

Lecture #9. 게임 오브젝트

2D 게임 프로그래밍

이대현 교수



한국공학대학교
TECH UNIVERSITY OF KOREA

학습 내용

- 게임 오브젝트
- 게임 루프

2D 게임?

- 게임이란?

- “가상 월드에 존재하는 여러 객체들의 상호작용”을 시뮬레이션하고 그 결과를 보여주는(렌더링) 것.

- 2D 게임?

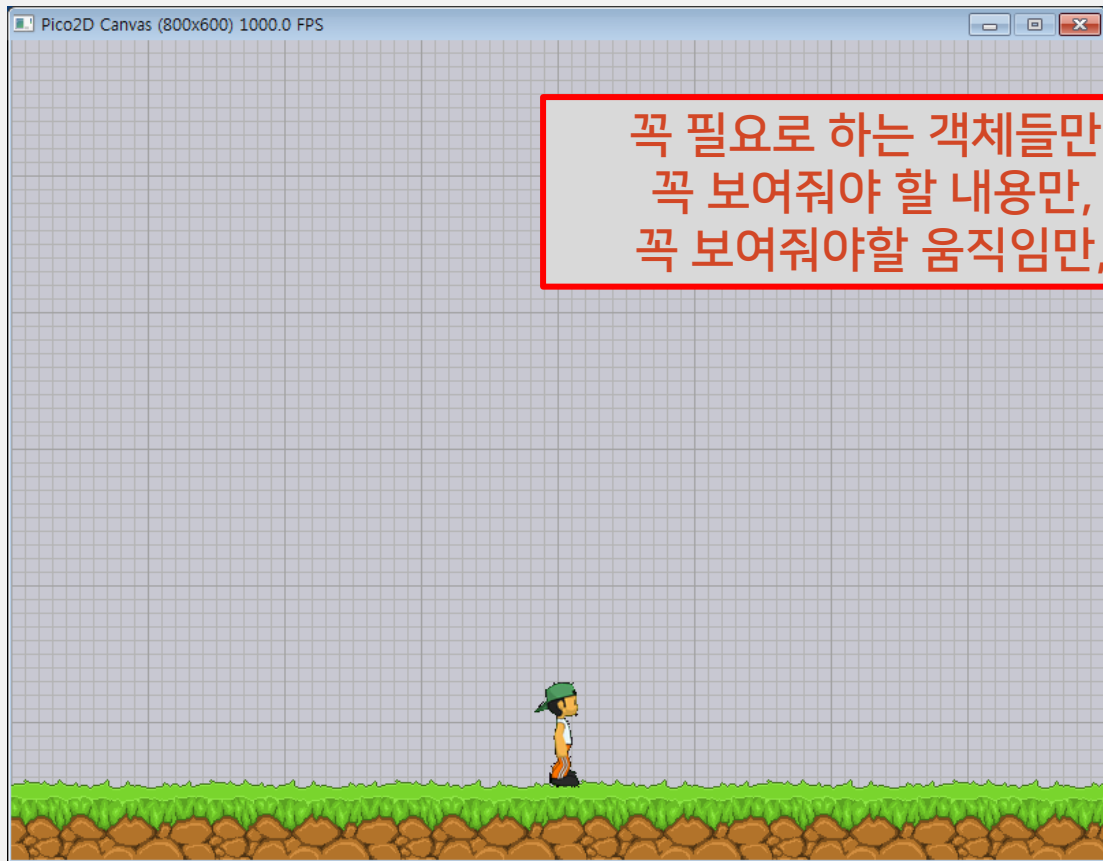
- 현재 진행 중인 게임 가상 월드의 내용을 화면에 2D 그림으로 보여주는 것

실제 세계

어떤 객체(Object)들이 있는가?
몇 개의 객체가 있는가?

