코드쌤: 지혜야, 슬기야! 지난 시간에는 pygame.key.get_pressed()로 드래곤을 부드럽게 움직이고, Clock으로 게임 속도도 일정하게 맞췄었지?

지혜: 네! 제 드래곤이 이제 키보드를 누르고 있는 동안 스르륵~ 하고 움직여서 훨씬 게임다워졌어요! 🙆⬜➡️⬜

슬기: 맞아요! 근데 선생님, 드래곤이 신나게 움직이다가 화면 밖으로 휙! 나가버리더라고요. 다시 데려오느라 힘들었어요. 😅 그리고 게임이라면 뭔가 목표가 있어야 할 것 같은데… 예를 들면 동전 먹기 같은 거요!

코드쌤: 오호! 슬기가 아주 중요한 두 가지를 이야기해 줬네! 바로 **캐릭터가 화면 경계를 벗어나지 않게 하는 방법**과, 게임에 재미를 더하는 **'충돌 감지'** 기능이야! 오늘은 이 두 가지 마법을 우리 드래곤 게임에 넣어볼 거란다. 드래곤이 벽에 "쿵!" 하고 부딪히지 않고, 반짝이는 동전을 "냠냠!" 먹도록 말이지!

(코드 조각 1: 화면 경계 처리 - 드래곤, 밖으로 나가지 마!)

```
import pygame, random # random 모듈은 동전 위치를 무작위로 바꾸는 데 쓸 거야!

pygame.init()

WINDOW_WIDTH = 600
WINDOW_HEIGHT = 300

display_surface = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
pygame.display.set_caption("드래곤, 동전을 먹어봐!") # 캡션 변경!

FPS = 60
clock = pygame.time.Clock()
VELOCITY = 5

# 드래곤 이미지 로드 및 초기 위치 (이전과 거의 동일)
dragon_img = pygame.image.load("data/dragon_right.png")
dragon_rect = dragon_img.get_rect()
dragon_rect.topleft = (25, 25) # 시작은 왼쪽 위 구석에서!

# --- 새로운 친구 등장! 반짝이는 동전! ---
coin_img = pygame.image.load("data/coin.png") # 동전 이미지도 불러오자!
coin_rect = coin_img.get_rect() # 동전의 사각형 정보!
coin_rect.center = (WINDOW_WIDTH//2, WINDOW_HEIGHT//2) # 처음엔 화면 중앙에!

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
```

```
            running = False

    keys = pygame.key.get_pressed()

    # --- 여기가 첫 번째 마법! 화면 경계 안에서만 움직이기! ---
    # 왼쪽으로 가는데, 드래곤의 왼쪽 끝(dragon_rect.left)이 화면 왼쪽 끝(0)보다 클 때만!
    if keys[pygame.K_LEFT] and dragon_rect.left > 0:
        dragon_rect.x -= VELOCITY
    # 오른쪽으로 가는데, 드래곤의 오른쪽 끝(dragon_rect.right)이 화면 오른쪽 끝(WINDOW_WIDTH)보다 작을 때
    elif keys[pygame.K_RIGHT] and dragon_rect.right < WINDOW_WIDTH:
        dragon_rect.x += VELOCITY
    # 위로 가는데, 드래곤의 위쪽 끝(dragon_rect.top)이 화면 위쪽 끝(0)보다 클 때만!
    elif keys[pygame.K_UP] and dragon_rect.top > 0:
        dragon_rect.y -= VELOCITY
    # 아래로 가는데, 드래곤의 아래쪽 끝(dragon_rect.bottom)이 화면 아래쪽 끝(WINDOW_HEIGHT)보다 작을 때만
    elif keys[pygame.K_DOWN] and dragon_rect.bottom < WINDOW_HEIGHT:
        dragon_rect.y += VELOCITY

    # ... (충돌 감지 및 그리기 코드는 아래에) ...
```

지혜: 와! `if keys[pygame.K_LEFT] and dragon_rect.left > 0:` 이렇게 `and` 뒤에 조건이 하나 더 붙었네요! `dragon_rect.left > 0`은 "드래곤의 왼쪽 가장자리가 화면 왼쪽 끝(0)보다 오른쪽에 있니?" 라고 물어보는 건가요?

코드쌤: 정확해, 지혜! `dragon_rect.left`는 드래곤 이미지의 가장 왼쪽 x좌표를 의미해. 그래서 `dragon_rect.left > 0`이라는 조건은 "드래곤이 아직 화면 왼쪽 벽에 부딪히지 않았니?" 라고 확인하는 거야. 이 조건이 참일 때만, 즉 드래곤이 화면 안에 있을 때만 왼쪽으로 움직이도록 하는 거지.

슬기: 아하! 그럼 `dragon_rect.right < WINDOW_WIDTH`는 "드래곤 오른쪽 끝이 화면 오른쪽 벽보다 왼쪽에 있니?", `dragon_rect.top > 0`은 "드래곤 위쪽 끝이 화면 위쪽 벽보다 아래에 있니?", `dragon_rect.bottom < WINDOW_HEIGHT`는 "드래곤 아래쪽 끝이 화면 아래쪽 벽보다 위에 있니?" 라고 확인하는 거네요! 그래서 드래곤이 화면 밖으로 못 나가는 거군요! 똑똑한데요!

코드쌤: 맞아! `Rect` 객체에는 `left`, `right`, `top`, `bottom`처럼 각 가장자리의 위치를 알려주는 편리한 속성들이 있어서, 이렇게 화면 경계를 쉽게 확인할 수 있단다.

