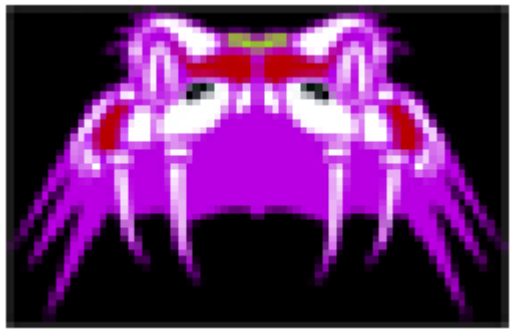
Chapter 07

악당과 마주칠 때

1. 악당 클래스

- 구름을 악당(bad guy)으로 바꿔 봅시다. 악당을 만드는 방식도 구름과 같습니다.
  - 이미지를 가져와 악당 객체를 만듭니다.
  - 악당을 여럿 만들고 각각 제어하기 위해 클래스를 만듭니다.
  - 악당의 움직임을 기록할 수 있도록 리스트를 만듭니다.
  - 게임 루프에서 악당과 관련된 함수를 불러옵니다.
  - 악당들을 괴롭힙니다
- 먼저 악당을 그리거나 웹사이트에서 이미지를 다운받읍시다.
- 참고로, 실제 이미지는 컬러이며 파일 이름은 badguy.png입니다.
- 악당 이미지는 가로가 70픽셀, 세로가 45픽셀이고 검은색 배경입니다.
- 직접 그렸을 경우에는 확장자가 png인 파일로 저장해야 합니다.



1. 악당 클래스

- 다음의 코드를 입력한 다음, 코드를 읽고 이해해 보세요.
- 거의 다 앞에서 다룬 것들입니다.
- 이 프로그램은 65쪽에서 배운 원래 빗방울과 비슷합니다.
- 다른 점은 선을 그리지 않고 이미지를 표시한다는 것입니다.
- 악당 인스턴스를 하나만 만들었기 때문에 리스트는 안 만들어도 됩니다.
- 또, 아직은 for 루프나 while 루프에서 함수를 불러오지 않아도 됩니다.

1. 악당 클래스

badguy.py

```
1  import pygame, sys, random
2  from pygame.locals import *
3  pygame.init()
4  clock = pygame.time.Clock()
5  pygame.display.set_caption("Space Invaders")
6  screen = pygame.display.set_mode((640,650))
7
8  badguy_image = pygame.image.load("images/badguy.png").convert()
9
10 class Badguy:
11     def __init__(self):
12         self.x = random.randint(0,570)
13         self.y = -100
14     def move(self):
15         self.y += 5
16     def draw(self):
17         screen.blit(badguy_image,(self.x,self.y))
18
19 badguy = Badguy()
```

1. 악당 클래스


```
20
21  while 1:
22      clock.tick(60)
23      for event in pygame.event.get():
24          if event.type == QUIT:
25              sys.exit()
26      screen.fill((0,0,0))
27
28      badguy.move()
29      badguy.draw()
30      pygame.display.update()
```

실행

1. 악당 클래스

- 실행시키면 악당 한 마리가 스크린 위쪽에서 내려옵니다.
- 실행시킬 때마다 내려오기 시작하는 지점이 달라집니다.
- x좌표를 x축 위, 0과 570 사이의 임의의 지점으로 정했으니까요.
- move() 함수가 불려올 때마다 self.y의 값이 증가됩니다.
- 매 게임 루프마다 1번씩 증가하지요
- 따라서 악당이 스크린에서 지퍼를 내리는 것처럼 내려옵니다.
- 너무 느리다고요?
- 그래도 지금은 조금씩 움직이도록 만들어야 합니다.
- 하나씩 살펴본 후 고쳐서 더 재미있게 만들어 봅시다!

2. 대각선으로 내려오는 악당

- 악당(Badguy) 클래스의 move()함수를 이렇게 바꿔 봅시다.

```
def move(self):
    self.y += 5
    if self.y > 300:
        self.x += 5
```

- 이렇게 하면 매 게임 루프당 self.y가 5씩 증가합니다.
- 만약 self.y가 300보다 크면, self. x도 5씩 증가합니다.

