

# Lecture #11. 캐릭터 컨트롤러 (2)

2D 게임 프로그래밍

이대현 교수



한국공학대학교  
TECH UNIVERSITY OF KOREA



캐릭터 컨트롤러  
구현(IDLE)

# boy.handle\_event 도입

```
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)
```

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

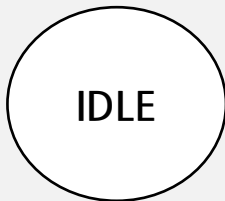
    def update(self):
        self.frame = (self.frame + 1) % 8

    def handle_event(self, event):
        pass

    def draw(self):
        self.image.clip_draw(self.frame * 100, 3 * 100, 100, 100, self.x, self.y)
```

# 상태 변환 구현 목표

---



# #1. 상태의 정의 - boy.py



```
class Idle:
    @staticmethod
    def enter(boy):
        print('Boy Idle Enter')

    @staticmethod
    def exit(boy):
        print('Boy Idle Exit')

    @staticmethod
    def do(boy):
        boy.frame = (boy.frame + 1) % 8

    @staticmethod
    def draw(boy):
        boy.image.clip_draw(boy.frame * 100, boy.action * 100, 100, 100, boy.x, boy.y)
```

여기서 class 의 역할은 특정함수를 모아서 그루핑하는 역할. 객체 생성이 아님!

## #2. 상태 머신 구현 – state\_machine.py



```
class StateMachine:

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

    def start(self, state):
        self.cur_state = state
        self.cur_state.enter(self.o)

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

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

### #3. 소년의 상태 머신 실행



```
class Boy:
    def __init__(self):
        self.x, self.y = 400, 90
        self.frame = 0
        self.action = 3
        self.image = load_image('animation_sheet.png')
        self.state_machine = StateMachine(self)
        self.state_machine.start(Idle)

    def update(self):
        self.state_machine.update()

    def handle_event(self, event):
        pass

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

## 참고: 객체 생성할 때 어떤 값의 전달

```
class Boy:
    def __init__(self, name):
        self.name = name
```

실질적인 첫번째 인자

```
    def print_name(self):
        print(self.name)
```

```
boy1 = Boy('Tom')
boy2 = Boy('John')
```

```
boy1.print_name()
boy2.print_name()
```



## #4. 소년 객체의 전달



```
class StateMachine:
    def __init__(self, o):
        self.o = o

    def start(self):
        self.cur_state.enter(self.o)

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

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

```
class Boy:
    def __init__(self):
        self.x, self.y = 400, 90
        self.frame = 0
        self.action = 3
        self.image = load_image('animation_sheet.png')
        self.state_machine = StateMachine(self)
        self.state_machine.start(Idle)
```

```
class Idle:
```

```
    @staticmethod
    def enter(boy):
        boy.action = 3
        boy.frame = 0
        pass
```

```
    @staticmethod
    def exit(boy):
        pass
```

```
    @staticmethod
    def do(boy):
        boy.frame = (boy.frame + 1) % 8
```

```
    @staticmethod
    def draw(boy):
        boy.image.clip_draw(boy.frame * 100, boy.action * 100, 100, 100, boy.x, boy.y)
```

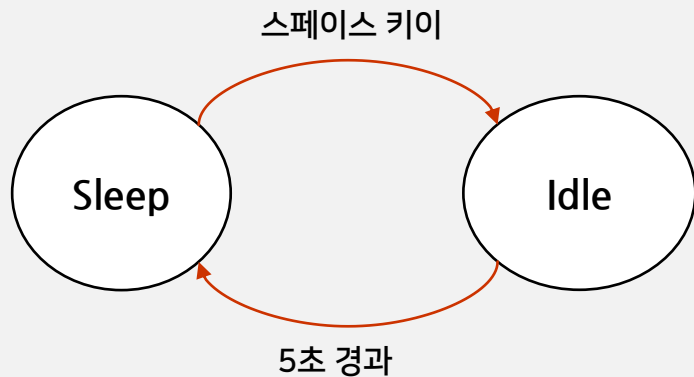




## 캐릭터 컨트롤러 구현(Sleep & Idle)

# 상태 변환 구현 목표

---



어떤 상태가 있는가?

어떤 이벤트가 있는가?

# Sleep 상태 구현



```
class Sleep:

    @staticmethod
    def enter(boy):
        boy.frame = 0

    @staticmethod
    def exit(boy):
        pass

    @staticmethod
    def do(boy):
        boy.frame = (boy.frame + 1) % 8

    @staticmethod
    def draw(boy):
        boy.image.clip_composite_draw(boy.frame * 100, 300, 100, 100,
                                       3.141592 / 2, '', boy.x - 25, boy.y - 25, 100, 100)
```