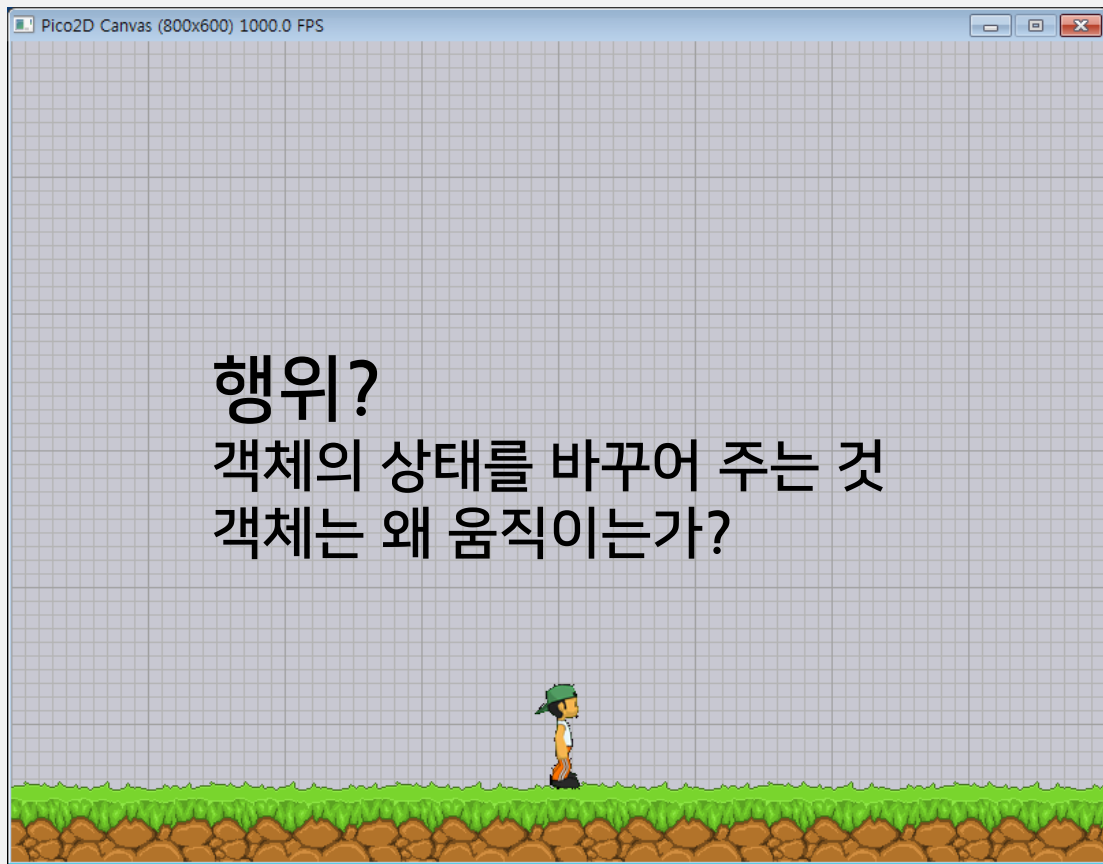


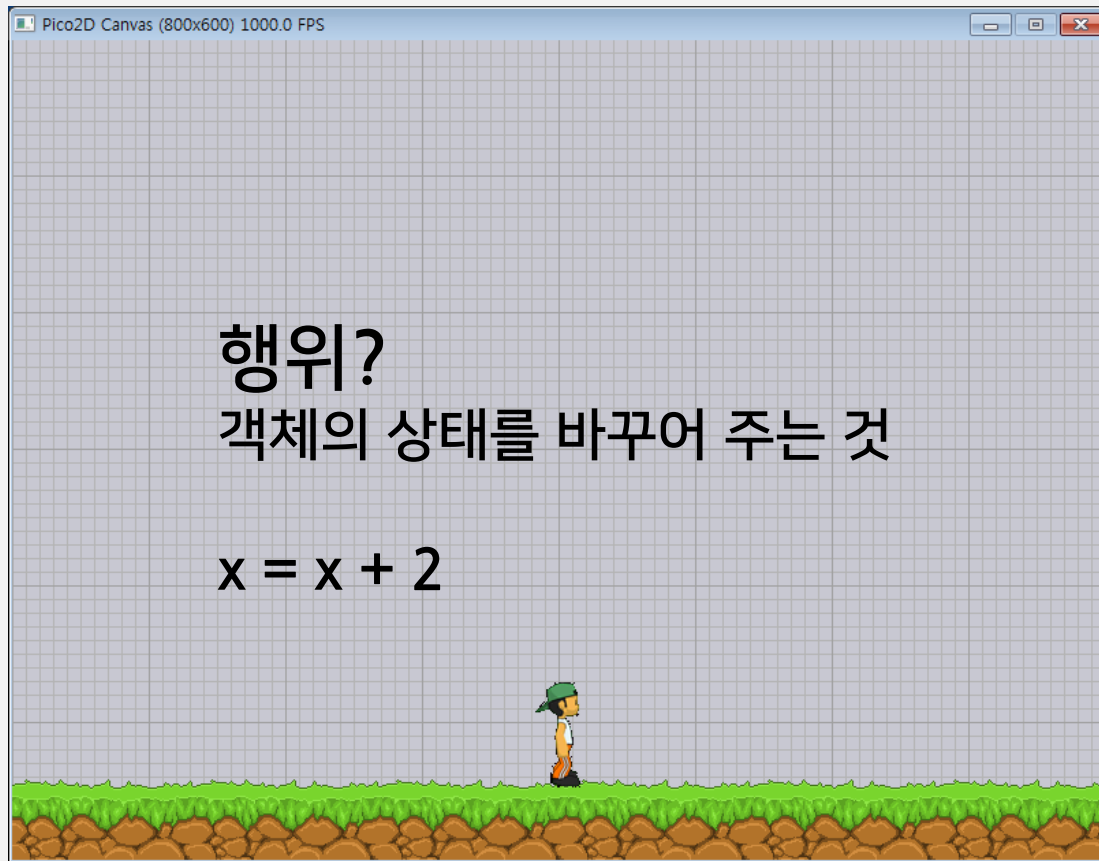
추상화(Abstraction)













