코드쌤: 지혜야, 슬기야! 우리가 지난 시간에는 키보드 방향키를 "딸깍" 누를 때마다 드래곤이 한 칸씩 움직이도록 만들었었지?

지혜: 네! 방향키를 누르면 드래곤이 뽕뽕 이동하는 게 신기했어요! ⌨️➡️

슬기: 맞아요! 근데 선생님, 게임에서는 키를 계속 누르고 있으면 캐릭터가 쭉~ 하고 부드럽게 움직이잖아요. 우리 드래곤도 그렇게 만들 수 있어요? 지금은 한 번 누를 때마다 한 번씩만 움직여서 좀 답답해요.

코드쌤: 오호! 슬기가 게임의 '조작감'에 대해 아주 중요한 포인트를 짚었네! 맞아. 대부분의 게임에서는 키를 누르고 있는 동안 캐릭터가 계속 움직이지. 오늘은 바로 그 **키를 누르고 있는 동안 드래곤이 부드럽게 연속적으로 움직이게 하는 방법**과, 게임의 시간을 일정하게 조절해서 어떤 컴퓨터에서든 똑같은 속도로 게임이 돌아가게 만드는 '**시간 마법사 Clock**'에 대해 배울 거란다!

(코드 조각 1: FPS와 Clock 설정 - 시간 마법사 등장!)

```
import pygame

pygame.init()

# 창의 크기 설정
WINDOW_WIDTH = 600
WINDOW_HEIGHT = 300 # WONDOW_HEIGHT 오타 수정했다고 가정할게요!

# 디스플레이 화면 생성
display_surface = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))

# 창 제목 설정
pygame.display.set_caption("드래곤, 부드럽게 움직여봐!") # 캡션 변경!

# --- 여기가 오늘의 첫 번째 마법! 시간 마법사 등장! ---
# 초당 프레임 설정 (FPS: Frames Per Second)
# 1초에 화면을 몇 번이나 새로 그릴까? 숫자가 클수록 부드럽게 보여!
FPS = 60
clock = pygame.time.Clock() # 시간을 관리하는 시계 요정(객체)을 만들자!
# pygame.time 모듈은 시간과 관련된 모든 마법을 부릴 수 있게 해줘.
# 예를 들어 pygame.time.delay(밀리초)는 잠깐 시간을 멈추는 마법이지!

# 이동 속도 설정
VELOCITY = 5

# 드래곤 이미지 로드 및 초기 위치 설정 (이전과 동일)
dragon_img = pygame.image.load("data/dragon_right.png")
```

```
dragon_rect = dragon_img.get_rect()
dragon_rect.center = (WINDOW_WIDTH // 2, WINDOW_HEIGHT // 2)

# ... (게임 루프 코드는 아래에) ...
```

지혜: FPS = 60이요? FPS는 게임에서 많이 들어봤는데, 'Frames Per Second'의 약자 맞죠? 1초에 화면을 60번 새로 그린다는 뜻인가요?

코드쌤: 정확해, 지혜! FPS는 1초 동안 화면이 몇 번이나 깜빡이며 새로운 그림으로 바뀌는지를 나타내는 숫자야. 이 숫자가 높을수록 우리 눈에는 움직임이 더 부드럽게 보인단다. 영화 필름이 빠르게 지나가면서 움직이는 것처럼 보이는 원리와 비슷해!

슬기: clock = pygame.time.Clock()은 뭐예요? Clock은 시계인데… Pygame 안에 시계 요정이 사는 건가요? 🕰️🖼️✨

코드쌤: 하하, 슬기 비유가 정말 귀여운데! 맞아. pygame.time.Clock()은 Pygame 세계의 시간을 정확하게 측정하고 관리하는 **'시계 요정'** 같은 존재라고 생각하면 돼. 이 시계 요정이 있어야 우리가 정한 FPS에 맞춰서 게임이 너무 빠르거나 느리지 않게, 일정한 속도로 돌아가도록 도와줄 수 있단다.

코드쌤: 자, 그럼 이제 이 시계 요정과 함께, 키를 계속 누르고 있을 때 드래곤이 부드럽게 움직이도록 만들어 볼까?

