

Pygame으로 게임만들기 2

1. 기초 파일 만들기

```
4 import pygame
5 import sys
6 from pygame.locals import *
7
8 # 초당 프레임 수
9 FPS = 30
10 # 윈도우 크기
11 WINDOWWIDTH = 1280
12 WINDOWHEIGHT = 640
13
14 # 윈도우 설정하기
15 DISPLAYSURF = pygame.display.set_mode((400, 300), 0, 32)
16 pygame.display.set_caption('PyFlying')
17
18 WHITE = (255, 255, 255)
19
20
21 def main():
22     global FPSLOCK, DISPLAYSURF
23     pygame.init()
24     FPSLOCK = pygame.time.Clock()
25     DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
26
27     while True: # 게임 루프
28         DISPLAYSURF.fill(WHITE)
29
30         for event in pygame.event.get():
31             if event.type == QUIT:
32                 pygame.quit()
33                 sys.exit()
34         pygame.display.update()
35         FPSLOCK.tick(FPS)
36
37     if __name__ == '__main__':
38         main()
```

<http://pictures-and-images.com/content/buy-6-vector-game.html>
<https://www.pygame.org/docs/ref/key.html>

1. 기초파일 만들기



1. 기초파일 만들기

- from pygame.locals import *
 - KEYUP, QUIT같은 변수를 pygame.locals.QUIT으로 사용하지 않기 위해 사용한다.
- 매직넘버는 피하자.
 - 나중에 윈도우 크기나, FPS를 변경하려 할 때, 일일이 다 고칠 필요가 없다.
 - 매직넘버: 코드에서 의미를 설명할 수 없는 숫자

```
FPS = 30
# 윈도우 크기
WINDOWWIDTH = 1280
WINDOWHEIGHT = 640
```

```
XMARGIN = int((WINDOWWIDTH - (BOARDWIDTH * (BOXSIZE + GAPSIZE))) / 2)
```

But if line 18 didn't use constant variables, it would look like this:

```
XMARGIN = int((640 - (10 * (40 + 10))) / 2)
```

1. 기초파일 만들기

```
global FPSLOCK, DISPLAYSURF
```

- global문과 전역변수
 - global문 뒤에 나오는 변수 이름은 전역변수 있다.
 - 이 함수 밖에서도 값이 유지된다.
 - 어디서든 바뀔 수 있어서 주의한다.
- 1. 함수의 시작부분에 global문으로 변수를 선언하면 전역변수이다.
- 2. 함수 안에서 전역변수와 같은 이름의 변수를 사용하나 값을 할당하지 않으면 전역변수이다.
- 3. 함수 안에서 전역변수와 같은 이름의 변수를 사용하나, 값을 할당하면 지역변수다.
- 4. 함수 안에서 전역변수와 같은 이름의 변수가 없으면 지역변수이다.

1. 기초파일 만들기

- 이벤트 처리
 - 이 for문은 게임 루프에서 발생했던 모든 이벤트를 처리하는 부분이다.
 - 이벤트 처리 루프라고 한다.
 - Pygame.event.get()함수에서 반환한 pygame.Event 객체에 대해 처리한다.
 - 이벤트 객체가 QUIT 이벤트(종료버튼 누름)이거나 ESC에 대한 KEYUP이벤트(키가 떨어지는 이벤트)이면 프로그램을 종료한다.

```
for event in pygame.event.get():  
    if event.type == QUIT or (event.type == KEYUP and event.key == K_ESCAPE):  
        pygame.quit()  
        sys.exit()
```

1. 기초화면 그리기

- 게임 상태를 화면에 그린다.

```
pygame.display.update()  
FPSLOCK.tick(FPS)
```

- 게임 상태가 플레이어의 지시에 따라 필요한 값을 모두 갱신했으면 DISPLAYSURF라는 display surface객체에 그려야 한다. 이를 위해 update()를 사용한다.
- FPS값에 따라서 그려주는 속도를 정해준다.
- 30fps일 경우에는 게임 루프안의 모든 코드를 33.3ms안에 모두 완료해야된다.

