코드쌤: 지혜야, 슬기야! 지난 시간에는 Player 캐릭터를 '스프라이트'로 만들고, '스프라이트 그룹'이라는 팀 가방에 넣어서 편리하게 관리하는 법을 배웠지?

지혜: 네! my_player_group.update()랑 my_player_group.draw() 두 줄만으로 플레이어가 움직이고 화면에 나타나서 정말 신기했어요! 🦸⛹️‍♀️🏹⚽

슬기: 맞아요! 이제 우리 히어로가 외롭지 않게 같이 놀(?) 몬스터들도 만들어주고 싶어요! 몬스터들도 막 움직이고 그러면 더 재미있을 것 같아요!

코드쌤: 오호! 아주 좋은 생각이야, 슬기! 오늘은 바로 그 **'몬스터' 스프라이트 클래스**를 자세히 살펴볼 거야. 이 몬스터들은 각자 다른 모습과 다른 속도로, 심지어 움직이는 방향도 제멋대로 정해져서 화면을 통통 튕기며 돌아다닌단다! 마치 살아있는 것처럼 말이지!

(코드 조각 1: 'Monster' 스프라이트 클래스 만들기 - 개성 만점 몬스터 군단 등장!)

```
import pygame, random # random 모듈이 몬스터를 다양하게 만드는 데 핵심 역할을 해!

pygame.init()

WINDOW_WIDTH = 1200
WINDOW_HEIGHT = 700

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

window_surface = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
pygame.display.set_caption("몬스터들이 나타났다!") # 캡션 변경!

class Game(): # Game 클래스는 아직 비어있네! (나중에 게임 전체를 관리하는 역할로 쓰일 수 있어)
  pass

# Player 클래스는 이전과 동일하다고 가정 (설명 생략)
class Player(pygame.sprite.Sprite):
  def __init__(self):
    super().__init__()
    self.image = pygame.image.load("hero.png")
    self.rect = self.image.get_rect()
    self.rect.centerx = WINDOW_WIDTH/2
    self.rect.bottom = WINDOW_HEIGHT
    self.velocity = 8
  def update(self):
    keys = pygame.key.get_pressed()
```

```
    if keys[pygame.K_LEFT] and self.rect.left > 0:
      self.rect.x -= self.velocity
    elif keys[pygame.K_RIGHT] and self.rect.right < WINDOW_WIDTH:
      self.rect.x += self.velocity
    elif keys[pygame.K_UP] and self.rect.top > 0:
      self.rect.y -= self.velocity
    elif keys[pygame.K_DOWN] and self.rect.bottom < WINDOW_HEIGHT:
      self.rect.y += self.velocity

# --- 여기가 오늘의 주인공! Monster 스프라이트 클래스! ---
class Monster(pygame.sprite.Sprite): # 역시 pygame.sprite.Sprite를 상속!
  # 몬스터를 만들 때 x좌표, y좌표, 그리고 어떤 이미지로 만들지 정보를 받아오네!
  def __init__(self, x, y, image):
    super().__init__() # 부모 클래스 초기화는 필수!
    self.image = image # 전달받은 이미지로 몬스터 모습 설정!
    self.rect = self.image.get_rect()
    self.rect.topleft = (x, y) # 전달받은 x, y 좌표에 몬스터 배치!

    # --- 몬스터만의 특별한 능력들! ---
    # 1. 각 몬스터마다 다른 속도! (1에서 5 사이의 무작위 정수)
    self.velocity = random.randint(1, 5)

    # 2. 각 몬스터마다 처음 움직일 가로(dx), 세로(dy) 방향을 무작위로 결정!
    # random.choice([-1, 1])는 -1 또는 1 중에서 하나를 아무거나 골라줘!
    # dx가 -1이면 왼쪽, 1이면 오른쪽으로 움직이겠지? dy도 마찬가지!
    self.dx = random.choice([-1, 1]) # direction x (x축 이동 방향)
    self.dy = random.choice([-1, 1]) # direction y (y축 이동 방향)

  # 몬스터의 update 메소드: 매 프레임마다 몬스터를 움직이고, 벽에 부딪히면 튕겨 나오게!
  def update(self):
    # 설정된 속도(velocity)와 방향(dx, dy)에 따라 몬스터 위치 업데이트!
    self.rect.x += self.velocity * self.dx
    self.rect.y += self.velocity * self.dy

    # --- "통통" 튕기는 마법! (벽에 부딪히면 방향 바꾸기) ---
    # 만약 몬스터의 왼쪽 끝이 화면 왼쪽(0)보다 작거나 같거나,
    # 또는 몬스터의 오른쪽 끝이 화면 오른쪽(WINDOW_WIDTH)보다 크거나 같다면? (가로 벽 충돌!)
    if self.rect.left <= 0 or self.rect.right >= WINDOW_WIDTH:
      self.dx *= -1 # x축 이동 방향을 반대로! (dx에 -1을 곱하면 부호가 바뀜)
```

```
    # 만약 몬스터의 위쪽 끝이 화면 위쪽(0)보다 작거나 같거나,
    # 또는 몬스터의 아래쪽 끝이 화면 아래쪽(WINDOW_HEIGHT)보다 크거나 같다면? (세로 벽 충돌!)
    # (원래 코드에는 elif로 되어있는데, 가로/세로 벽 동시 충돌(코너)을 고려하면 각각 if로 하는게 나을 수도
    #  설명은 원래 코드의 elif를 따르되, 이런 부분을 아이들이 생각하게끔 유도!)
    elif self.rect.top <= 0 or self.rect.bottom >= WINDOW_HEIGHT:
      self.dy *= -1 # y축 이동 방향을 반대로!


# ... (플레이어 그룹 생성 및 몬스터 그룹 생성/추가 코드는 아래에) ...
```

