

Lecture #8. 캐릭터 컨트롤러 (1)

2D 게임 프로그래밍

이대현 교수



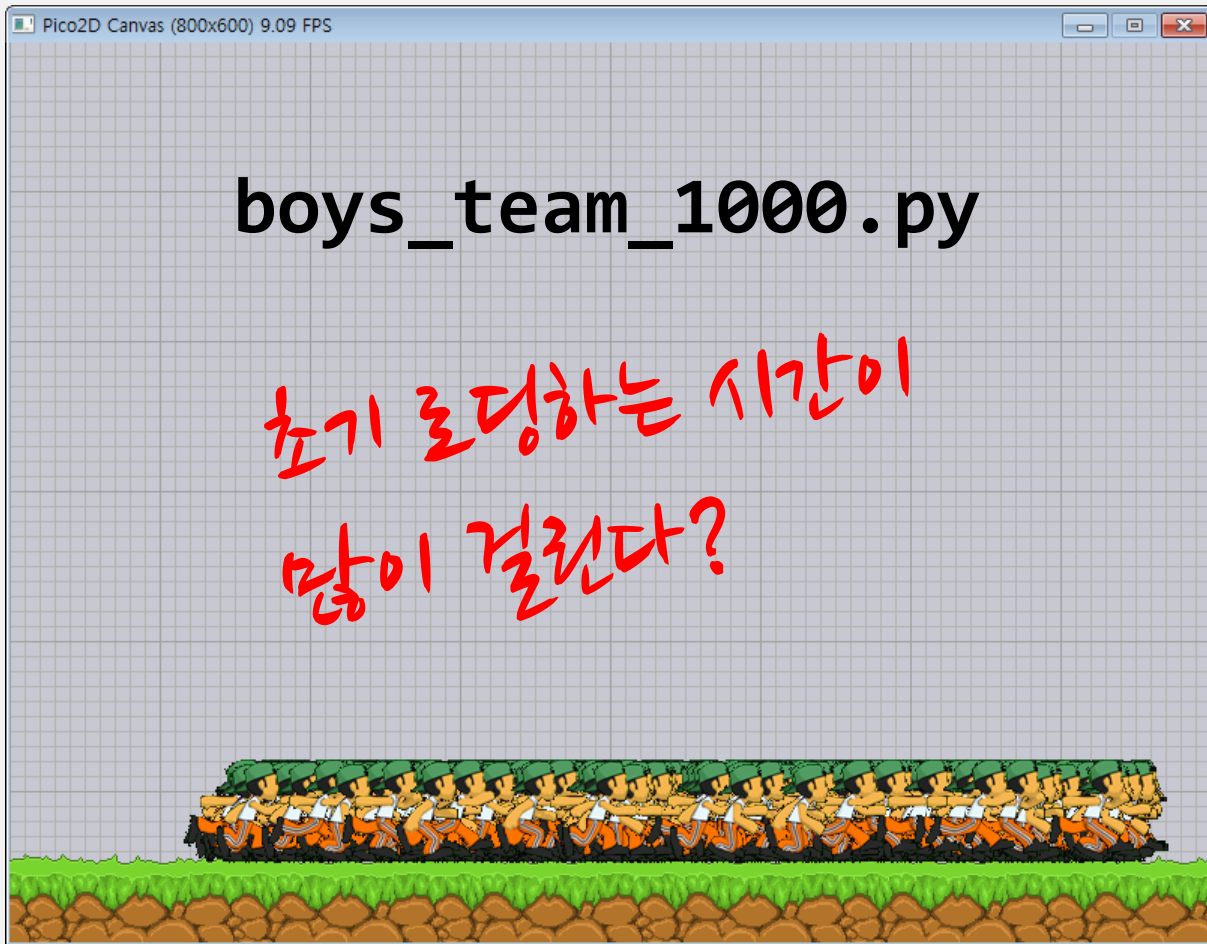
한국공학대학교
TECH UNIVERSITY OF KOREA

학습 내용

- 클래스 변수
- 모듈과 소스 코드 분리
- 캐릭터 컨트롤러
- 상태 머신
- IDLE 상태 구현



1000명 선수
리소스 로딩 최적화



문제점은?

```
class Boy:

    def __init__(self):
        self.x, self.y = random.randint(100, 700), 90
        self.frame = random.randint(0, 7)
        self.image = load_image('run_animation.png')
```

객체의 멤버변수는 객체마다 따로 만들어진다!