1. 기초파일 만들기

- 하지만 요즘 컴퓨터는 너무 빠르기 때문에 시간이 남아 FPSLOCK의 `pygame.Clock` 객체의 `tick()` 메소드를 호출하여 33.3 ms에서 남는 시간동안은 프로그램이 멈춰 있도록 한다.
- 만일 33.3ms보다 길어질 경우 `tick()` 메소드는 기다리지 않는다.

1. 기초파일 만들기

```
37 ▶ if __name__ == '__main__':  
38     main()
```

- 이렇게 main()함수를 사용하지 않았으면 지역변수로 사용할 수 있다.
- 전역변수가 없을수록 디버깅과 유지하기 쉽다.
- 하지만 main()을 사용할 경우, import를 통해 개별 함수를 테스트할 수 있다.
- 만일 main()함수를 쓰지 않고 global scope에 있으면 import하면 자동으로 수행돼서 개별 함수를 호출할 수 없다.

1. 기초함수 만들기

- `__name__`은 python의 빌트인 함수로서, `main()` 함수를 호출했는지 여부를 알 수 있다.
- 파일을 실행했을 경우에는 `__name__`의 값을 `"__main__"`으로 설정한다.
- 만일 `sound.py`를 import했을 경우에는 `__name__`에 `"sound"`가 들어가게 된다.

2. 비행기 움직이기

```
4 import pygame
5 import sys
6 from pygame.locals import *
7
8 # 상수영역
9 # 초당 프레임 수
10 FPS = 30
11 # 윈도우 크기
12 WINDOWWIDTH = 1280
13 WINDOWHEIGHT = 640
14 # 배경 크기
15 BACKGROUNDWIDTH = 1280
16 BACKGROUNDHEIGHT = 640
17 # 색
18 WHITE = (255, 255, 255)
19
20 def draw_object(image, x, y):
21     global DISPLAYSURF
22     DISPLAYSURF.blit(image, (x, y))
23
24 def main():
25     global FPSLOCK, DISPLAYSURF
26     global AIRPLANE
27
28     # 비행기 왼쪽 초기 위치
29     airplane_x = WINDOWWIDTH * 0.05
30     airplane_y = WINDOWHEIGHT * 0.8
31     airplane_y_change = 0
32     airplane_x_change = 0
33
34     # 비행기 크기
35     AIRPLANEWIDTH = AIRPLANE.get_width()
36     AIRPLANEHEIGHT = AIRPLANE.get_height()
37
38     # game loop
39     while True:
40         # event handle
41         for event in pygame.event.get():
42             # 종료
43             if event.type == QUIT or \
44                 (event.type == KEYUP and event.key == K_ESCAPE):
45                 pygame.quit()
46                 sys.exit()
47             if event.type == KEYDOWN:
48                 if event.key == K_UP:
49                     airplane_y_change = -5
50                 elif event.key == K_DOWN:
51                     airplane_y_change = 5
52                 elif event.key == K_RIGHT:
53                     airplane_x_change = 5
54                 elif event.key == K_LEFT:
55                     airplane_x_change = -5
56             if event.type == KEYUP:
57                 if event.key == K_UP or event.key == K_DOWN:
58                     airplane_y_change = 0
59                 elif event.key == K_RIGHT or event.key == K_LEFT:
60                     airplane_x_change = 0
61
62         # event에 따른 비행기 위치 변경 및 제한
63         airplane_y += airplane_y_change
64         if airplane_y < 0:
65             airplane_y = 0
66         elif airplane_y > WINDOWHEIGHT - AIRPLANEHEIGHT:
67             airplane_y = WINDOWHEIGHT - AIRPLANEHEIGHT
68
69         airplane_x += airplane_x_change
70         if airplane_x < 0:
71             airplane_x = 0
72         elif airplane_x > WINDOWWIDTH - AIRPLANEWIDTH:
73             airplane_x = WINDOWWIDTH - AIRPLANEWIDTH
```