지혜: 와! Monster 클래스도 Player처럼 pygame.sprite.Sprite를 상속받았네요! 그리고 __init__ 메소드에서 x, y, image를 받아서 몬스터의 처음 위치랑 모습을 정하는군요!

코드쌤: 정확해! Player는 처음 위치가 고정되어 있었지만, Monster는 만들 때마다 다른 위치, 다른 모습으로 나타날 수 있도록 __init__에서 정보를 받아오는 거야. 그리고 정말 재미있는 부분은 바로 그 아래에 있는 self.velocity, self.dx, self.dy 설정이야!

슬기: self.velocity = random.randint(1, 5)요? random.randint()는 저번에 동전 위치 정할 때 썼던 거잖아요! 그럼 몬스터마다 움직이는 속도가 1에서 5 사이 값으로 다 다르게 정해지는 거예요? 어떤 녀석은 느릿느릿, 어떤 녀석은 쌩쌩!

코드쌤: 맞아! random.randint(1, 5) 덕분에 각 몬스터는 자신만의 고유한 속도를 가지게 돼. 그리고 self.dx = random.choice([-1, 1]) 이건 더 신기하지? random.choice()는 괄호 안에 있는 것들 중에서 아무거나 하나를 '선택'해주는 마법이야. 여기서는 -1 또는 1 중에서 하나를 골라서 self.dx에 저장해.

지혜: dx는 'direction x'의 줄임말인가요? 그럼 dx가 -1이면 왼쪽, 1이면 오른쪽으로 움직이는 방향을 나타내는 거네요! dy도 마찬가지로 위(-1) 또는 아래(1) 방향을 정하는 거고요! 그럼 몬스터마다 처음 움직이는 방향도 다 제멋대로겠네요! 완전 개성 넘치는 몬스터들이겠어요!

코드쌤: 바로 그거야! 각 몬스터는 태어날 때부터 자신만의 속도와 초기 이동 방향을 부여받는 거지. 그리고 이 정보들은 몬스터의 update() 메소드에서 사용돼.

슬기: self.rect.x += self.velocity * self.dx 이 부분이요! self.velocity에 self.dx를 곱해서 x좌표에 더하니까, 만약 dx가 -1이면 속도만큼 왼쪽으로, dx가 1이면 속도만큼 오른쪽으로 움직이는 거네요! dy도 똑같고요!

코드쌤: 정확하게 이해했어! 그리고 그 아래에 있는 if 조건문들이 바로 몬스터가 화면 벽에 부딪혔을 때 "통통!" 튕겨 나오는 마법이란다.

지혜: if self.rect.left <= 0 or self.rect.right >= WINDOW_WIDTH: 이 부분은 몬스터가 화면 왼쪽이나 오른쪽에 부딪혔다는 뜻이죠? 그때 self.dx *= -1 이라고 되어있는데, *= -1은 뭐예요?

코드쌤: self.dx *= -1은 self.dx = self.dx * -1과 똑같은 의미야. 즉, self.dx의 값에 -1을 곱하는 거지. 만약 self.dx가 1(오른쪽으로 이동 중)이었다면 -1이 되고, -1(왼쪽으로 이동 중)이었다면 1이 돼. 결국 **x축 이동 방향을 정반대로 바꿔주는 마법**인 거야! 그래서 벽에 부딪히면 반대 방향으로 튕겨 나가는 것처럼 보이는 거지. self.dy *= -1도 마찬가지로 y축 이동 방향을 바꿔주는 거고.

