect.left, self.rect.top
```

- bat class
  - 박쥐를 만드는 class
- class 변수로는 bat_num과 bat_remove_time을 사용한다.

6. 적 공격하기

```
60      def __init__(self):
61          """ ... """
64          global IMAGESDICT
65          super().__init__()
66          self.image = IMAGESDICT["bat"]
67          self.rect = self.image.get_rect()
68          self.rect.left, self.rect.top = init_enemy_pos(IMAGESDICT["bat"])
69          BatEnemy.bat_num += 1
```

- super().__init__()
  - super()는 자식클래스에서 부모클래스의 내용을 사용하고 싶을 경우, 사용한다.
  - 여기서는
  - super().__init__() == pygame.sprite.Sprite.__init__(self) 와 같다.

6. 적 공격하기

```
60      def __init__(self):
61          """..."""
64          global IMAGESDICT
65          super().__init__()
66          self.image = IMAGESDICT["bat"]
67          self.rect = self.image.get_rect()
68          self.rect.left, self.rect.top = init_enemy_pos(IMAGESDICT["bat"])
69          BatEnemy.bat_num += 1
```

- BatEnemy.bat_num를 통해 class변수에 접근해서 박쥐가 몇 개인지 확인한다.
- 생성할 때, 강제로 1을 올린다.

6. 적 공격하기

```
71      def __del__(self):
72          """..."""
76          BatEnemy.bat_num -= 1
77          BatEnemy.bat_remove_time = time.time()
```

- __del__(self):
  - 소멸자이다.
  - 이 부분은 객체가 소멸될 때, 불러온다.
  - class 변수인 bat_num을 1개 줄이고
  - time.time()을 통해 언제 삭제가 되었는지 저장한다.
- sprite로 사용할 때, 소멸자가 실행이 안될 때가 있다.

6. 적 공격하기

- 예를 들어, a = 생성자()를 통해서 만들고, kill()을 통해 sprite를 삭제하면 a의 refence counter가 남아서 sprite가 소멸되지 않는다.
- 이 때, del(a)를 통해서 소멸자를 부르면서 객체 소멸이 가능하나, 뒤쪽에 이 변수를 사용하면 오류가 발생해서 사용할 수 없다.
- 또한, a = 생성자()부분이 새로 만들어질 때, 소멸되는 것을 볼 수 있다.

6. 적 공격하기

- 따라서 player같이 삭제가 될 필요가 없는 경우가 아니면 group으로 묶어서 사용하는게 좋다.
- 그래서 group()에 바로 집어 넣으면 group에서 삭제되면 자동적으로 삭제된다.
- 또한, kill()을 하면 여러 group에 있어도 전부 삭제해준다.

6. 적 공격하기

```
79      def update(self):
80          self.rect = self.rect.move(-self.BATSPEED, 0)
81          if self.rect.left < 0:
82              self.kill()
83
84      def position(self):
85          return self.rect.left, self.rect.top
```

- update에서 맨 왼쪽으로 가면 죽도록 만들었다.
- position은 죽기 전에 bat의 위치를 반환해서, boom을 만들 위치를 받아오게 만들었다.

6. 적 공격하기

```
172     # 총알 sprite group
173     bullet_group = pygame.sprite.Group()
174     bat_group = pygame.sprite.Group()
175     boom_group = pygame.sprite.Group()
176     # 전체 sprite group
177     sprite_group = pygame.sprite.Group()
```

- sprite는 group으로 묶어서 사용할 수 있다.
- 충돌을 처리하기 위해 각각에 대해서 group을 묶어서 sprite group을 만든다.
- 전체 update를 편하게 하기 위해 sprite 총 group을 만든다.

6. 적 공격하기

- Group에서 다음과 같은 것을 할 수 있다.
  - in 연산자: 원하는 sprite가 내부에 있는지 확인
  - len 함수: sprite가 group내에 몇 개 있는지 확인
  - bool 연산자: sprite가 내부에 하나라도 있는지 확인
  - print(): sprite가 몇 개 포함되었는지 확인
- Group이 가진 함수
  - Group.sprites()는 group이 가진 sprite를 전부 반환해준다.
  - Group.add(sprite)는 group에 sprite를 넣는다.
  - Group.update()는 group내 모든 sprite의 update를 한다.

6. 적 공격하기

- Group이 가진 함수
  - Group.draw(surface)는 group에 담겨있는 모든 sprite를 그린다.
  - Group.empty()는 group내에 있는 모든 sprite를 비운다.

6. 적 공격하기