2. 비행기 움직이기

```
77 # 배경 그리기
78 DISPLAYSURF.fill(WHITE)
79
80 # 다른 스프라이트 그리기
81 draw_object(AIRPLANE, airplane_x, airplane_y)
82 pygame.display.update()
83 FPSLOCK.tick(FPS)
84
85
86 def game_init():
87     global FPSLOCK, DISPLAYSURF
88     global AIRPLANE
89     FPSLOCK = pygame.time.Clock()
90     pygame.init()
91
92     # DISPLAY Surface 설정하기
93     DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
94     pygame.display.set_caption('PyFlying')
95
96     # 이미지 받아오기
97     AIRPLANE = pygame.image.load('images/plane.png')
98
99     main()
100
101
102 if __name__ == '__main__':
103     game_init()
104
```

2. 비행기 움직이기

```
17 def gameinit():
18     global FPSLOCK, DISPLAYSURF
19     global AIRCRAFT
20     FPSLOCK = pygame.time.Clock()
21
22     pygame.init()
23     # 윈도우 설정하기
24     DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
25     pygame.display.set_caption('PyFlying')
26
27     # 이미지 받아오기
28     AIRCRAFT = pygame.image.load('images/plane.png')
29
30     main()
```

- Global로 aircraft를 가지고 있게 만든다.
- Pygame.image.load()를 통해 비행기 surface객체 AIRCRAFT로 저장한다.

2. 비행기 움직이기

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

- Drawobject는 surface를 받아 displaysurf에 blit하게 만든다.

2. 비행기 움직이기

```
# 비행기 왼쪽 초기 위치  
airplane_x = WINDOWWIDTH * 0.05  
airplane_y = WINDOWHEIGHT * 0.8  
airplane_y_change = 0  
airplane_x_change = 0
```

```
# 비행기 크기  
AIRPLANEWIDTH = AIRPLANE.get_width()  
AIRPLANEHEIGHT = AIRPLANE.get_height()
```

- 윈도우 크기로 비행기의 x와 y좌표의 초기 위치를 준다.
- 또한 변화값을 저장하기 위한 변수를 만든다.
- 비행기 크기는 받아오는 그림에 상관없도록 함수로 처리한다.

2. 비행기 움직이기

```
sys.exit()
if event.type == KEYDOWN:
    if event.key == K_UP:
        airplane_y_change = -5
    elif event.key == K_DOWN:
        airplane_y_change = 5
    elif event.key == K_RIGHT:
        airplane_x_change = 5
    elif event.key == K_LEFT:
        airplane_x_change = -5
```

- 키가 눌렸는데, 방향키 위쪽과 아래쪽이면 y의 변화를 왼쪽과 오른쪽이면 x의 변화를 준다.
- 키가 떨어졌는데, 위와 아래쪽 키면 y의 변화를 0으로, 좌, 우 키면 x의 변화를 0으로 준다.

```
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
```


2. 비행기 움직이기

```
# 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
```

- Y의 변화를 더하여 비행기의 y의 값을 고치는데, 위치가 0보다 작아지면 0으로 고정하고, 윈도우 범위를 넘어가지 않게 height로 조정한다.
- X도 마찬가지로 한다.

2. 비행기 움직이기

```
airplaneY += airplaneY_change
DISPLAYSURF.fill(WHITE)
drawobject(AIRCRAFT, airplaneX, airplaneY)
pygame.display.update()
FPSLOCK.tick(FPS)
```

- Airplane의 Y좌표에 change를 저장하고, displaysurf에 흰색을 채운다.
- drawobject를 통해 airplane surface를 넣는다.

