Pygame 애니메이터 만들기

1. 모듈 import 하기

```
import pygame
import os
import re
```

2. 이미지를 로드하는 함수 만들기

```
def load_images(path):
    #로드한 이미지를 저장할 리스트 만들기
    images = []
```

```
#이미지가 위치한 폴더에서 이미지 파일 리스트 읽어오기
file_list = os.listdir(path)
#파일명에 따라 이미지 파일 정렬하기
file_list.sort(key=lambda s: int(re.search(r'\d+', s).group()))
#이미지 파일 리스트로부터 파일 읽어들이기
for file_name in file_list:
    if "Walk" in file_name :
        image = pygame.image.load(path + os.sep + file_name).convert()
        #이미지 파일 크기 설정
        image = pygame.transform.scale(image, (128, 128))
        #이미지 리스트에 읽어들인 이미지 추가하기
        images.append(image)
return images
```

3. 애니메이터 클래스 만들기

1.1. 클래스 이름 : AnimatorSprite

1.2. 부모클래스 : pygame.sprite.Sprite

1.3. 클래스 생성자 만들기

1.3.1.1. 위치와 2번에서 만든 이미지 리스트, 이미지 너비와 높이를 인자로 받는다.

1.3.1.2. 상속받은 부모클래스 생성자 호출

1.3.1.3. 이미지의 이동 속도와 프레임 속도를 지정

1.3.1.4. 이미지를 좌우 반전한 이미지를 추가 생성하여 움직이는 방향에 따라 다른 이미지를 출력하도록 한다.

```
#정답코드
class AnimatedSprite(pygame.sprite.Sprite):

    def __init__(self, position, images, w, h):

        super(AnimatedSprite, self).__init__()

        size = (w, h)

        self.rect = pygame.Rect(position, size)
        self.images = images
        self.images_right = images
        self.images_left = [pygame.transform.flip(image, True, False) for image in images]
        self.index = 0
        self.image = images[self.index]
```

```
self.velocity = pygame.math.Vector2(0, 0)

self.animation_time = 0.1
self.current_time = 0

self.animation_frames = 8
self.current_frame = 0
```

4. 프레임과 시간에 따라 이미지를 교체하는 업데이트 함수 만들기

1.1. 어떻게 하면 이미지를 시간에 맞춰서 교체하여 애니메이션 효과를 줄 수 있을지 생각해 봅시다.

```
#정답코드 3번에 이어서 작성
 def update_time_dependent(self, dt):

        if self.velocity.x > 0:
            self.images = self.images_right
        elif self.velocity.x < 0:
            self.images = self.images_left

        self.current_time += dt
        if self.current_time >= self.animation_time:
            self.current_time = 0
            self.index = (self.index + 1) % len(self.images)
            self.image = self.images[self.index]

        self.rect.move_ip(self.velocity)

 def update_frame_dependent(self):
```

```
        if self.velocity.x > 0:
            self.images = self.images_right
        elif self.velocity.x < 0:
            self.images = self.images_left

        self.current_frame += 1
        if self.current_frame >= self.animation_frames:
            self.current_frame = 0
            self.index = (self.index + 1) % len(self.images)
            self.image = self.images[self.index]

        self.rect.move_ip(*self.velocity)

    def update(self, dt):
        self.update_time_dependent(dt)
        #self.update_frame_dependent()
```

5. pygame을 만들어서 지금까지 만든 애니메이터를 이용해 화면에 캐릭터를 만들어 움직여 봅시다.

5.1. pygame 초기화 하기

5.2. 배경화면 설정하기

5.3. FPS 60 으로 설정하기

5.4. 화면 크기 설정하기

5.5. 이미지를 로드하는 함수를 호출해 봅시다. 이미지가 저장된 폴더명을 함수에 전달해야 합니다.

5.6. 위에서 작성한 애니메이터 클래스 변수를 만들어 봅시다.

5.7. pygame을 처리할 무한반복문을 작성해 봅시다.

5.8. 무한 반복문 안에서 pygame의 키보드 이벤트를 처리하도록 코드를 작성해 봅시다.

5.9. 애니메이터를 업데이트하고 화면에 그려주는 코드를 추가해 봅시다.

```
#정답코드
def main():
    pygame.init()

    SIZE = WIDTH, HEIGHT = 720, 480
    BACKGROUND_COLOR = pygame.Color('black')
    FPS = 60

    screen = pygame.display.set_mode(SIZE)
    clock = pygame.time.Clock()

    images = load_images(path="cat")
    player = AnimatedSprite(position=(100, 100), images=images, w=128, h=128)
    all_sprites = pygame.sprite.Group(player)

    running = True
    while running:
```

```
dt = clock.tick(FPS) / 1000  # Amount of seconds between each loop.

for event in pygame.event.get():
    if event.type == pygame.QUIT:
        running = False
    elif event.type == pygame.KEYDOWN:
        if event.key == pygame.K_RIGHT:
            player.velocity.x = 2
        elif event.key == pygame.K_LEFT:
            player.velocity.x = -2
        elif event.key == pygame.K_DOWN:
            player.velocity.y = 2
        elif event.key == pygame.K_UP:
            player.velocity.y = -2
        elif event.key == pygame.K_SPACE:
            player.velocity.y = -4
    elif event.type == pygame.KEYUP:
        if event.key == pygame.K_RIGHT or event.key == pygame.K_LEFT:
```

```
                    player.velocity.x = 0
                elif event.key == pygame.K_DOWN or event.key == pygame.K_UP:
                    player.velocity.y = 0

        all_sprites.update(dt)

        screen.fill(BACKGROUND_COLOR)
        all_sprites.draw(screen)
        pygame.display.update()

    pygame.quit()

main()
```