1000번의 로딩이 반복



```
class Boy:
    image = None

    def __init__(self):
        self.x, self.y = random.randint(100, 700), 90
        self.frame = random.randint(0, 7)
        if Boy.image == None:
            Boy.image = load_image('run_animation.png')
```



클래스 변수

클래스 자체에 할당되는 변수.
객체들은 공유하는 동일한 변수를 갖게 됨.

```
class Boy:  
    image = None  
  
...  
    def __do_some():  
  
...  
    Boy.image = ...
```



```
class Boy:
    image = None

    def __init__(self):
        self.x, self.y = random.randint(100, 700), 90
        self.frame = random.randint(0, 7)
        if Boy.image == None:
            Boy.image = load_image('run_animation.png')
```

단 한번의 이미지 로딩만 수행.
이미지 리소스를 모든 객체가 공유하게 됨.

Python Module

- 파이썬의 정의(definitions)와 문장(statements)을 담고 있는 파일
- 파일이름: 00000.py (확장자:py)

```
class Grass:
    pass

def update():
    global x
    x = x + 1

x = 0
update()
```

클래스 정의

함수 정의

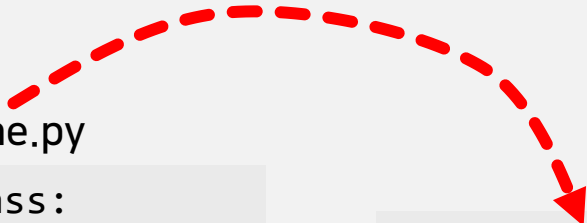
문장

- 그 자체로도 실행 가능하며, 다른 모듈에서 импорт(import)해서 사용할 수도 있음. импорт되면, 그 자체가 하나의 객체가 됨.(싱글톤 객체가 됨)

module 의 사용: 임포트한 후, 모듈이름.0000

game.py

```
class Grass:  
    pass  
  
def update():  
    global x  
    x = x + 1  
  
x = 0  
update()
```



```
import game  
  
grass = game.Grass()  
  
game.update()
```

boy.py 로 분리

```
from pico2d import load_image

class Boy:
    def __init__(self):
        self.x, self.y = 400, 90
        self.frame = 0
        self.action = 3
        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, self.action * 100, 100, 100, self.x, self.y)
```

정리된 후 Source Code Files

- **main.py**
- **boy.py**
- **grass.py**

캐릭터 컨트롤러(Character Controller)

- 게임 주인공의 행동을 구현한 것!
 - 키입력에 따른 액션
 - 주변 객체와의 인터랙션
- 게임 구현에서 가장 핵심적인 부분임.



우리의 “주인공”은?

■ 캐릭터 컨트롤러의 행위를 적으면...