3. 배경 추가

```
4 import pygame
5 import sys
6 from pygame.locals import *
7
8 # 상수 영역
9 # # 설정 프레임 수
10 FPS = 30
11 # 윈도우 크기, 비율 일정하게 만들
12 WINDOWWIDTH = 960
13 WINDOWHEIGHT = int(WINDOWWIDTH / 2)
14 # 배경 최대 크기
15 ORIGINBACKGROUNDWIDTH = 1280
16 ORIGINBACKGROUNDHEIGHT = 640
17 # 배경 움직임 속도
18 BACKGROUNDSPED = 2
19
20 # 색
21 WHITE = (255, 255, 255)
22
23 def draw_object(image, x, y):
24     global DISPLAYSURF
25     DISPLAYSURF.blit(image, (x, y))
26
27 def main():
28     global FPSLOCK, DISPLAYSURF
29     global IMAGESOICT
30
31     # 비행기 왼쪽 초기 위치
32     airplane_x = WINDOWWIDTH * 0.05
33     airplane_y = WINDOWHEIGHT * 0.8
34     airplane_y_change = 0
35     airplane_x_change = 0
36
37     # 비행기 크기
38     AIRPLANEWIDTH = IMAGESOICT["airplane"].get_width()
39     AIRPLANEHEIGHT = IMAGESOICT["airplane"].get_height()
40
41     # 윈도우 변경에 따른 배경크기 변경
42     BACKGROUNDWIDTH = IMAGESOICT["background"].get_width()
43
44     # 배경 초기 위치
45     background_x = 0
46     other_background_x = BACKGROUNDWIDTH
47
48     # game loop
49     while True:
50         # event handle
51         for event in pygame.event.get():
52             # 종료
53             if event.type == QUIT or (event.type == KEYUP and event.key == K_ESCAPE):
54                 pygame.quit()
55                 sys.exit()
56             if event.type == KEYDOWN:
57                 if event.key == K_UP:
58                     airplane_y_change = -5
59                 elif event.key == K_DOWN:
60                     airplane_y_change = 5
61                 elif event.key == K_RIGHT:
62                     airplane_x_change = 5
63                 elif event.key == K_LEFT:
64                     airplane_x_change = -5
65             if event.type == KEYUP:
66                 if event.key == K_UP or event.key == K_DOWN:
67                     airplane_y_change = 0
68                 elif event.key == K_RIGHT or event.key == K_LEFT:
69                     airplane_x_change = 0
```

3. 배경추가

```
72 # events에 따른 비행기 위치 변경 및 제한
73 airplane_y += airplane_y_change
74 if airplane_y < 0:
75     airplane_y = 0
76 elif airplane_y > WINDOWHEIGHT - AIRPLANEHEIGHT:
77     airplane_y = WINDOWHEIGHT - AIRPLANEHEIGHT
78
79 airplane_x += airplane_x_change
80 if airplane_x < 0:
81     airplane_x = 0
82 elif airplane_x > WINDOWWIDTH - AIRPLANEWIDTH:
83     airplane_x = WINDOWWIDTH - AIRPLANEWIDTH
84
85 # 배경 그리기
86 background_x -= BACKGROUNDSPEED
87 if background_x == -BACKGROUNDWIDTH:
88     background_x = BACKGROUNDWIDTH
89 draw_object(IMAGEOBJECT["background"], background_x, 0)
90
91 other_background_x -= BACKGROUNDSPEED
92 if other_background_x == -BACKGROUNDWIDTH:
93     other_background_x = BACKGROUNDWIDTH
94 draw_object(IMAGEOBJECT["background"], other_background_x, 0)
95
96 # 다른 스프라이트 그리기
97 draw_object(IMAGEOBJECT["airplane"], airplane_x, airplane_y)
98 pygame.display.update()
99 FPSLOCK.tick(FPS)
```

3. 배경 추가

```
102 def game_init():
103     global FPSLOCK, DISPLAYSURF
104     global IMAGESDICT
105     FPSLOCK = pygame.time.Clock()
106     pygame.init()
107
108     # DISPLAY Surface 설정하기
109     DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
110     pygame.display.set_caption('PyFlying')
111
112     # 이미지 받아오기
113     IMAGESDICT = {"airplane": pygame.image.load('images/plane.png'),
114                  "background": pygame.image.load('images/background.png')}
115
116     # 배경 이미지 게임 윈도우 크기에 맞추기
117     assert WINDOWWIDTH <= ORIGINBACKGROUNDWIDTH or WINDOWHEIGHT <= ORIGINBACKGROUNDHEIGHT, \
118            '게임 윈도우 크기가 너무 큼니다.'
119     IMAGESDICT["background"] = pygame.transform.scale(IMAGESDICT["background"], (WINDOWWIDTH,
120                                         WINDOWHEIGHT))
121     main()
122
123
124 if __name__ == '__main__':
125     game_init()
126
```