# 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))
```

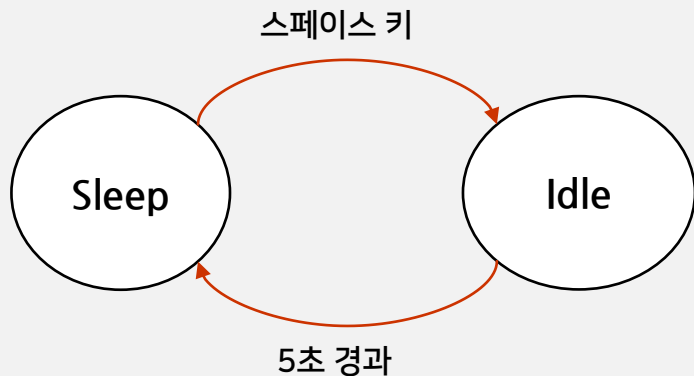
## 상태 이벤트 구현 (2)

- 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'
```

# 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



## 참고: dictionary 아이템 꺼내기

---

```
data = { 1:'LEE', 7:'KIM', 9:'PARK'}  
for no, name in data.items():  
    print(no, name)
```

```
1 LEE  
7 KIM  
9 PARK
```

## 상태 변환 구현 (2)



```
class StateMachine:
    def __init__(self, o):
        self.o = o
        self.event_que = []

    def start(self, state):
        self.cur_state = state
        self.cur_state.enter(self.o, ('START', 0))
        pass

    def add_event(self, e):
        self.event_que.append(e)

    def set_transitions(self, transitions):
        self.transitions = transitions

    def update(self):
        self.cur_state.do(self.o)
        if self.event_que:
            event = self.event_que.pop(0)
            self.handle_event(event)
```

---

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

def handle_event(self, e):
    for event, next_state in self.transitions[self.cur_state].items():
        if event(e):
            self.cur_state.exit(self.o, e)
            self.cur_state = next_state
            self.cur_state.enter(self.o, e)
    return
```

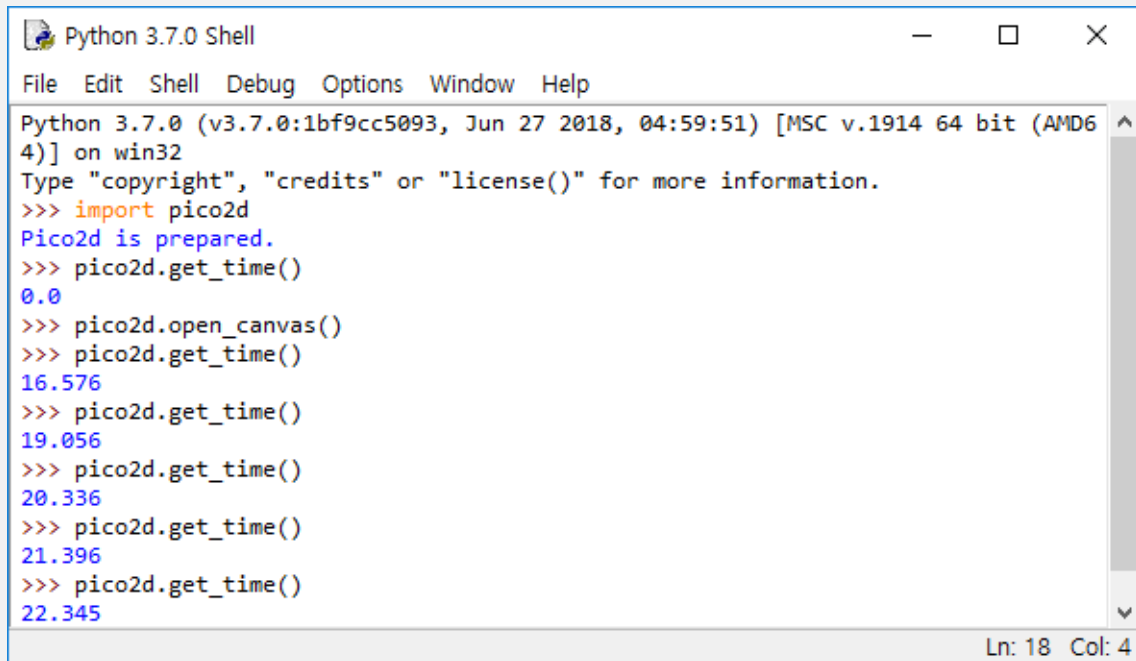
# boy.py

---

```
def __init__(self):
    self.x, self.y = 400, 90
    self.frame = 0
    self.action = 3
    self.image = load_image('animation_sheet.png')
    self.state_machine = StateMachine(self)
    self.state_machine.start(Idle)
    self.state_machine.set_transitions(
        {
            Idle: {time_out: Sleep},
            Sleep: {space_down: Idle}
        }
    )
```

# 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:
```

```
    @staticmethod
```

```
    def enter(boy):
```

```
        boy.action = 3
```

```
        boy.dir = 0
```

```
        boy.frame = 0
```

```
        boy.wait_time = get_time() # pico2d import 필요
```

```
        pass
```

```
    @staticmethod
```

```
    def do(boy):
```