슬기: 와! 그럼 몬스터들이 화면 안에서 계속 통통 튕기면서 돌아다니겠네요! 완전 살아있는 공 같아요! 근데 만약에 몬스터가 정확히 코너에 부딪히면 어떻게 돼요? 지금 코드는 elif로 되어 있어서 가로 벽에 먼저 부딪히면 세로 벽은 확인 안 할 것 같은데…

코드쌤: 오, 슬기가 정말 예리한 질문을 했어! 맞아. 지금처럼 elif로 되어 있으면, 만약 몬스터가 왼쪽 벽과 위쪽 벽에 동시에 부딪히는 상황(왼쪽 위 코너)이라면, 첫 번째 `if self.rect.left <= 0 ...` 조건만 참이 되어서 dx만 바뀌고 dy는 안 바뀔 수 있어. 그래서 코너에서 좀 어색하게 튕길 수도 있지. 만약 가로 충돌과 세로 충돌을 각각 독립적으로 확인하고 싶다면, 두 번째 if도 elif가 아니라 그냥 if로 바꿔주면 된단다. 그렇게 하면 코너에서도 더 자연스럽게 튕겨 나올 수 있을 거야! (이런 걸 직접 수정해보고 결과를 확인하는 게 코딩의 큰 재미 중 하나지!)

(코드 조각 2: 몬스터 군단 생성 및 게임 루프에서의 활약!)

```
# ... (Player 클래스 및 Monster 클래스 정의는 위와 동일) ...

my_player_group = pygame.sprite.Group()
my_player = Player()
my_player_group.add(my_player)

# --- 몬스터들을 담을 '팀 가방'도 만들고, 여러 마리 몬스터를 추가해보자! ---
my_monster_group = pygame.sprite.Group() # 몬스터 팀 가방!

# 몬스터 이미지 파일 이름들을 미리 리스트에 담아두자!
monster_images = ["blue_monster.png", "green_monster.png", "orange_monster.png", "purple_monster.png"]
# (이 이미지 파일들이 코드와 같은 폴더 또는 지정된 경로에 있어야 해요!)

# for 반복문으로 여러 마리 몬스터 만들기!
for i in monster_images: # monster_images 리스트에 있는 파일 이름을 하나씩 꺼내서
  # 무작위 x, y 위치에, 현재 파일 이름(i)에 해당하는 이미지로 몬스터 생성!
  # (이미지 크기를 고려해서 64를 뺐네요. 몬스터 이미지가 64x64인가 봐요!)
  loaded_image = pygame.image.load(i) # 먼저 이미지를 불러오고
  my_monster = Monster(random.randint(0, WINDOW_WIDTH - 64), random.randint(0, WINDOW_HEIGHT - 64), loade
  my_monster_group.add(my_monster) # 만들어진 몬스터를 몬스터 팀 가방에 쏙!

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

  # --- 화면 그리기 ---
```

```
    window_surface.fill((0, 0, 0)) # 배경 지우고

    # 플레이어 그룹 업데이트하고 그리기!
    my_player_group.update()
    my_player_group.draw(window_surface)

    # 몬스터 그룹도 업데이트하고 그리기! (팀워크 발휘!)
    my_monster_group.update() # 이 한 줄이면 모든 몬스터가 알아서 움직이고 튕김!
    my_monster_group.draw(window_surface) # 이 한 줄이면 모든 몬스터가 화면에 짠!

    pygame.display.update()
    clock.tick(FPS)

pygame.quit()
```

지혜: monster_images 리스트에 몬스터 그림 파일 이름을 넣어두고, for 반복문으로 그 리스트에 있는 그림마다 하나씩 몬스터를 만들어서 my_monster_group에 추가하는 거네요! 그럼 지금은 4마리 몬스터가 서로 다른 모습으로 화면에 나타나겠어요!

코드쌤: 정확해! pygame.image.load(i)로 각기 다른 몬스터 이미지를 불러와서 Monster 클래스의 image 매개변수로 전달해주니까, 파란 몬스터, 초록 몬스터, 주황 몬스터, 보라 몬스터가 각각 다른 속도와 방향으로 움직이게 되는 거지!

슬기: 그리고 게임 루프 안에서는 my_monster_group.update()랑 my_monster_group.draw(window_surface) 이 두 줄만으로 모든 몬스터들이 알아서 움직이고, 벽에 튕기고, 화면에 그려지는 거잖아요! 스프라이트 그룹 진짜 편하네요!

코드쌤: 맞아! 이게 바로 스프라이트와 그룹을 사용하는 마법 같은 편리함이지! 오늘 우리는 Monster 클래스를 통해 각 객체(몬스터)가 자신만의 고유한 상태(속도, 방향)를 가질 수 있게 하고, update() 메소드 안에서 그 상태에 따라 스스로 움직이고 주변 환경(벽)과 상호작용하는 방법을 배웠어. 이제 이 몬스터들과 우리 플레이어가 만나면 어떤 일이 벌어질까? 그건 다음 시간에 함께 알아볼 충돌 감지 마법에서 확인할 수 있을 거야!

지혜: 몬스터들이 막 제멋대로 움직이니까 진짜 살아있는 것 같아요! random 모듈이랑 dx, dy 같은 변수를 쓰는 방법도 정말 신기하고요!

코드쌤: 훌륭해! 오늘 배운 내용으로 몬스터의 종류를 더 늘려보거나, 몬스터마다 튕기는 방식을 다르게 해보는 건 어떨까? 너희들의 상상력으로 더욱 개성 넘치는 몬스터 군단을 만들어보렴!