

Lecture #9. 캐릭터 컨트롤러 (2)

2D 게임 프로그래밍

이대현 교수

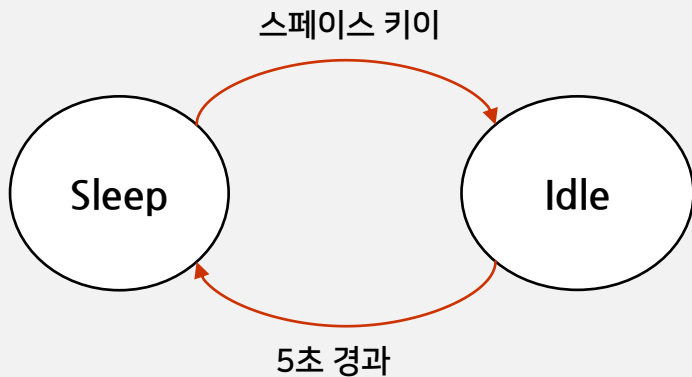


한국공학대학교
TECH UNIVERSITY OF KOREA



캐릭터 컨트롤러 구현(Sleep & Idle)

상태 변환 구현 목표



어떤 상태가 있는가?

어떤 이벤트가 있는가?

Sleep 상태 구현



```
class Sleep:

    def __init__(self, boy):
        self.boy = boy

    def enter(self):
        self.boy.dir = 0

    def exit(self):
        pass

    def do(self):
        self.boy.frame = (self.boy.frame + 1) % 8
        pass

    def draw(self):
        if self.boy.face_dir == 1:
            self.boy.image.clip_composite_draw(self.boy.frame * 100, 300, 100, 100, 3.141592/2, '', self.boy.x - 25, self.boy.y - 25, 100, 100)
        else:
            self.boy.image.clip_composite_draw(self.boy.frame * 100, 200, 100, 100, -3.141592/2, '', self.boy.x + 25, self.boy.y - 25, 100, 100)
```

boy.handle_event 도입

main.py

```
def handle_events():
    global running

    events = get_events()
    for event in events:
        if event.type == SDL_QUIT:
            running = False
        elif event.type == SDL_KEYDOWN and event.key == SDLK_ESCAPE:
            running = False
        else:
            boy.handle_event(event)
```

boy.py

```
class Boy:
    # 중략

    def handle_event(self, event):
        self.state_machine.handle_state_event(('INPUT', event))
```

상태 이벤트 구현 (1)



- 상태 이벤트는 pico2d 의 `get_events` 를 통해서 얻는 "입력" 이벤트와는 다름.
 - ex) `TIME_OUT` - 시간 초과 이벤트 등등이 필요함.
- 튜플을 이용해서 상태 이벤트를 나타내도록 함.
 - (상태 이벤트 종류, 실제 이벤트값)
 - ('INPUT', 실제입력이벤트값)
 - ('TIME_OUT', 0)
 - ('NONE', 0)

```
class Boy:
    def handle_event(self, event):
        self.state_machine.handle_event(('INPUT', event))
```

상태 이벤트 구현 (2)

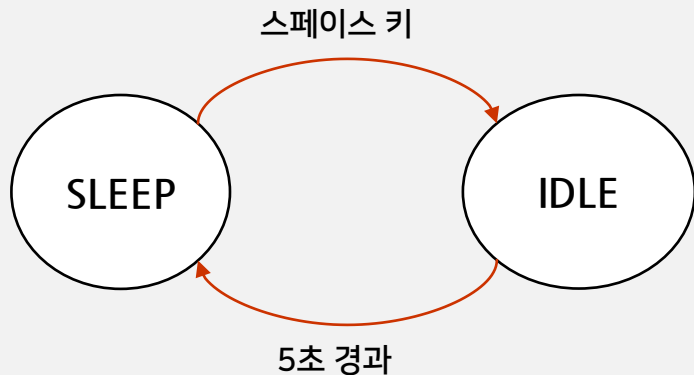


- 이벤트 체크 함수를 이용해서 어떤 이벤트인지 판단할 수 있도록 함.

```
def space_down(e):  
    return e[0] == 'INPUT' and e[1].type == SDL_KEYDOWN and e[1].key == SDLK_SPACE  
  
def time_out(e):  
    return e[0] == 'TIME_OUT'  
  
# 이렇게 쓸수도 있음.  
# time_out = lambda e : e[0] == 'TIME_OUT'
```

boy.py

상태 변환 구현 (1)