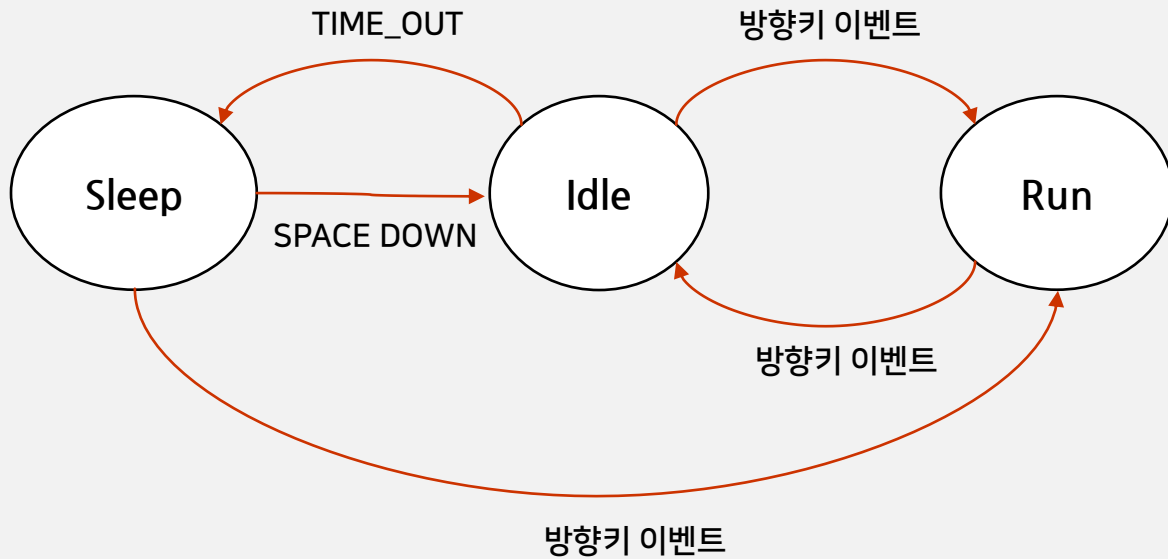
```
        boy.frame = (boy.frame + 1) % 8
```

```
        if get_time() - boy.wait_time > 2:
```

```
            boy.state_machine.add_event(('TIME_OUT', 0))
```



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
```

# Run 상태 구현



```
class Run:
```

```
    @staticmethod
```

```
    def enter(boy, e):
```

```
        if right_down(e) or left_up(e): # 오른쪽으로 RUN
```

```
            boy.dir, boy.action = 1, 1
```

```
        elif left_down(e) or right_up(e): # 왼쪽으로 RUN
```

```
            boy.dir, boy.action = -1, 0
```

```
    @staticmethod
```

```
    def exit(boy, e):
```

```
        pass
```

```
    @staticmethod
```

```
    def do(boy):
```

```
        boy.frame = (boy.frame + 1) % 8
```

```
        boy.x += boy.dir * 5
```

```
        pass
```

```
    @staticmethod
```

```
    def draw(boy):
```

```
        boy.image.clip_draw(boy.frame * 100, boy.action * 100, 100, 100, boy.x, boy.y)
```

# 상태 변환 규칙

---

```
self.state_machine.set_transitions(  
    {  
        Idle: {right_down: Run, left_down: Run, left_up: Run, right_up: Run, time_out: Sleep},  
        Run: {right_down: Idle, left_down: Idle, right_up: Idle, left_up: Idle},  
        Sleep: {right_down: Run, left_down: Run, right_up: Run, left_up: Run, space_down: Idle}  
    }  
)
```

# enter, exit 액션에서 이벤트 전달 추가

```
def start(self, state):
    self.cur_state = state
    self.cur_state.enter(self.o, ('START', 0))
    pass
```

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

```
def handle_event(self, e):
    for event, next_state in self.transitions[self.cur_state].items():
        if event(e):
            self.cur_state.exit(self.o, e)
            self.cur_state = next_state
            self.cur_state.enter(self.o, e)
            return
```

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

# Sleep 상태 수정



```
class Sleep:
    @staticmethod
    def enter(boy, e):
        if start_event(e):
            boy.face_dir = -1
            boy.action = 2
            boy.frame = 0

    @staticmethod
    def exit(boy, e):
        pass

    @staticmethod
    def do(boy):
        boy.frame = (boy.frame + 1) % 8

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

enter, exit 에 이벤트 전달 추가

Idle 상태에서 캐릭터의 방향에  
맞게 회전 구현.

# Idle 상태 수정

```
class Idle:
    @staticmethod
    def enter(boy, e):
        if start_event(e):
            boy.action = 2
            boy.face_dir = -1
        elif right_down(e) or left_up(e):
            boy.action = 2
            boy.face_dir = -1
        elif left_down(e) or right_up(e):
            boy.action = 3
            boy.face_dir = 1

        boy.frame = 0
        boy.wait_time = get_time()
```

이전에 달리고 있는 상황이었으면, 그 때 이동 방향을 반영해서 정지 상태의 방향 결정.