```
230         # 박쥐가 죽으면 재시작 시간 이후 박쥐를 만든다.
231         if BatEnemy.BATTIME <= time.time() - BatEnemy.bat_remove_time \
232                 and BatEnemy.bat_num <= 0:
233             bat_group.add(BatEnemy())
234             sprite_group.add(bat_group)
```

- BatEnemy가 가진 BATTIME이라는 상수를 통해 언제 부활할지 찾아낸다. 그리고 BatEnemy.bat_num이 <=0이여야 한다.
- bat_group.add()에 바로 생성하여 sprite를 집어넣는다.
- 그 후 sprite_group.add에 bat_group을 넣어 같은 sprite를 넣어준다.

6. 적 공격하기

- 처음에 sprite_group.add(bat_group)을 넣어서 만들면 한 번에 들어가 있을 것으로 예상했는데, add를 새로 해주지 않으면 들어가지 않는 것을 확인했다. 내부 sprite만 넘기는 것으로 예상된다.

6. 적 공격하기

```
# 충돌을 검사한다.
bat_collision_dict = pygame.sprite.groupcollide(bullet_group, bat_group, False, False)
if bat_collision_dict:
    for bullet in bat_collision_dict.keys():
        bat_x, bat_y = bat_collision_dict[bullet][0].position()
        boom_group.add(Boom(bat_x, bat_y))
    sprite_group.add(boom_group)
    pygame.sprite.groupcollide(bullet_group, bat_group, True, True)
```

- 그룹 간 내의 sprite 충돌 감지함수를 통해 충돌을 확인하였다.
- pygame.sprite.groupcollide(group1, group2, dokill1, dokill2): 충돌을 확인할 group 두 개를 넣어준다. dokill부분은 충돌했으면 그 sprite를 삭제할지 확인하는 부분이다. True이면 sprite를 삭제한다.

6. 적 공격하기

```
# 충돌을 검사한다.
bat_collision_dict = pygame.sprite.groupcollide(bullet_group, bat_group, False, False)
if bat_collision_dict:
    for bullet in bat_collision_dict.keys():
        bat_x, bat_y = bat_collision_dict[bullet][0].position()
        boom_group.add(Boom(bat_x, bat_y))
    sprite_group.add(boom_group)
    pygame.sprite.groupcollide(bullet_group, bat_group, True, True)
```

- groupcollide에서는 dictionary로 충돌한 sprite를 반환해준다. 만일, 충돌한게 없을 경우에는 빈 딕셔너리가 되어 if문 내부를 돌지 않게 된다.
- bat_group의 좌표값을 얻기 위해 keys()를 통해 딕셔너리에 접근한다. 그리고 boom을 생성하여 만든다.

6. 적 공격하기

```
# 충돌을 검사한다.
bat_collision_dict = pygame.sprite.groupcollide(bullet_group, bat_group, False, False)
if bat_collision_dict:
    for bullet in bat_collision_dict.keys():
        bat_x, bat_y = bat_collision_dict[bullet][0].position()
        boom_group.add(Boom(bat_x, bat_y))
    sprite_group.add(boom_group)
    pygame.sprite.groupcollide(bullet_group, bat_group, True, True)
```

- boom을 만들었으면, 충돌한 sprite를 제거한다.