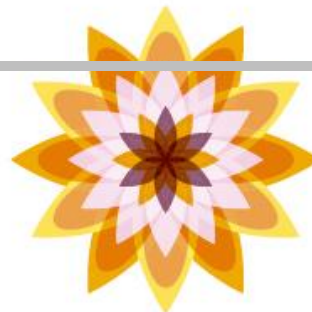
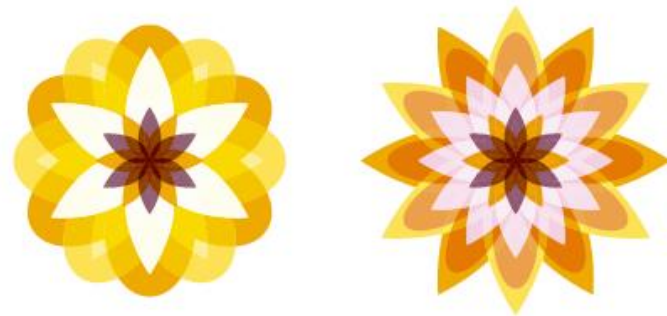


Chapter 10
악당 처치



1. 충돌 감지

- 앞에서 빗방울과 마이크 사이의 충돌을 감지했습니다.
- 직사각형 모양의 마이크 이미지와 점 사이의 충돌을 감지하기 위해 `pygame.Rect().collidepoint()` 함수를 사용했지요.
- 여기에서는 원과 점들 사이의 충돌을 감지해 봅시다.
- 이런 걸 해 주는 함수는 없기 때문에 수고스럽게 직접 만들어야 하지만, 다행스럽게도 그리 힘들진 않습니다.
- 악당들을 직사각형이라고 생각하고 `Rect.collidepoint()` 함수를 사용해도 됩니다.
- 그러면 아래와 같은 코드가 되겠지요.

```
def touching(self,missile):  
    return pygame.Rect((self.x,self.y),(70,45)).collidepoint(missile.x,missile.y)
```

1. 충돌 감지

- 하지만 새로운 것을 배워야 하니까 악당들을 원이라고 가정합시다.
- 아래 코드는 악당을 원으로 취급하고, 미사일이 원에 닿는지 확인하는 함수입니다. 악당 클래스 안에 씁니다.

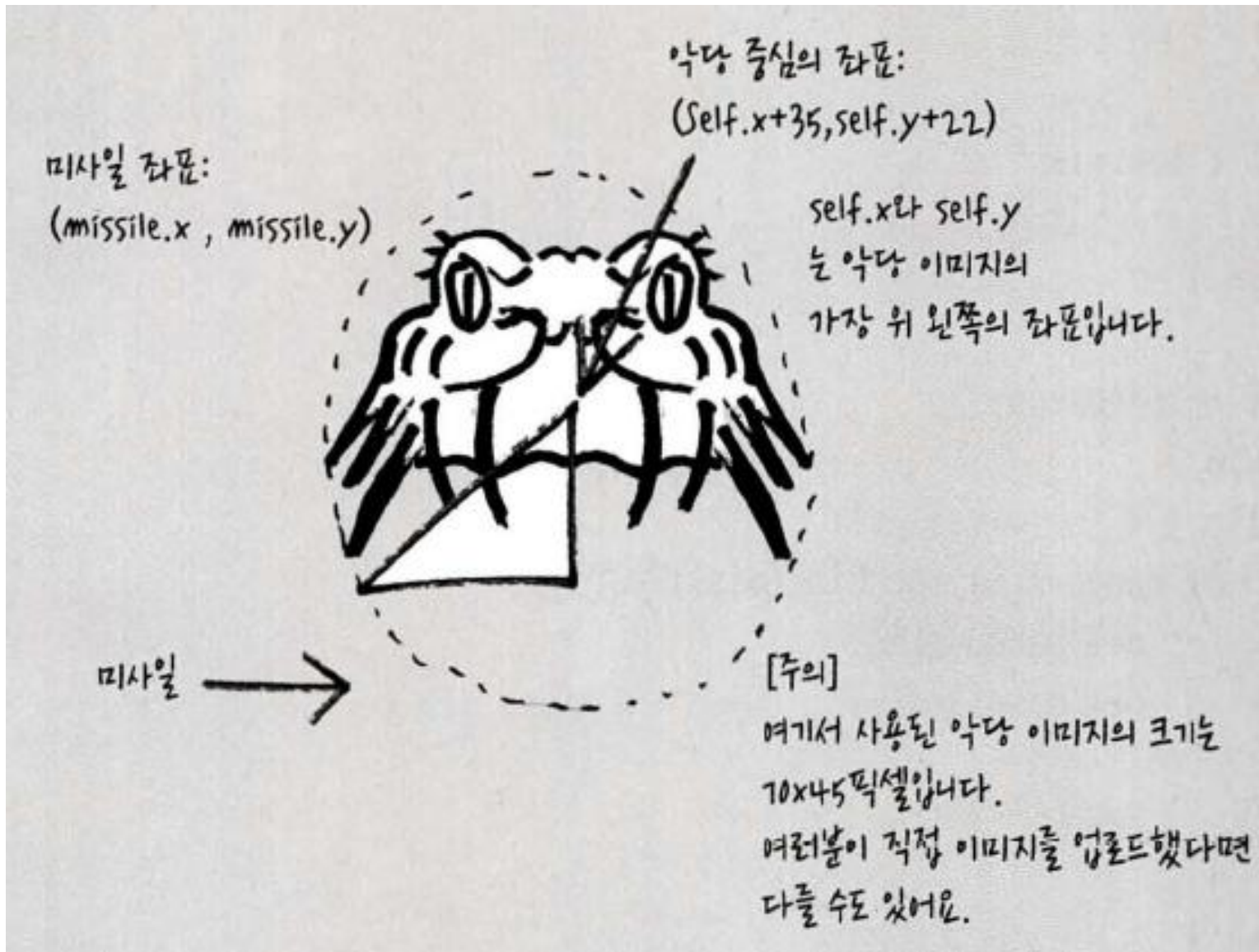
```
class Badguy:  
    (중략)  
    def touching(self, missile):  
        return (self.x+35-missile.x)**2+(self.y+22-missile.y)**2 < 1225
```