3. 배경 추가

```
# 상수영역
# 초당 프레임 수
FPS = 30
# 윈도우 크기, 비율 일정하게 만들
WINDOWWIDTH = 1080
WINDOWHEIGHT = int(WINDOWWIDTH / 2)
# 배경 최대 크기
ORIGINBACKGROUNDWIDTH = 1280
ORIGINBACKGROUNDHEIGHT = 640
# 배경 움직임 속도
BACKGROUNDSPED = 2
# 색
WHITE = (255, 255, 255)
```

- WINDOWWIDTH와 HEIGHT를 각각 변경하면 그림의 비율이 바뀌기 때문에, 상수를 통해 자동적으로 관리되게 하였다.
- 매직넘버를 최대한 줄인다.
- 상수는 대문자로 쓰기로 약속한다.

3. 배경 추가

```
# 이미지 받아오기
IMAGESDICT = {"airplane": pygame.image.load('images/plane.png'),
              "background": pygame.image.load('images/background.png')}
```

- 스프라이트를 계속 넣을 예정임으로 간단하게 접근하기 위해 dictionary로 접근한다.

3. 배경 추가

```
# 배경 이미지 게임 윈도우 크기에 맞추기
assert WINDOWWIDTH <= ORIGINBACKGROUNDWIDTH or WINDOWHEIGHT <= ORIGINBACKGROUNDHEIGHT,\
    '게임 윈도우 크기가 너무 큼니다.'
IMAGESDICT["background"] = pygame.transform.scale(IMAGESDICT["background"], (WINDOWWIDTH, WINDOWHEIGHT))
```

- 처음에 윈도우 크기를 수정하면, 따라서 배경그림이 변화하도록 만들었다. -윈도우 줄인다.
- Assert문은
assert 조건문, 오류문
 - 조건문이 참이면 통과하고, 거짓이면 프로그램이 오류문을 보여주고 종료된다.
 - 만일 이 부분이 나중에 변경되어, 조건이 바뀌었을 때 오류가 날 가능성이 있는 것을 방지한다.

3. 배경 추가

- Assert문

- Assert문을 통과했다면, 표현식이 반드시 참인 것을 확인할 수 있고, 이를 Sanity check라고 한다.
- 프로그램에서 crash가 발생하는 건 별로 좋지 않지만, 문제가 있으면 빨리 발생하는 것이 좋다.
- 예를 들어, 덮어진 카드들을 열어 같은 두 쌍 씩 맞추는 게임이 있다고 해보자. 그런데 프로그래머가 아무 생각없이 3행 5열로 만들었다면, 다른 곳에서 프로그램 오류가 날 것이다. 하지만 오류가 나는 이유와 상관없이 다른 곳에서 오류가 나게 될 것이다. 이런 것을 방지한다.
- 따라서 crash는 자주 그리고 빨리 발생할수록 좋다.

3. 배경 추가

- 이 부분에서 assert문을 꼭 쓸 필요는 없지만, 너무 윈도우가 커져 배경이 깨지는 것을 막기 위해 넣었다.

3. 배경 추가

- `Pygame.transform.scale(surface, (width, height))`는 받은 `surface`의 크기를 변경한다.
- `Pygame.transform.rotate(surface, angle)`는 `surface`를 counterclockwise로 회전시킨다.
- `Pygame.transform.flip(surface, xbool, ybool)`은 `surface`를 반전시키며, `xbool`에 `True`를 주면 좌우 반전, `ybool`에 `True`를 주면 상하반전이 된다.
- 이 때, 변환된 `surface`는 반환해줌으로 다시 저장해야 된다.

3. 배경 추가

```
# 비행기 크기
AIRPLANEWIDTH = IMAGESDICT["airplane"].get_width()
AIRPLANEHEIGHT = IMAGESDICT["airplane"].get_height()

# 윈도우 변경에 따른 배경크기 변경
BACKGROUNDWIDTH = IMAGESDICT["background"].get_width()
```

- Surface.get_width()는 surface의 너비를 반환한다.
- Surface.get_height()는 surface의 높이를 반환한다.
- 함수를 호출해 항상 사용할 수는 있지만, 너무 길어져서 상수로 저장하였다.