코드쌤: 자, 이제 드래곤이 화면 안에서만 안전하게 움직이게 되었으니, 반짝이는 동전을 먹는 마법을 배워볼까?

(코드 조각 2: 충돌 감지 - 드래곤, 동전 냠냠!)

```
# ... (이전 코드는 그대로) ...

    # --- 여기가 두 번째 마법! 드래곤과 동전이 만났나? (충돌 감지!) ---
    # "드래곤의 사각형(dragon_rect)과 동전의 사각형(coin_rect)이 서로 겹쳤니?"
```

```
    if dragon_rect.colliderect(coin_rect):
        print("우와! 드래곤이 동전을 먹었어요! 냠냠!") # (원래 코드에는 print문이 없지만, 이해를 돕기 위
        # 동전을 먹었으니, 동전을 새로운 무작위 위치로 옮기자!
        # random.randint(최소값, 최대값)은 최소값과 최대값 사이의 정수를 아무거나 하나 골라줘!
        # 동전 이미지의 가로/세로 크기가 32x32라고 가정하고, 36으로 넉넉하게 뺐어요. (원래 코드의 36)
        coin_rect.x = random.randint(0, WINDOW_WIDTH - 36)
        coin_rect.y = random.randint(0, WINDOW_HEIGHT - 36)


    # --- 화면 그리기 ---
    display_surface.fill((0, 0, 0)) # 배경 지우기

    # (새로운 코드!) 드래곤과 동전의 사각형 테두리를 그려서 충돌 영역을 눈으로 확인해보자!
    pygame.draw.rect(display_surface, (255, 0, 0), dragon_rect, 1) # 빨간색 테두리 (드래곤)
    pygame.draw.rect(display_surface, (0, 255, 0), coin_rect, 1)   # 초록색 테두리 (동전)

    display_surface.blit(dragon_img, dragon_rect) # 드래곤 그리기
    display_surface.blit(coin_img, coin_rect)   # 동전 그리기!

    pygame.display.update() # 화면 보여주기!
    clock.tick(FPS) # 시간 제어!

pygame.quit()
```

지혜: `if dragon_rect.colliderect(coin_rect):` 요? `colliderect`는 "사각형끼리 충돌했니?" 라고 물어보는 특별한 마법 주문인가요?

코드쌤: 바로 그거야, 지혜! `colliderect()`는 하나의 `Rect` 객체가 다른 `Rect` 객체와 **서로 겹치는지(충돌하는지)** 확인해주는 아주 편리한 메소드란다. 만약 두 사각형이 조금이라도 겹치면 `True`를, 전혀 겹치지 않으면 `False`를 돌려줘.

슬기: 와! 그럼 드래곤 이미지의 사각형(`dragon_rect`)이랑 동전 이미지의 사각형(`coin_rect`)이 만나면 저 `if` 문이 참이 되는 거네요! 그리고 그 안에서 `coin_rect.x = random.randint(...)` 이렇게 동전 위치를 막 아무데나 바꿔버리니까, 동전을 먹으면 다른 곳에 새로 나타나는 것처럼 보이겠어요!

코드쌤: 정확해! `random.randint(0, WINDOW_WIDTH - 36)`는 0부터 (화면 너비 - 36) 사이의 숫자 중에서 아무거나 하나를 골라서 동전의 새로운 x좌표로 정하는 거야. y좌표도 마찬가지고. 여기서 36을 빼는 이유는 동전 이미지가 화면 밖으로 완전히 벗어나지 않도록 여유를 주는 거겠지? (동전 이미지 크기에 따라 이 값은 조절할 수 있어.)

지혜: 아! 그리고 `pygame.draw.rect(...)`로 드래곤이랑 동전 주변에 빨간색, 초록색 네모 테두리를 그리는 코드가 추가됐네요! 저 테두리가 바로 `colliderect()`가 충돌을 확인하는 실제 영역인 거죠?

코드쌤: 맞아! 저 테두리를 그려보면 이미지의 어느 부분까지가 충돌 영역으로 인식되는지 눈으로 직접 확인할 수 있어서 디버깅할 때 아주 유용하단다. 지금은 이미지 전체를 감싸는 사각형을 사용하고 있지만, 나중에는 더 정교한 충돌 감지 방법도 배울 수 있을 거야.

슬기: 우와! 이제 드래곤이 화면 밖으로 나가지도 않고, 동전도 먹을 수 있게 됐어요! 막 움직여서 동전 다 먹어버릴 거예요! 냠냠! 😋

코드쌤: 하하, 슬기가 아주 신났네! 오늘 우리는 화면 경계 처리 방법과 `colliderect()`를 이용한 간단한 충돌 감지 방법을 배웠어. 이 두 가지는 게임을 만드는 데 정말 정말 기본적이고 중요한 기술이란다! 이제 너희는 이 원리를 응용해서 더 많은 아이템을 먹거나, 장애물을 피하는 게임도 만들 수 있을 거야!

지혜: 화면 경계에서는 `Rect` 객체의 `left`, `right`, `top`, `bottom` 속성을 쓰고, 충돌은 `colliderect()` 메소드를 쓰고! 그리고 `random.randint()`로 무작위 위치도 만들고! 오늘 정말 많은 걸 배웠어요!

코드쌤: 훌륭해! 오늘 배운 내용으로 동전의 크기나 모양을 바꿔보거나, 동전을 먹을 때마다 점수가 올라가도록 코드를 추가해 보는 건 어떨까? 너희들의 창의력으로 더욱 재미있는 게임을 만들어보렴!