현재 상태	이벤트	다음 상태
SLEEP	스페이스 키	IDLE
IDLE	5초 경과	SLEEP

상태 변환 테이블

현재상태와 이벤트로부터 다음 상태를 계산

상태 변환 구현 (2)

```
class Boy:
    def __init__(self):
        self.x, self.y = 400, 90
        self.frame = 0
        self.face_dir = 1
        self.dir = 1
        self.image = load_image('animation_sheet.png')

        self.IDLE = Idle(self)
        self.SLEEP = Sleep(self)
        self.state_machine = StateMachine(
            self.SLEEP,
            {
                self.SLEEP : {space_down: self.IDLE},
                self.IDLE : {time_out: self.SLEEP}
            }
        )
```

상태 변환 구현 (3)



state_machine.py

```
from event_to_string import event_to_string
```

```
class StateMachine:
```

```
    def __init__(self, start_state, state_transitions):
        self.cur_state = start_state
        self.state_transitions = state_transitions
        self.cur_state.enter()
        pass
```

```
    def update(self):
        self.cur_state.do()
```

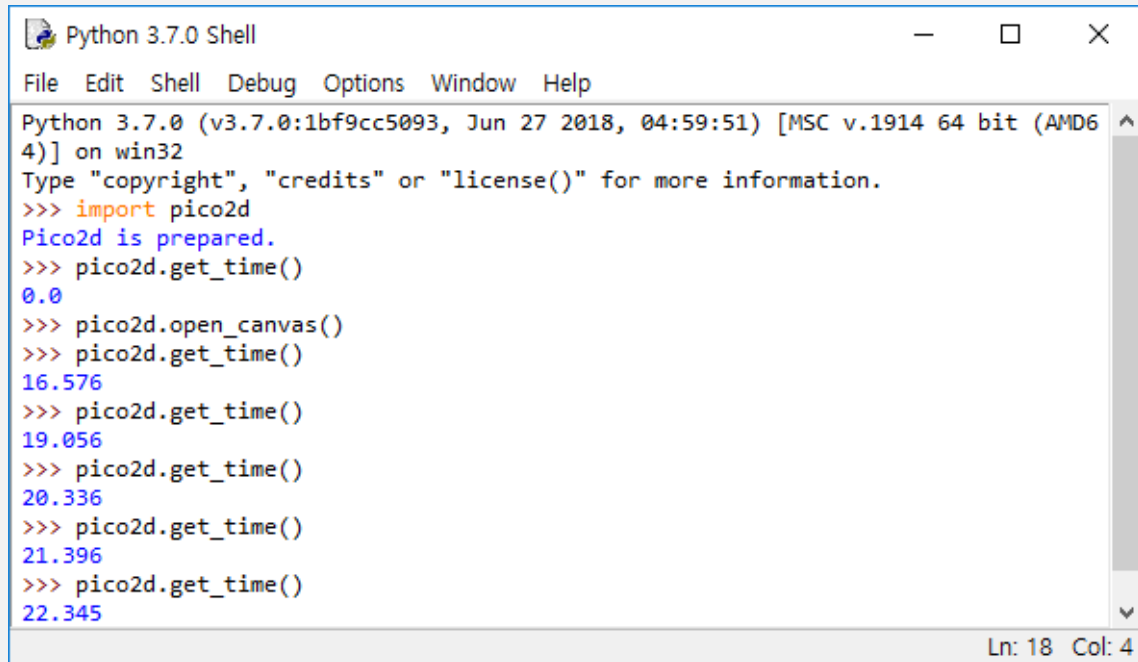
```
    def handle_state_event(self, event):
        for check_event in self.state_transitions[self.cur_state].keys():
            if check_event(event):
                self.cur_state.exit()
                self.next_state = self.state_transitions[self.cur_state][check_event]
                self.next_state.enter()
                print(f'{self.cur_state.__class__.__name__} - {event_to_string(event)} -> {self.next_state.__class__.__name__}')
                self.cur_state = self.next_state
                return
```

```
        print(f'처리되지 않은 이벤트 {event_to_string(event)} 가 있습니다.')
```

```
    def draw(self):
        self.cur_state.draw()
```

get_time()

- pico2d 의 시간 획득 함수.
- Canvas 가 열린 시점의 시간이 0.0이고, 이후 현재까지의 경과 시간을 구하는 함수임.



```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import pico2d
Pico2d is prepared.
>>> pico2d.get_time()
0.0
>>> pico2d.open_canvas()
>>> pico2d.get_time()
16.576
>>> pico2d.get_time()
19.056
>>> pico2d.get_time()
20.336
>>> pico2d.get_time()
21.396
>>> pico2d.get_time()
22.345
Ln: 18 Col: 4
```