3. 배경 추가



3. 배경 추가

```
# 윈도우 변경에 따른 배경크기 변경  
BACKGROUNDWIDTH = IMAGESDICT["background"].get_width()  
  
# 배경 초기 위치  
background_x = 0  
other_background_x = BACKGROUNDWIDTH
```

- Background_x는 처음 이동하는 background의 좌표, other_background_x는 배경의 끝에 새로 붙이기 위한 x좌표이다.

3. 배경 추가

```
# 배경 그리기
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)
```

- 두 background를 backgroundspeed를 통해 왼쪽으로 움직이게 한다.
- 처음 이동하는 background의 오른쪽이 0에 도달하면 다시 맨 왼쪽으로 민다.

4. 적 추가

주의
""" ... """
부분은 주석을
숨긴 것임으로
입력할 필요가
없다.

```
4 import pygame
5 import sys
6 import random
7 import time
8 from pygame.locals import *
9
10 # 상수영역
11 # 초당 프레임 수
12 FPS = 30
13 # 윈도우 크기, 비율 일정하게 만들
14 WINDOWWIDTH = 1080
15 WINDOWHEIGHT = int(WINDOWWIDTH / 2)
16 # 배경 최대 크기
17 ORIGINBACKGROUNDWIDTH = 1280
18 ORIGINBACKGROUNDHEIGHT = 640
19
20 # 스프라이트 속도
21 BACKGROUNDSPEED = 2
22 BATSPEED = 7
23 FIREBALLSPEED = 15
24 # 박쥐 재시각 시간
25 BATTIME = 3
26 # 색
27 WHITE = (255, 255, 255)
28
29 def init_enemy_pos(image):
30     """ ... """
31     x = WINDOWWIDTH
32     y = random.randrange(0, WINDOWHEIGHT - image.get_height())
33     return x, y
34
35 def draw_object(image, x, y):
36     """ ... """
37     global DISPLAYSURF
```


4. 적 추가

```
52 def main():
53     global FPSLOCK, DISPLAYSURF
54     global IMAGESDICT
55
56     # 비행기 왼쪽 초기 위치
57     airplane_x = WINDOWWIDTH * 0.05
58     airplane_y = WINDOWHEIGHT * 0.8
59     airplane_y_change = 0
60     airplane_x_change = 0
61
62     # 비행기 크기
63     AIRPLANEWIDTH = IMAGESDICT["airplane"].get_width()
64     AIRPLANEHEIGHT = IMAGESDICT["airplane"].get_height()
65
66     # 윈도우 변경에 따른 배경크기 변경
67     BACKGROUNDWIDTH = IMAGESDICT["background"].get_width()
68
69     # 배경 초기 위치
70     background_x = 0
71     other_background_x = BACKGROUNDWIDTH
72
73     # 박쥐 초기 위치
74     bat_x, bat_y = init_enemy_pos(IMAGESDICT["bat"])
75
76     # 박쥐 초기화 시간변수
77     bat_remove_time = 0
78
79     # 파이어볼 초기화 및 초기 위치
80     # 2/7확률로 fireball이 날아간다.
81     fireball_choice = random.randint(1, 7)
82     if fireball_choice == 1 or fireball_choice == 2:
83         fireball_x, fireball_y = init_enemy_pos(IMAGESDICT["fireballs" * fireball_choice])
84     else:
85         fireball_x, fireball_y = WINDOWWIDTH, 0
```