2. 대각선으로 내려오는 악당

- 이렇게만 해도 되지만 다음 코드도 실험해 봅시다.

```
def move(self):
    self.y += 5
    if self.y > 300 150 and self.y < 250:
        self.x += 5
    if self.y > 250:
        self.x -= 5
```

- **move() 함수는 self.y에 5픽셀을 더합니다.**
- **그러면 악당은 아래쪽으로 움직입니다.**
- **만약 self.y가 150보다 크고 250보다 작으면, self.x에 5픽셀을 더합니다.**
- **이렇게 하면 악당이 오른쪽으로 움직이게 됩니다.**
- **그러나 self.y가 250보다 크면, self.x에서 5를 뺍니다.**
- **그러면 악당은 왼쪽으로 움직입니다.**

2. 대각선으로 내려오는 악당

- __init__() 함수 안의, self. x를 임의의 숫자로 지정하는 코드는 주석 처리하고 self.x를 285로 고정하고 실험해 보는 것이 좋을 것 같습니다. 아래처럼요.

```
def __init__(self):
    #self.x = random.randint(0, 570)
    self.x = 285
    self.y = -100
```

- 이렇게 하면 악당은 항상 스크린 한가운데에서 아래로 내려갑니다.

3. dy로 가속시키기

- 원래의 __init__(), move() 함수로 돌아가 봅시다.

```
def __init__(self):
    #self.x = random.randint(0, 570)
    self.x = 285
    self.y = -100

def move(self):
    self.y += 5
    if self.y > 150 and self.y < 250:
        self.x += 5
    if self.y > 250:
        self.x -= 5
```

3. dy로 가속시키기

- move() 함수에서는 매 루프마다 self.y의 값을 얼마나 바꿀지 정합니다.
- 여기서는 5만큼 더하도록 했지요.
- 수학적으로 y값의 바뀌는 양은 dy, x값이 바뀌는 양은 dx라고 합니다.
- dx는 x축 방향으로의, dy는 y축 방향으로의 속도를 나타냅니다.
- 만약 dy가 0 이면, 세로로는 움직이지 않습니다. y값이 바뀌지 않으니까요.
- dy를 사용해 move() 함수를 다시 써 봅시다.

```
def move(self):
    dy = 5
    self.y += 5 dy
```

3. dy로 가속시키기

- 코드와 변수를 하나씩 추가했을 뿐 같은 내용의 코드입니다.
- 그런데 매 게임 루프마다 dy를 1씩 증가시킨다고 생각해 봅시다.
- 루프를 2번 돌면 2픽셀만큼 커질 것입니다. 5번 돌면 5픽셀만큼 커지겠지요.
- 10번 돌면 10픽셀만큼 커질 거고요.
- 악당은 점점 빠르게 스크린 아래로 질주할 것입니다.
- 이런 걸 가속이라고 합니다. 이렇게 만들려면 매 루프마다 dy를 바꿔야 합니다.

```
def move(self):
    dy = 5
    dy += 1
    self.y += dy
```

3. dy로 가속시키기

- 이제 move() 함수가 불려올 때 (매 게임 루프)마다 dy의 값은 1씩 커집니다.
- 그런데 이번에도 문제가 발생했습니다.
- 변수 dy에 값을 주지 않은 채 1을 더하려 했기 때문입니다.
- 변수를 만들 때는 값부터 정해야 합니다.
- 안 그러면 기분이 나빠진 파이썬이 “dy? 이거 뭐야?"라고 말하며 충돌합니다.
- dy의 값으로 0을 줘 봅시다

```
def move(self):
    dy = 0
    dy += 1
    self.y += dy
```

3. dy로 가속시키기

- 제대로 변수를 만들고 값도 주었습니다. 파이썬도 별말 없겠지요.
- 하지만 move() 함수가 불려올 때마다 dy는 0으로 리셋되고 1만큼 증가합니다.
- self.y는 dy만큼 증가하니까, 1만큼만 증가합니다.
- 그러니 __init__() 함수 안에 dy를 만들어 봅시다.

```
def __init__(self):
    self.x = random.randint(0,570)
    self.y = -100
    dy = 0

def move(self):
    dy = 0
    dy += 1
    self.y += dy
```

3. dy로 가속시키기

- __init__() 함수는 악당이 처음 만들어질 때만 불려오므로 dy의 값은 매 게임 루프마다 0으로 돌아가지 않습니다.
- 좋은 해결책 같아 보이지요.
- 그런데 함수 안에 변수를 만들면, 그 변수는 해당 함수 안에서만 사용됩니다.
- move() 함수가 불려오면, 파이썬은 move() 함수 안에 dy라는 변수가 있는지 찾습니다.
- 하지만 dy는 move() 함수 안에 없으니까 파이썬은 dy를 발견하지 못하고 불평하겠지요.