- 먼저 touching() 함수가 미사일을 인자로 갖는다는 사실에 주목하세요.
- 이 인자는 함수가 불려올 때 주어집니다.

1. 충돌 감지

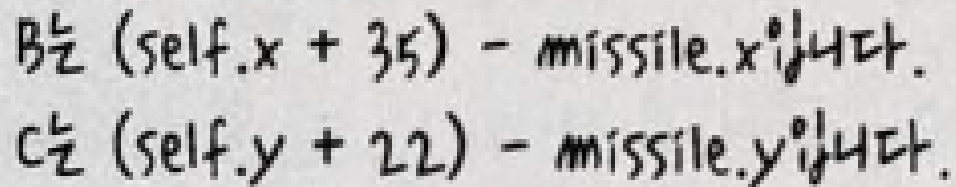
- 이 함수는 미사일 리스트를 도는 while 루프 안에서 불려옵니다.
- 따라서 이 인자는 함수가 불려올 때 while 루프가 다루던 미사일입니다.
- 그런데 우리는 악당을 원으로 취급할 생각입니다.
- 악당의 중심과 미사일 사이의 거리가 35픽셀보다 작으면 이 함수는 참값을 돌려줍니다.
- 왜 그러냐고요? 다음 쪽에서 피타고라스 정리를 사용해 자세히 설명하겠습니다.
- 1225는 35의 제곱입니다. 코드에 그냥 $35^{**}2$ 라고 써도 괜찮지만, 우연히도 우리는 35^2 이 1225임을 알고 있습니다.
- $**2$ 는 파이썬에서 제곱을 표시하는 방법입니다.

1-1. 피타고라스 정리



1-1. 피타고라스 정리

- 피타고라스 정리로부터 A^2 은 B^2+C^2 이 됨을 알 수 있습니다.
- A가 악당을 둘러싼 원의 반지름보다 작으면 미사일이 악당에 닿은 것입니다.
- 원의 반지름은 35픽셀입니다.
- 이 말은 A^2 이 1225(즉, 35^2)보다 작다는 것입니다 따라서 B^2+C^2 도 1225보다 작아야 합니다.



```
B는 (self.x + 35) - missile.x입니다.  
C는 (self.y + 22) - missile.y입니다.
```

- 따라서 $((self.x + 35) - missile.x)^2 + ((self.y + 22) - missile.y)^2$ 이 1225보다 작으면 미사일이 악당에 닿은 것입니다.

2. 미사일로 악당을 맞추면

- 게임 루프 안에서 악당과 미사일을 움직이고 그리는 두 코드 뭉치 아래 다음 코드를 씁니다.

```
while 1:
    (중략)
    i = 0
    while i < len(missiles):
        (중략)
        i += 1
    i = 0
    while i < len(badguys):
        j = 0
        while j < len(missiles):
            if badguys[i].touching(missiles[j]):
                del badguys[i]
                del missiles[j]
                i -= 1
                break
            j += 1
        i += 1
    pygame.display.update()
```

2. 미사일로 악당을 맞추면

- 루프 안에 루프가 있지요? 악당이 미사일에 맞았는지 하나하나 확인하는 코드입니다.
- 악당 리스트의 첫 번째 악당(악당번호 i)부터 시작하지요.
- j 루프는 모든 미사일 위치와 악당이 미사일에 맞았는지를 확인하고, i 로 갔다가 다시 돌아요.
- i 루프 안에 j 루프가 여럿 있는 셈입니다.
- 악당이 미사일에 맞으면 둘 다 삭제하고 다음 악당으로 넘어가지만, 아니라면 그냥 다음 악당을 확인합니다.
- 미사일과 악당을 삭제했다면 더 이상 존재하지 않으니 찾아볼 필요도 없습니다.
- 따라서 j 루프를 멈춰야 합니다. 그 다음엔 i 에 1을 더해 처음으로 돌아갑니다.
- 스크린 어딘가에서 악당이 미사일에 맞았을지도 모르거든요.

2. 미사일로 악당을 맞추면

- 루프를 멈추기 위해서 `break` 문을 사용합니다.
- 어차피 `j` 루프의 시작 지점으로 돌아가서 다시 시작하니까 `j` 루프에서는 1을 빼지 않아도 됩니다.
- 악당과 미사일 리스트에 각각 5개의 항목이 들어 있고, 2번 악당에 2번 미사일이 닿았다고 해 봅시다.
- `break` 문이 우리를 `j` 루프 밖으로 토해내기 때문에 다음 코드는 `i += 1`이어야 합니다.
- 그럼 다음 `i` 루프는 새로운 2번 악당을 다루겠지요.
- 반면 `j` 루프는 0부터 다시 시작할 테고, 리스트의 모든 미사일은 지난번보다 1만큼 줄었겠지요.
- 새로운 미사일을 방금 발사하지 않았다면요.



Thank You