4. 적 추가

```
207 # game loop
208 while True:
209     # event handle
210     for event in pygame.event.get():
211         # 종료
212         if event.type == QUIT or (event.type == KEYUP and event.key == K_ESCAPE):
213             pygame.quit()
214             sys.exit()
215         if event.type == KEYDOWN:
216             if event.key == K_UP:
217                 airplane_y_change = -5
218             elif event.key == K_DOWN:
219                 airplane_y_change = 5
220             elif event.key == K_RIGHT:
221                 airplane_x_change = 5
222             elif event.key == K_LEFT:
223                 airplane_x_change = -5
224         if event.type == KEYUP:
225             if event.key == K_UP or event.key == K_DOWN:
226                 airplane_y_change = 0
227             elif event.key == K_RIGHT or event.key == K_LEFT:
228                 airplane_x_change = 0
229
230     # event에 따른 비행기 위치 변경 및 제한
231     airplane_y += airplane_y_change
232     if airplane_y < 0:
233         airplane_y = 0
234     elif airplane_y > WINDOWHEIGHT - AIRPLANEHEIGHT:
235         airplane_y = WINDOWHEIGHT - AIRPLANEHEIGHT
236
237     airplane_x += airplane_x_change
238     if airplane_x < 0:
239         airplane_x = 0
240     elif airplane_x > WINDOWWIDTH - AIRPLANEWIDTH:
241         airplane_x = WINDOWWIDTH - AIRPLANEWIDTH
```

4. 저 추기

```

123 # 배경 위치 설정
124 background_x -= BACKGROUNDSPEED
125 if background_x == -BACKGROUNDWIDTH:
126     background_x = BACKGROUNDWIDTH
127 draw_object(IMGESDICT["background"], background_x, 0)
128
129 other_background_x -= BACKGROUNDSPEED
130 if other_background_x == -BACKGROUNDWIDTH:
131     other_background_x = BACKGROUNDWIDTH
132 draw_object(IMGESDICT["background"], other_background_x, 0)
133
134 # 박쥐 위치 설정
135 if BATTIME <= time.time()-bat_remove_time:
136     bat_x -= BATSPPEED
137 if bat_x <= 0:
138     bat_remove_time = time.time()
139     bat_x, bat_y = init_enemy_pos(IMGESDICT["bat"])
140
141 # fireball 위치 설정
142 if fireball_choice == 1 or fireball_choice == 2:
143     fireball_x -= FIREBALLSPEED
144 else:
145     fireball_x -= 2 * FIREBALLSPEED
146
147 if fireball_x <= 0:
148     fireball_choice = random.randint(1, 7)
149 if fireball_choice == 1 or fireball_choice == 2:
150     fireball_x, fireball_y = init_enemy_pos(IMGESDICT["fireballs" % fireball_choice])
151 else:
152     fireball_x, fireball_y = WINDOWWIDTH, 0
153
154 # 다른 스크린에 그리기
155 draw_object(IMGESDICT["airplane"], airplane_x, airplane_y)
156 draw_object(IMGESDICT["bat"], bat_x, bat_y)
157 if fireball_choice == 1 or fireball_choice == 2:
158     draw_object(IMGESDICT["fireballs" % fireball_choice], fireball_x, fireball_y)
159 pygame.display.update()
160 FPSLOCK.tick(FPS)

```

4. 적 추가

```
163 def game_init():
164     """게임 초기화"""
165
166     global FPSLOCK, DISPLAYSURF
167     global IMAGESDICT
168     FPSLOCK = pygame.time.Clock()
169     pygame.init()
170
171     # DISPLAY Surface 설정하기
172     DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
173     pygame.display.set_caption('PyFlying')
174
175     # 이미지 받아오기
176     IMAGESDICT = {'airplane': pygame.image.load('images/plane.png'),
177                  'background': pygame.image.load('images/background.png'),
178                  'bat': pygame.image.load('images/bat.png'),
179                  'fireball1': pygame.image.load('images/fireball.png'),
180                  'fireball2': pygame.image.load('images/fireball2.png')}
181
182     # 배경 이미지 게임 윈도우 크기에 맞추기
183     assert WINDOWWIDTH <= ORIGINBACKGROUNDWIDTH or WINDOWHEIGHT <= ORIGINBACKGROUNDHEIGHT, \
184            '게임 윈도우 크기가 너무 큼니다.'
185     IMAGESDICT['background'] = pygame.transform.scale(IMAGESDICT['background'], (WINDOWWIDTH, WINDOWHEIGHT))
186     main()
187
188 if __name__ == '__main__':
189     game_init()
190
```

4. 적 추가

```
29 def init_enemy(image):
30     """
31     """
32     x = WINDOWWIDTH
33     y = random.randrange(0, WINDOWHEIGHT - image.get_height())
34     return x, y
```

- `init_enemy_pos(image)`
 - Surface를 받아 초기 위치를 설정한다.
 - X는 끝으로 잡지만,
 - Y는 랜덤하게 0에서 게임 창 높이에서 surface 높이를 뺀 만큼으로 설정한다.