(코드 조각 2: 연속적인 키 입력 처리 - pygame.key.get_pressed())

```
# ... (이전 코드는 그대로) ...

running = True
while running:
    # 이벤트 처리 (창 닫기 등)
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

        # (원래 코드에 있던 KEYDOWN 이벤트 처리 부분은 여기서 생략하고,
        #  연속 입력 처리 부분에 집중해서 설명할게요!
        #  원래 코드는 KEYDOWN 이벤트도 있고, 아래의 get_pressed()도 같이 사용하고 있어요.)
        # if event.type == pygame.KEYDOWN:
        #     ... (생략) ...

    # --- 여기가 오늘의 두 번째 마법! 연속 키 입력 감지! ---
    # "지금 어떤 키들이 눌려있는지 싹 다 알려줘!" 하는 마법 주문!
    keys = pygame.key.get_pressed() # 모든 키의 현재 눌림 상태를 가져옴!

    # 만약 '왼쪽 방향키'가 눌려 있다면? (keys 리스트에서 해당 키가 True라면)
```

```
    if keys[pygame.K_LEFT]:
        dragon_rect.x -= VELOCITY
    # 'elif'를 사용한 이유는 한 번에 하나의 방향으로만 움직이게 하려는 의도일 수 있어요!
    # 만약 동시에 여러 방향키를 눌렀을 때 대각선 이동 등을 구현하려면 'if'를 여러 개 쓸 수도 있겠죠.
    elif keys[pygame.K_RIGHT]:
        dragon_rect.x += VELOCITY
    elif keys[pygame.K_UP]:
        dragon_rect.y -= VELOCITY
    elif keys[pygame.K_DOWN]:
        dragon_rect.y += VELOCITY

    # --- 화면 그리기 ---
    display_surface.fill((0, 0, 0)) # 배경 지우기
    display_surface.blit(dragon_img, dragon_rect) # 드래곤 그리기
    pygame.display.update() # 화면 보여주기!

    # --- 시간 마법사의 활약! ---
    # "시계 요정아, 우리가 정한 FPS(60)에 맞춰서 다음 그림 그릴 때까지 잠깐 쉬어줘!"
    # 이렇게 해야 게임이 너무 빨리 돌아가지 않고 일정한 속도를 유지해!
    clock.tick(FPS)

pygame.quit()
```

지혜: keys = pygame.key.get_pressed()요? 이건 "지금 눌려있는 모든 키들의 상태를 알려줘!" 라는 주문인가요? keys 안에는 뭐가 들어있어요?

코드쌤: 아주 정확해, 지혜! pygame.key.get_pressed()는 지금 이 순간, 키보드의 모든 키들이 **"눌려있다(True)"** 또는 **"안 눌려있다(False)"** 상태인지 알려주는 아주 긴~ 리스트(또는 비슷한 꾸러미)를 keys 변수에 담아줘.

슬기: 와! 그럼 if keys[pygame.K_LEFT]: 이건 "keys 꾸러미 안에서 왼쪽 방향키에 해당하는 칸이 True(눌려있음)니?" 라고 물어보는 거네요! 그래서 왼쪽 방향키를 계속 누르고 있으면 저 if 문이 계속 참이 되니까 드래곤이 계속 왼쪽으로 움직이겠어요!

코드쌤: 빙고! 바로 그거야! KEYDOWN 이벤트는 키를 "딸깍" 누르는 순간에만 발생하지만, pygame.key.get_pressed()는 게임 루프가 한 바퀴 돌 때마다 **"지금 이 순간에도 계속 눌려있니?"** 를 확인해. 그래서 키를 누르고 있는 동안에는 keys[해당키]가 계속 True가 되고, 드래곤은 VELOCITY만큼 계속해서 부드럽게 움직이는 거지!

지혜: 아! 그래서 KEYDOWN 이벤트로 한 칸씩 움직이는 거랑, get_pressed()로 계속 움직이는 거랑 느낌이 다르군요! get_pressed()가 훨씬 게임 같아요!

코드쌤: 맞아! 그리고 게임 루프의 맨 마지막에 있는 clock.tick(FPS) 이게 바로 시계 요정의 가장 중요한 역할이야.

슬기: clock.tick(FPS)는 뭐하는 거예요? 아까 FPS는 1초에 60번 화면 그린다고 했는데…

코드쌤: clock.tick(FPS)는 시계 요정에게 "이봐, 우리가 1초에 60번만 그림을 그리기로 약속했잖아? 이번 그림 다 그렸으니까, 다음 그림 그릴 시간까지 딱 맞춰서 잠깐만 쉬었다 가자!" 하고 알려주는 거야. 만약 이 clock.tick(FPS)가 없으면, 컴퓨터는 최대한 빨리 게임 루프를 휙휙 돌면서 그림을 엄청나게 많이 그리려고 할 거야. 그럼 어떤 컴퓨터에서는 게임이 너무 빨리 진행되고, 어떤 컴퓨터에서는 느리게 진행될 수 있겠지?

지혜: 아! clock.tick(FPS)가 있어서 게임 루프가 너무 빨리 돌지 않도록 속도를 조절해주고, 그래서 1초에 FPS만큼만 화면이 업데이트되도록 맞춰주는 거군요! 그래서 모든 컴퓨터에서 비슷한 속도로 게임을 즐길 수 있게 되는 거고요!

코드쌤: 완벽해! clock.tick(FPS)는 게임의 일관된 속도를 유지하고, 부드러운 움직임을 만드는 데 아주 중요한 역할을 한단다. 오늘 우리는 pygame.key.get_pressed()로 연속적인 키 입력을 받는 방법과, pygame.time.Clock()으로 게임의 시간을 제어하는 방법을 배웠어. 이제 너희 드래곤은 훨씬 더 자연스럽고 멋지게 움직일 수 있을 거야!

슬기: FPS, Clock, get_pressed()! 오늘 배운 마법 주문들로 제 드래곤이 춤추듯이 움직이게 만들 거예요! 너무 신나요! 🐉💃

코드쌤: 하하, 슬기의 상상력은 정말 대단한걸! 오늘 배운 내용으로 드래곤의 움직임을 마음껏 조절해보고, FPS 값을 바꿔보면 움직임이 어떻게 달라지는지도 실험해보렴! 다음 시간에는 또 어떤 재미있는 마법이 기다리고 있을까?