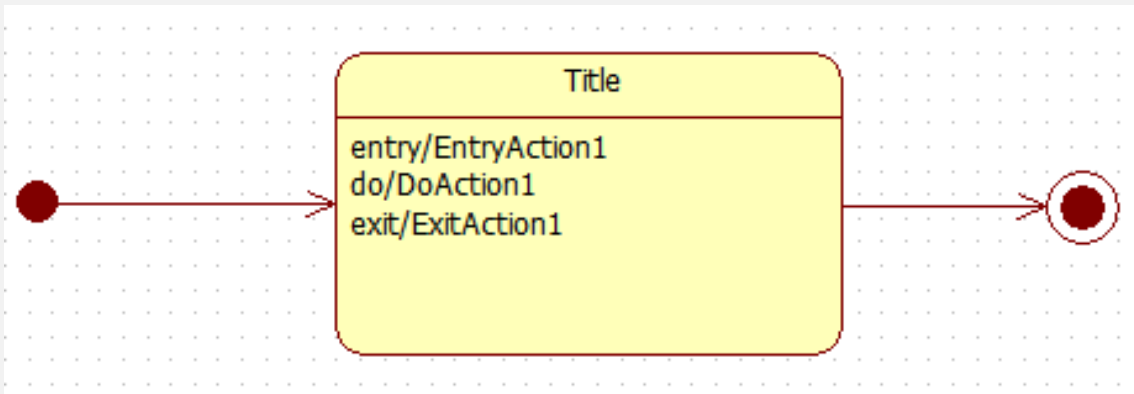
- 처음 소년의 상태는 제자리에 서서 휴식을 하고 있습니다.
- 이 상태에서 오른쪽 방향키를 누르면 소년은 오른쪽으로 달리게 됩니다.
- 방향키를 계속 누르고 있으면, 소년도 계속 오른쪽으로 달리죠.
- 방향키에서 손가락을 떼면 소년은 달리기를 멈추고 휴식상태에 들어갑니다.
- 한참 지나도, 방향키 입력이 없으면 소년은 취침에 들어갑니다.
- 달리는 중에, Dash 키를 누르면 빠르게 달립니다.
- 왼쪽 방향키 조작에 대해선 왼쪽으로 달리게 됩니다.
- 캔버스의 좌우측 가장자리에 도착하면 더 이상 달려나가지는 않습니다.

상태 다이어그램(State Diagram)

- 시스템의 변화를 모델링하는 다이어그램.
- 사건이나 시간에 따라 시스템 내의 객체들이 자신의 상태(state)를 바꾸는 과정을 모델링함.
- 모델링, 명세, 그리고 구현에 모두 사용되는 강력한 툴
- 상태(state)의 변화 예
 - 스위치를 누를 때마다 탁상 전등 상태는 “켜짐”에서 “꺼짐”으로 바뀐다.
 - 리모트 컨트롤의 버튼을 누르면 TV의 상태는 한 채널을 보여주다가 다른 상태를 보여주게 된다.
 - 얼마간의 시간이 흐르면 세탁기의 상태는 “세탁”에서 “헹굼”으로 바뀐다.

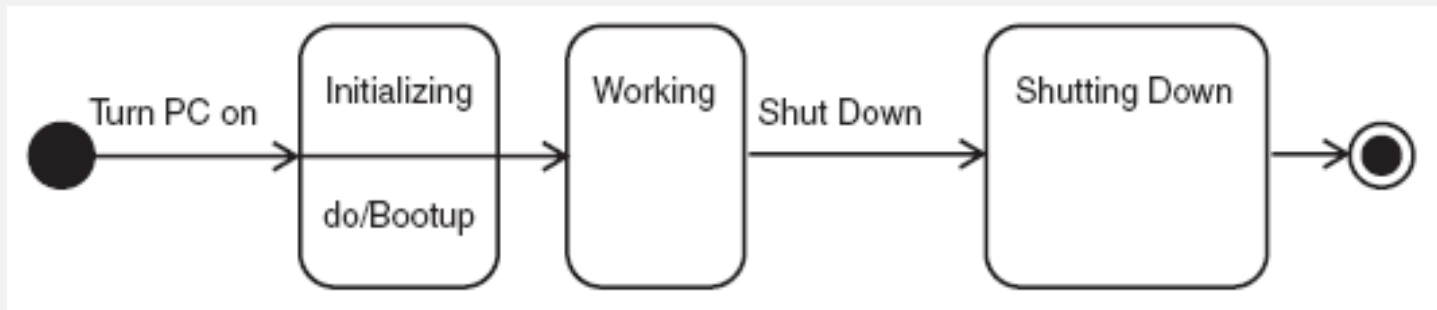
상태(State)

- 상태 : 어떤 조건을 만족하는 동안 머무르면서, 정해진 일을 수행하고 이벤트를 기다리는 “상황”
- Entry action : 특정한 상태로 들어갈 때마다 발생하는 일
- Exit action : 특정한 상태에서 나갈 때마다 발생하는 일
- Do activity : 특정 상태에 머무르는 동안 수행하는 일(반복될 수 있음)



