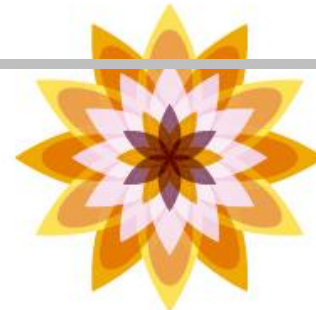
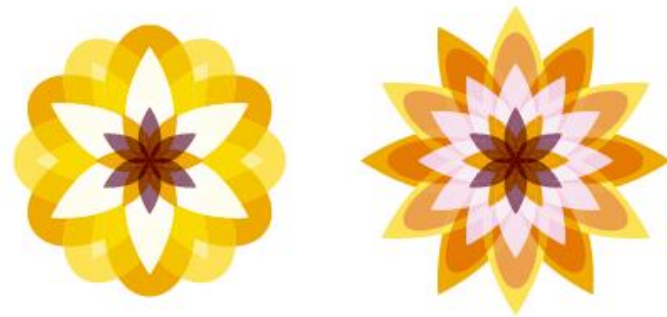


Chapter 13

게임을 바꾸자



1. pong 게임 할 사람?

- pong 게임의 기본 코드입니다. 대부분 앞에서 봤던 것이지요?
- 앞에서 다루지 않은 코드 위주로 설명할게요.
- 그 후 게임을 보다 재미있게 만들 새로운 테크닉을 소개하겠습니다.
- 지금은 위아래로 움직이는 배트와 계속 같은 방향으로 움직이는 공만 있으니까요.
- 충돌 감지도 아직 없지요



pong.py

```
1 import pygame, sys
2 from pygame.locals import *
3 pygame.init()
4 pygame.display.set_caption("Pong")
5 screen = pygame.display.set_mode((1000,600))
6 clock = pygame.time.Clock()
7 ball_image = pygame.image.load("images/ball.png").convert_alpha()
8
```

1. 풍 게임 할 사람?

■ 풍 게임의 기본 코드입니다. 대부분 앞에서 봤던 것이지요?

```
9 class Bat:
10     def __init__(self,ctrls,x):
11         self.ctrls=ctrls
12         self.x=x
13         self.y=260
14
15     def move(self):
16         if pressed_keys[self.ctrls[0]] and self.y > 0:
17             self.y -= 10
18         if pressed_keys[self.ctrls[1]] and self.y < 520:
19             self.y += 10
20     def draw(self):
21         pygame.draw.line(screen,(255,255,255),(self.x,self.y),(self.x,self.y+80),6)
22
23 class Ball:
24     def __init__(self):
25         self.dx=12
26         self.dy=0
27         self.x=475
28         self.y=275
29
```

1. pong 게임 할 사람?

- pong 게임의 기본 코드입니다. 대부분 앞에서 봤던 것이지요?

```
30     def move(self):
31         self.x += self.dx
32         self.y += self.dy
33
34     def draw(self):
35         screen.blit(ball_image,(self.x, self.y))
36
37 ball = Ball()
38 bats = [ Bat( [K_a,K_z], 10), Bat( [K_UP,K_DOWN], 984) ]
39
```

1. pong 게임 할 사람?

- pong 게임의 기본 코드입니다. 대부분 앞에서 봤던 것이지요?

```
40 while 1:
41     clock.tick(30)
42     for event in pygame.event.get():
43         if event.type == QUIT:
44             sys.exit()
45     pressed_keys = pygame.key.get_pressed()
46
47     screen.fill((0,0,0))
48
49     for bat in bats:
50         bat.move()
51         bat.draw()
52
53     ball.move()
54     ball.draw()
55
56     pygame.display.update()
```

2. 투명하게

- 공 이미지를 불러왔습니다.

```
ball_image = pygame.image.load("images/ball.png").convert_alpha()
```

- 자세히 보면 마이크, 파이터, 악당의 이미지를 불러오던 코드와 조금 다를 거예요
- 여기서 사용하는 공 이미지는 배경이 투명하기 때문이지요.
- 이미지 포맷으로 PNG를 사용하면 투명한 배경을 사용할 수 있습니다.
- JPEG로는 불가능하지요.
- 모든 이미지 포맷이 투명한 배경을 지원하지는 않는다는 사실을 기억하세요.
- 일반적인 `convert()` 함수로 이미지를 불러오면, 이미지의 투명한 부분을 이미지의 원래 배경색으로 보여 줍니다.
- 이것을 해결하기 위해 `convert_alpha()` 함수를 사용합니다.

2. 투명하게

- 공 이미지를 불러왔습니다.


```
ball_image = pygame.image.load("images/ball.png").convert_alpha()
```

- 만약 `convert_alpha()` 함수를 사용하면, `set_colorkey()` 함수는 실행 중지됩니다.
- 따라서 투명한 부분이 있는 이미지를 사용할 때만 `convert_alpha()` 함수를 사용하세요
- `set_colorkey()` 함수는 단색을 투명하게 만들고 싶을 때만 사용하세요.
- 우리는 배경에 색이 있는 이미지를 다운받아서 배경을 투명하게 수정했습니다
- 그런데 배경에 빨간색을 넣고, `set_colorkey()` 함수를 사용하여 빨간 색을 투명하게 만들 수도 있습니다.
- 두 가지 방법 모두 가능합니다.