4. 적 추가

```
79 # 파이어볼 초기화 및 초기 위치
80 # 2/7확률로 fireball이 날아간다.
81 fireball_choice = random.randint(1, 7)
82 if fireball_choice == 1 or fireball_choice == 2:
83     fireball_x, fireball_y = init_enemy_pos(IMAGESDICT["fireball%s" % fireball_choice])
84 else:
85     fireball_x, fireball_y = WINDOWWIDTH, 0
86
```

- Fire ball이 항상 날아오면 어렵기 때문에, 2/7 확률로 fire ball이 날아가도록 한다.
- Fireball_choice가 1, 2면 fireball의 위치를 보낸다.

4. 적 추가

```
134 # 박쥐 위치 설정
135 if BATTIME <= time.time()-bat_remove_time:
136     bat_x -= BATSPEED
137 if bat_x <= 0:
138     bat_remove_time = time.time()
139     bat_x, bat_y = init_enemy_pos(IMGESDICT["bat"])
```

- `time.time()`은 이 함수를 호출한 시간을 초로 반환한다.
- `bat_remove_time`은 처음 박쥐가 죽은 시간을 저장한다.
- `BATTIME`은 맨 위에 설정된 상수로, 박쥐가 바로 나오지 않고 기다리는 시간을 설정한다.

4. 적 추가

```
134 # 박쥐 위치 설정
135 if BATTIME <= time.time()-bat_remove_time:
136     bat_x -= BATSPEED
137 if bat_x <= 0:
138     bat_remove_time = time.time()
139     bat_x, bat_y = init_enemy_pos(IMAGESDICT["bat"])
```

- 박쥐의 위치가 0보다 작아지면, bat_remove_time를 현재 시간으로 수정한다.
- 그리고, BATTIME이 지날 때까지 bat_x의 변화가 중지된다.

4. 적 추가

```
141     # fireball 위치 설정
142     if fireball_choice == 1 or fireball_choice == 2:
143         fireball_x -= FIREBALLSPEED
144     else:
145         fireball_x -= 2 * FIREBALLSPEED
```

- fireball일 경우에는 원래 스피드로 fireball이 날아 오게 한다.
- fireball이 없으면, 시간 지연을 시키기 위해 fireball_x를 이동하게 만든다.

4. 적 추가

```
147 if fireball_x <= 0:
148     fireball_choice = random.randint(1, 7)
149     if fireball_choice == 1 or fireball_choice == 2:
150         fireball_x, fireball_y = init_enemy_pos(IMAGE_DICT["fireball%s" % fireball_choice])
151     else:
152         fireball_x, fireball_y = WINDOWWIDTH, 0
```

- fireball이 끝까지 가면 fireball을 새로 만드는 부분이다.
- IMAGE_DICT["fireball%s" % fireball_choice]
 - 게임을 만들 때, 같은 이름이고 1, 2, 3등으로 이미지가 다를 때, 이러한 식으로 사용할 수 있다.

4. 적 추가

```
system()
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:
```

- 키보드의 event.type에는 두 가지가 있다. 키가 눌리는 keydown, 키가 올라가는 key up
- 그 때, 그 key가 어느 것인지 저장하는 것이 event.key이다.
- 이때 or을 사용하여 두 키를 사용하는 것은 문제가 발생할 수 있어 if로 적절하게 나눠준다.
- 저번에 elif는 K_UP가 눌린 것이 확인되면 뒤쪽의 키를 전부 무시함으로, if로 고쳐주게 된다.