3. dy로 가속시키기

- 변수 dy를 프로그램의 셋업에서 만들 수도 있습니다.
- 스크린과 이미지들을 정의하는 곳에서요.
- 하지만 이것도 잘 안 될 겁니다.
- 각각의 악당은 자신만의 dy를 가져야 하거든요.
- 안 그러면 모든 악당이 같은 속도로 내려오게 되니까요.
- 속도도 매우 급격하게 빨라지겠죠.
- 파이씬은 move() 함수안만 살펴보기 때문에 dy를 못 찾고 있는 데도요.

3. dy로 가속시키기

- 이렇게 해야 합니다.

```
def __init__(self):
    self.x = random.randint(0,570)
    self.y = -100
    self.dy = 0

def move(self):
    self.dy += 1
    self.y += self.dy
```

- self.를 dy 앞에 써서 파이썬이 move() 함수 안뿐만 아니라 클래스 전체를 살펴보도록 만듭니다.

3. dy로 가속시키기

- 이렇게 하면 악당을 여러 마리 만들었을 때, 각각의 악당마다 변수 dy가 만들어집니다.
- dy는 매 게임 루프마다 특정 값(이 경우에는 0)에서 시작하여 move() 함수에 의해 1씩 증가합니다.
- 이렇게 쓰면, 악당 한 마리가 스크린 아래로 가속하며 떨어지는 것을 볼 수 있습니다. 상당히 빠른 속도입니다.
- 그러면 self.dy += 1에서 1을 0.1 같은 수로 바꾸어 속도를 늦춰도 됩니다.
- 이제 객체를 가속시키는 방법을 알겠지요? 이건 게임에 응용된 물리학입니다.
- 또 다른 쉬운 예로 중력이 있지요.
- 이제 점점 빨라지는 악당들을 게임 속에 등장시킬 수 있겠네요!

4. 벽에 부딪히면 튕겨 나오기

- 악당들을 쏴서 날려 버리기 전에 스크린 안에 가둬야 합니다.
- 벽에 부딪히면 안으로 튕겨 나오게요. 안 그러면 스크린 밖으로 날아가 버리니까요.
- 악당들을 스크린 안에 가두려면 self.dy라는 변수를 만든 것처럼 self.dx라는 변수를 만들어야 합니다.
- 코드를 좀 정리해 볼까요?

```
def __init__(self):
    self.x = random.randint(0,570)
    self.y = -100
    self.dy = 0
    self.dx = 3

def move(self):
    self.x += self.dx
    self.dy += 1 0.1
    self.y += self.dy
```

4. 벽에 부딪히면 튕겨 나오기

- self.dx는 __init__() 함수 안에 만들고, 값은 3으로 정했습니다.
- move() 함수를 보면 self. x가 매 게임 루프마다 self.dx만큼 바뀝니다.
- 악당들이 만들어지면 매 게임 루프마다 오른쪽으로 3픽셀씩 움직일 것이라는 뜻입니다.
- 매 게임 루프마다 1픽셀씩 아래로 내려간다는 것도 알 수 있습니다.
- 매 게임 루프마다 self.y에 self.dy를 더하는데 , self.dy는 1로 __init__() 함수에서 정했으니까요.

4. 벽에 부딪히면 튕겨 나오기

- move() 함수에 아래 두 줄을 추가합니다.

```
def move(self):
    if self.x < 0 or self.x > 570:
        self.dx *= -1
    self.x += self.dx
    self.dy += 0.1
    self.y += self.dy
```

4. 벽에 부딪히면 튕겨 나오기

- 벽에 부딪힌 악당들이 밖으로 날아가지 못하고 안으로 튕기게 하려면, 그러니까 x 방향을 반대로 바꾸려면 어떻게 해야 할까요?
- 왼쪽 벽에 닿을 때는 self.x가 0이고, 오른쪽 벽에 닿을 때는 self.x가 570입니다.
- 따라서 if self.x<0 or self.x>570 :라고 써서 악당이 스크린의 왼쪽이나 오른쪽 벽에 닿았는지를 확인합니다.
- self.dx가 0보다 작거나 570보다 크면, 반대 방향으로 가게 하기 위해 self.dx에 1을 곱합니다. 간단하지요.