3. 배트 클래스

- 이제 배트(Bat) 클래스를 봅시다.
- 먼저 `__init__()` 함수, `move()` 함수, `draw()` 함수를 만들었습니다.
- 이걸 평소대로지요.

```
class Bat:
    def __init__(self,ctrls,x):
        self.ctrls = ctrls
        self.x = x
        self.y = 260
```




- `__init__()` 함수는 `self` 외 2개의 인자, `ctrls`와 `x`를 갖습니다.
- `ctrls`는 배트를 제어하기 위해 사용하는 키들의 목록입니다.
- `x`는 배트의 x좌표의 초기값을 정합니다.
- 이 인자들은 나중에 배트의 특정 인스턴스에 전달됩니다.
- 클래스 안의 다른 함수들도 이 인자를 사용하려면 이렇게 해야 합니다.

3. 배트 클래스

- 이제 배트(Bat) 클래스를 봅시다.

```
class Bat:
    def __init__(self,ctrls,x):
        self.ctrls = ctrls
        self.x = x
        self.y = 260
```




- `self.y`는 `__init__()` 함수 안에서 정의돼야 하지만, 모든 배트에 대해 같은 값이므로 인자로 쓰지는 않았습니다.
- `__init__()` 함수는 인스턴스가 만들어질 때(이 경우엔 배트 1개) 오직 1번만 실행된다는 것을 잊지 마세요.
- 이 함수는 다른 함수들이 사용할 것들을 준비합니다.

4. 배트의 move() 함수

- 이제 move() 함수를 봅시다.

```
def move(self):  
    if pressed_keys[self.ctrls[0]] and self.y > 0:  
        self.y -= 10  
    if pressed_keys[self.ctrls[1]] and self.y < 520:  
        self.y += 10
```



- 많이 보던 코드가 있지 않나요?


```
if pressed_keys[K_LEFT] and self.x > 0:  
    self.x -= 3
```

- 눌린 키들 리스트(pressed_keys)의 대괄호 안에 들어 있는 것만 다르네요.
- 눌린 키들 리스트는 모든 눌린 키의 리스트입니다.
- 이 리스트는 다른 함수에도 사용되기 때문에, move() 함수가 아니라 게임 루프 안에서 만들었습니다.

4. 배트의 move() 함수

- 이제 move() 함수를 봅시다.

```
def move(self):  
    if pressed_keys[self.ctrls[0]] and self.y > 0:  
        self.y -= 10  
    if pressed_keys[self.ctrls[1]] and self.y < 520:  
        self.y += 10
```




- 앞에서는 왼쪽 화살표(K_LEFT)가 눌렸는지 확인했습니다.
- 여기 있는 새 move() 함수에서는 self.ctrls[0]가 눌렸는 지를 물어봅니다.
- self.ctrls[0]는 무엇일까요?
- __init__() 함수에서, self.ctrls = ctrls이었습니니다.
- 한마디로, 하나의 특정 배트가 만들어질 때 그 배트에 주어진 ctrls입니다.
- 어떤 것이 ctrls로 주어지는지는 조금 뒤에서 설명하겠습니다.

4. 배트의 move() 함수

- 이제 move() 함수를 봅시다.

```
def move(self):  
    if pressed_keys[self.ctrls[0]] and self.y > 0:  
        self.y -= 10  
    if pressed_keys[self.ctrls[1]] and self.y < 520:  
        self.y += 10
```




- 눌린 키들 리스트는 move() 함수 밖에서 만들어지지만 move() 함수가 사용해야 하는 변수입니다
- 앞에서 파이썬은 이런 식으로 작동하지 않는다고 말했습니다.
- 파이썬이 어떤 변수를 찾으려 함수 밖으로 나가게 하려면, 그 변수는 전역(global) 변수라고 말해 줘야 합니다
- 그런데 우리는 global 이라고 써주지 않았지요
- 파이썬은 어떻게 눌린 키들이 전역 변수라는 것을 알고 있을까요?

4. 배트의 move() 함수

- 이제 move() 함수를 봅시다.

```
def move(self):  
    if pressed_keys[self.ctrls[0]] and self.y > 0:  
        self.y -= 10  
    if pressed_keys[self.ctrls[1]] and self.y < 520:  
        self.y += 10
```



- 이 변수는 값이 주어지기 전에 사용되기 때문입니다 .
- 스크린이나 이미지 변수 및 기타 변수에 대해서도 마찬가지입니다.
- 지금은 무슨 말인지 몰라도 괜찮습니다.
- 나중에 다시 읽어 보면 이해가 될 거예요.