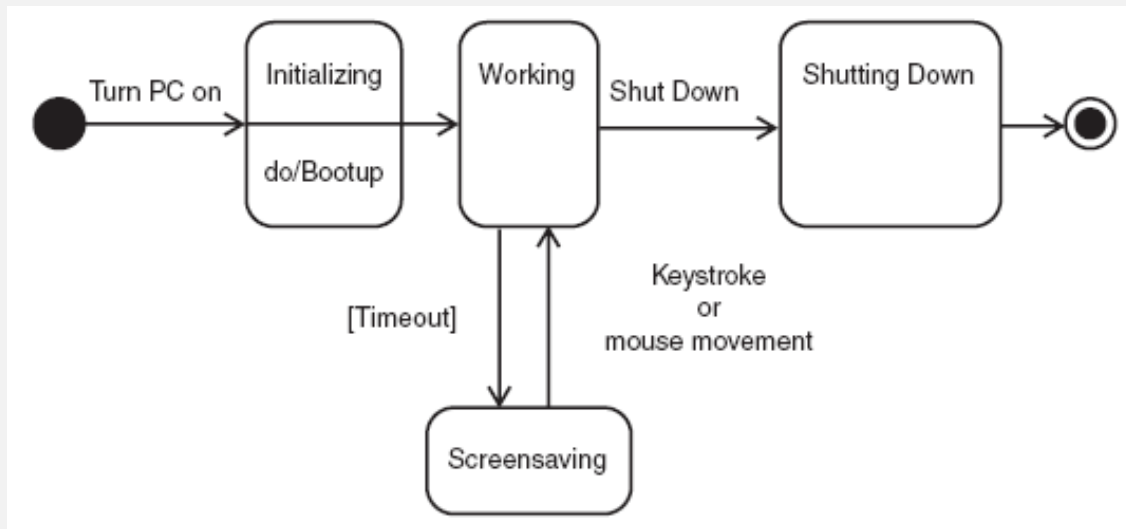
상태 변화(State Transition)

- A transition is a relationship between two states; it indicates that an object in the first state will perform certain actions, then enter the second state when a given event occurs.



이벤트(Event)

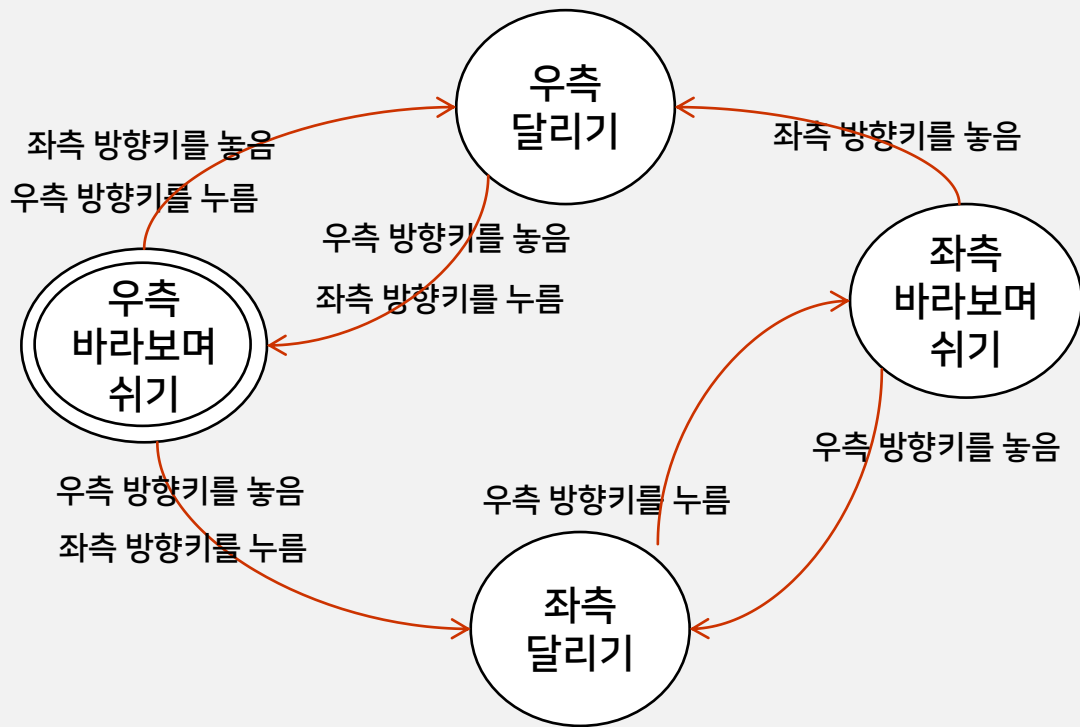
- 상태 변화(State Transition)을 일으키는 원인이 되는 일
 - 외부적인 이벤트 : 예) 키보드 입력
 - 내부적인 이벤트 : 예) 타이머
 - 경우에 따라서는 이벤트 없이도 상태 변화가 있을 수 있음.