4. 벽에 부딪히면 튕겨 나오기

- 위에서 새로 추가한 두 줄을 move()함수에 넣지 말고, 새로운 함수를 만들어 거기에 넣읍시다.
- 새 함수의 이름은 bounce() 라고 합시다.
- bounce() 함수의 위치는 move() 함수 밑이든 위든 상관없습니다.

```
def move(self):
    if self.x < 0 or self.x > 570:
        self.dx *= -1
    self.x += self.dx
    self.dy += 0.1
    self.y += self.dy

def bounce(self):
    if self.x < 0 or self.x > 570:
        self.dx *= -1
```

4. 벽에 부딪히면 튕겨 나오기

- 왜 이렇게 하는 걸까요? 반드시 이렇게 해야만 하는 것일까요?
- 솔직히 반드시 이래야만 하는 것은 아닙니다.
- 하지만 코드 한 뭉치를 별도로 분리해 놓으면 전체 코드에서 찾기도, 이해하기도 쉽습니다. 한마디로 깔끔해지죠.
- 덧붙여, 만약 이렇게 했다면 게임 루프에서 bounce() 함수를 불러와야 합니다.
- move() 함수와 draw() 함수를 불러왔던 것과 같은 방법으로요.

```
badguy.move()
badguy.bounce()
badguy.draw()
```

5. 바닥으로 떨어지는 악당들

- 함수를 만든 김에, 스크린 바닥에 떨어진 악당을 감지할 장치도 만들어 봅시다.
- 이 함수는 악당 클래스 안에 씁니다.
- 악당 클래스 안 어디에 만들어도 상관없습니다.
- 빗방울에서 만들었던 것과 거의 비슷합니다.

```
def off_screen(self):
    return self.y > 640
```

5. 바닥으로 떨어지는 악당들

- 악당이 스크린 바닥에 떨어지면 새 악당을 만들어야 합니다.
- 아래 코드는 while 루프 안, 악당 관련 함수들을 불러오는 부분 밑에 씁니다.

```
while 1:
(중략)
    badguy.move()
    badguy.bounce()
    badguy.draw()

    if badguy.off_screen():
        badguy = Badguy()
```

5. 바닥으로 떨어지는 악당들

- badguy는 우리가 만든 악당입니다.
- 악당들에 대해 off_screen() 함수를 불러와서, 만약 off_screen() 함수가 참값을 돌려주면 (self.y가 640보다 크면) 2번째 줄을 실행합니다.
- 2번째 줄에 있는 badguy = Badguy()는 악당 생성자입니다.
- 즉, badguy라는 악당 클래스의 인스턴스를 만드는 코드입니다.
- 우리는 이미 badguy라는 악당 클래스의 인스턴스를 갖고 있습니다.
- 스크린 바닥으로 떨어진 녀석 말이에요.
- 사실 파이썬은 새로운 악당을 만들 때 낡은 악당을 쓰레기통에 넣습니다.
- 그러면 새로운 악당이 스크린 아래로 질주하지요.

5. 바닥으로 떨어지는 악당들

- 이제 dy와 dx의 값을 조금씩 바꿔 봅시다.

```
def __init__(self):
    self.x = random.randint(0,570)
    self.y = -100
    self.dy = 0 random.randint(2,6)
    self.dx = 3 random.choice((-1,1))*self.dy
```

- dy의 값이 게임 루프마다 2에서 6픽셀 사이의 임의의 수가 되도록 정합니다.
- dx의 값을 5로 고정하면 오른쪽으로만 가는데, 우리는 왼쪽 또는 오른쪽으로(양수 또는 음수) 가도록 하고 싶습니다.
- 따라서 dx의 값은 dy의 값에 random.choice() 함수를 사용해서 1 또는 -1을 곱한 값으로 정합니다.

Thank You