5. 배트를 위로 움직이기

- 다음 코드로 배트를 만들었습니다.
- 보통 리스트를 놓는 곳, 그러니까 클래스 바로 밑에 썼지요.

```
bats = [ Bat( [K_a,K_z], 10), Bat( [K_UP,K_DOWN], 984) ]
```

- 지금까지는 리스트를 만들고, 프로그램 다른 데서 새로운 객체를 만든 다음, 파이썬에게 리스트 추가를 시켰습니다.
- 반면 여기에서는 배트들 bats이라는 리스트를 만들고, 그 안에서 배트 2개를 직접 만들었습니다.
- 위 코드를 간단히 하면 bats = [Bat(), Bat()] 이지요.
- 배트 2개가 들어 있는 리스트를 만드는 것입니다.
- 파이썬은 미사일에서 그랬던 것처럼 각각의 배트를 리스트 순서로 구분합니다.
- 따라서 배트들의 이름이 같아도 상관없습니다.
 - 실제로 여기 있는 건 두 배트의 생성자들이지만, 지금은 배트라고 생각해도 괜찮습니다

5. 배트를 위로 움직이기

- 배트 옆 소괄호 안에는 `__init__()` 함수에서 요구하는 인자 2개를 씁니다.
- `ctrls`와 `x`이지요.
- 첫 번째 인자인 `ctrls`는 키(key)들이 들어 있는 리스트입니다.
- 이 리스트에는 항목이 2개밖에 없습니다.
- `A`와 `Z` 또는 위쪽 화살표 키와 아래쪽 화살표 키 말입니다.
- `x`는 배트의 `x`좌표를 나타내는 숫자입니다.
- `move()` 함수를 다시 봅시다.

```
def move(self):  
    if pressed_keys[self.ctrls[0]] and self.y > 0:  
        self.y -= 10
```

5. 배트를 위로 움직이기

- `move()` 함수를 다시 봅시다.

```
def move(self):  
    if pressed_keys[self.ctrls[0]] and self.y > 0:  
        self.y -= 10
```

- `if pressed_keys[self.ctrls[0]]`는 눌린 키들 리스트 `pressed_keys`에 `self.ctrls[0]`가 있는지 물어보는 코드입니다.
- `self.ctrls[0]`는 `self.ctrls[]` 리스트의 첫 번째 항목이지요.
- 리스트의 첫 번째 항목 번호는 항상 0이니깐요.
- 첫 번째 배트에 대한 눌린 키들 리스트의 첫 번째 항목은 `K_a`입니다.
- `A`가 눌린 키들 리스트 안에 있는지 물어보는 것이랍니다.
- 게임 플레이어가 `A`를 눌렀냐고 말이에요.
- `if pressed_keys[K_a]` 라고 바꿔 써도 됩니다.
- 또 `self.y > 0`(배트의 y좌표가 0보다 큼)이면 `self.y -= 10`(배트의 y좌표를 10만큼 줄임)이지요. 배트를 위로 움직이는 거예요.

6. 배트를 아래로 움직이기

- `move()` 함수의 3번째, 4번째 줄을 봅시다.

```
if pressed_keys[self.ctrls[1]] and self.y < 520:  
    self.y += 10
```

- 이번에는 배트를 아래로 움직입니다.
- `if pressed_keys[self.ctrls[1]]` 는 `self.ctrls` 리스트의 두 번째 항목(`self.ctrls[1]`)을 찾습니다.
- 이 경우엔 `z` 겠지요.
- `z` 가 눌렸고 `self.y < 520`(배트의 y좌표가 520보다 작음)이라면 `self.y += 10`(배트의 y좌표를 10만큼 증가)가 되겠지요.
- 배트를 밑으로 이동시킨다는 뜻입니다.
- `self.y < 520`라는 조건이 있으니까 배트가 스크린 바닥보다 위에 있을 때만 움직이겠지요.

6. 배트를 아래로 움직이기

- `move()` 함수의 3번째, 4번째 줄을 봅시다.

```
if pressed_keys[self.ctrls[1]] and self.y < 520:  
    self.y += 10
```

- 왜 520 일까요?
- `draw()` 함수에서 배트의 세로 길이가 80픽셀임을 알 수 있었습니다
- 스크린의 높이는 600픽셀이므로, 배트의 y좌표가 520보다 크면 배트의 일부분은 스크린 밖으로 빠져나가 있는 상태이므로 더 내려가게 하면 안됩니다

6. 배트를 아래로 움직이기

- `move()` 함수의 3번째, 4번째 줄을 봅시다.

```
if pressed_keys[self.ctrls[1]] and self.y < 520:  
    self.y += 10
```

- 배트 1개를 위아래로 움직이는 것치고는 조금 장황한 것도 같지만, 일단 클래스를 만들어 놓으면 인스턴스를 추가하기 쉬우니 첫 번째 배트를 만들어 놓고 (`Bat([K_UP, K_DOWN], 984)`), 두 번째 배트를 추가하는 것입니다.
- 만약 3명이서 플레이한다면 세 번째 배트를 쉽게 추가할 수 있습니다
- `Bat([K_j, K_n], 530)`이라고 쓰면 스크린 한가운데에 배트 1개가 추가됩니다
- 이 배트를 위아래로 움직이려면, **J**, **N**를 누르도록 설정하면 되겠지요



Thank You