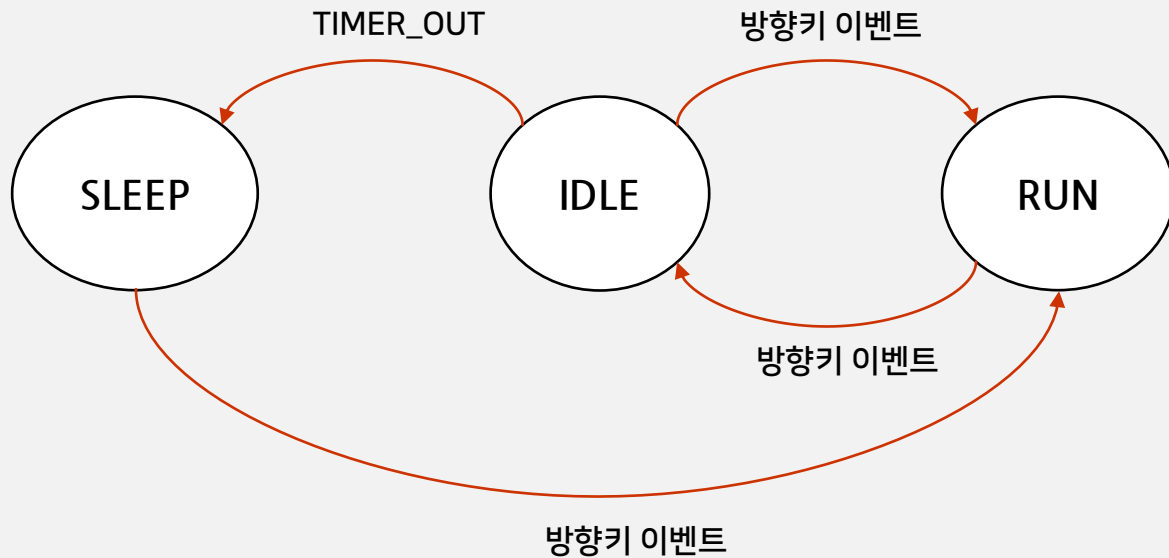
상태와 이벤트 찾기

- 주인공의 상태를 찾아보자.
- 주인공의 상태에 변화를 일으킬 수 있는 이벤트를 찾아보자.

상태 다이어그램 #1



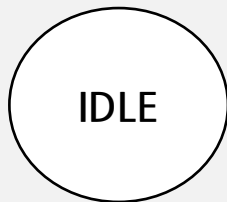
상태 다이어그램 #2





캐릭터 컨트롤러
구현(IDLE)

상태 변환 구현 목표



#1. 상태의 정의 – boy.py



```
class Idle:

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

    def enter(self):
        pass

    def exit(self):
        pass

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

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

#2. 상태 머신 구현 – state_machine.py



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

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

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

#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.IDLE = Idle(self)
        self.state_machine = StateMachine(self.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()
```