TIME_OUT 이벤트 발생과 처리



```
class Idle:
```

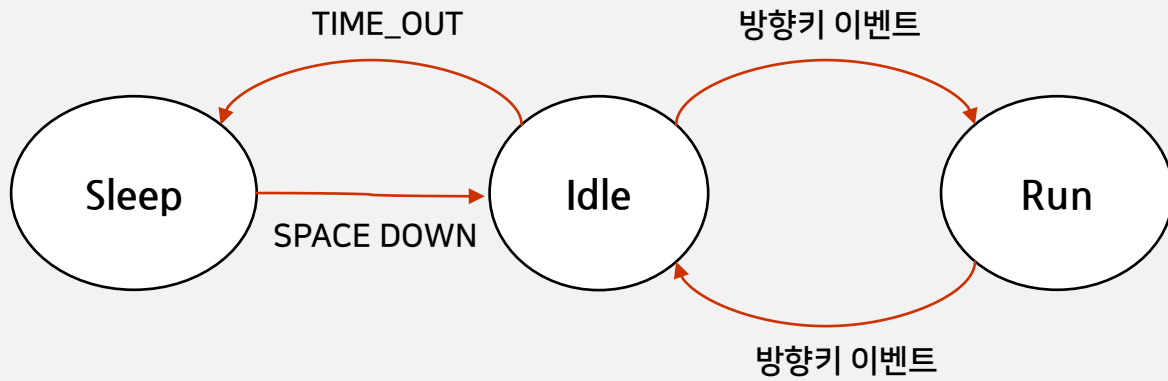
```
    def enter(self):  
        self.boy.wait_time = get_time()
```

```
    def do(self):  
        self.boy.frame = (self.boy.frame + 1) % 8  
        if get_time() - self.boy.wait_time > 2:  
            self.boy.state_machine.handle_state_event(('TIMEOUT', None))
```

boy.py



캐릭터 컨트롤러 구현
(Sleep & Idle & Run)



방향키 이벤트 체크 함수



```
def right_down(e):  
    return e[0] == 'INPUT' and e[1].type == SDL_KEYDOWN and e[1].key == SDLK_RIGHT
```

```
def right_up(e):  
    return e[0] == 'INPUT' and e[1].type == SDL_KEYUP and e[1].key == SDLK_RIGHT
```

```
def left_down(e):  
    return e[0] == 'INPUT' and e[1].type == SDL_KEYDOWN and e[1].key == SDLK_LEFT
```

```
def left_up(e):  
    return e[0] == 'INPUT' and e[1].type == SDL_KEYUP and e[1].key == SDLK_LEFT
```

boy.py

Run 상태 구현



boy.py

```
class Run:
    def __init__(self, boy):
        self.boy = boy

    def enter(self, e):
        if right_down(e) or left_up(e):
            self.boy.dir = self.boy.face_dir = 1
        elif left_down(e) or right_up(e):
            self.boy.dir = self.boy.face_dir = -1

    def exit(self, e):
        pass

    def do(self):
        self.boy.frame = (self.boy.frame + 1) % 8
        self.boy.x += self.boy.dir * 5

    def draw(self):
        if self.boy.face_dir == 1: # right
            self.boy.image.clip_draw(self.boy.frame * 100, 100, 100, 100, self.boy.x, self.boy.y)
        else: # face_dir == -1: # left
            self.boy.image.clip_draw(self.boy.frame * 100, 0, 100, 100, self.boy.x, self.boy.y)
```


상태 변환 규칙

boy.py

```
self.IDLE = Idle(self)
self.SLEEP = Sleep(self)
self.RUN = Run(self)
self.state_machine = StateMachine(
    self.IDLE,
    {
        self.SLEEP : {space_down: self.IDLE},
        self.IDLE : {time_out: self.SLEEP, right_down: self.RUN, left_down: self.RUN, right_up: self.RUN, left_up: self.RUN},
        self.RUN : {right_up: self.IDLE, left_up: self.IDLE, right_down: self.IDLE, left_down: self.IDLE}
    }
)
```

enter, exit 액션에서 이벤트 전달 추가

```
def __init__(self, start_state, state_transitions):
    self.cur_state = start_state
    self.state_transitions = state_transitions
    self.cur_state.enter(('START', None))

def update(self):
    self.cur_state.do()

def handle_state_event(self, event):
    for check_event in self.state_transitions[self.cur_state].keys():
        if check_event(event):
            self.cur_state.exit(event)
            self.next_state = self.state_transitions[self.cur_state][check_event]
            self.next_state.enter(event)
            print(f'{self.cur_state.__class__.__name__} ---- {event_to_string(event)} ----> {self.next_state.__class__.__name__}')
            self.cur_state = self.next_state
            return

# 처리되지 않은 event 를 알려준다.
print(f'처리되지 않은 이벤트 {event_to_string(event)} 가 있습니다.')
```

시작 상태로 들어갈 때는,
START 이벤트에 의해 상태
변화가 일어난 것으로 처리.

exit와 enter에 e를
전달해줌. 상태 변화의 원인인
e를 알려줌으로써 필요한
작업을 할 수 있도록 하기 위함.