속성 + 행위 = 소년 객체

게임 객체(Game Object)

- 게임 객체(Game Object)

- 게임 월드를 구성하는 모든 요소들을 지칭

- 게임 객체의 본질: 속성과 행동의 모음

- 속성 - 게임 객체의 현재 상태
- 행동 - 시간에 따라, 혹은 이벤트에 반응해서 상태가 변하는 방식

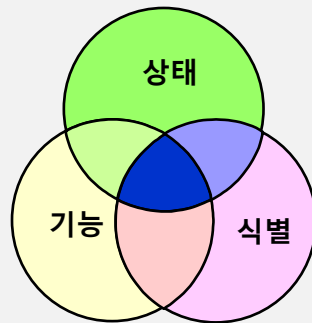
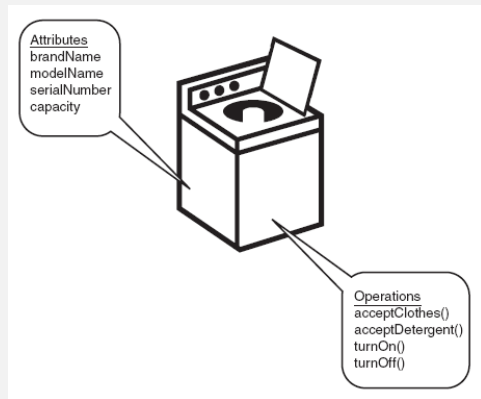
객체(Object)

- 문제 영역의 실세계에 존재하는 구체적인 대상을 모델링한 것.
 - 게임 - 지형, 캐릭터, 몬스터, 우주선, 보스, ...
 - 그래픽 라이브러리 - 점, 선, 사각형, 원, 윈도우, ...
 - 학사 관리 소프트웨어 - 사람, 학생, 직원, 시간제 직원, ...
 - 문서 편집기 프로그램 - 글자, 단어, 문장, 문단, 문서, 글씨체, ...

객체의 정의

■ 모델링 방법

- 데이터와 그 데이터 위에 수행되는 함수들을 가진 소프트웨어 모듈을 이용.
- 데이터는 객체의 상태(State, Attributes)를 저장하는 데 사용
- 함수는 그 객체가 수행하는 기능(Behavior, Operations, Methods)을 정의
- 객체의 조건
 - (State + Behavior) with Unique Identity



클래스(Class)

■ 클래스란?

- 유사한 여러 객체들에게 공통적으로 필요로 하는 데이터와 이 데이터 위에서 수행되는 함수들을 정의하는 소프트웨어 단위.
- 객체를 찍어내는 “도장”

클래스에 비유할 수 있는 것들		
	붕어빵 틀	제품 설계도
객체에 비유할 수 있는 것들		
	붕어빵	제품

객체 생성

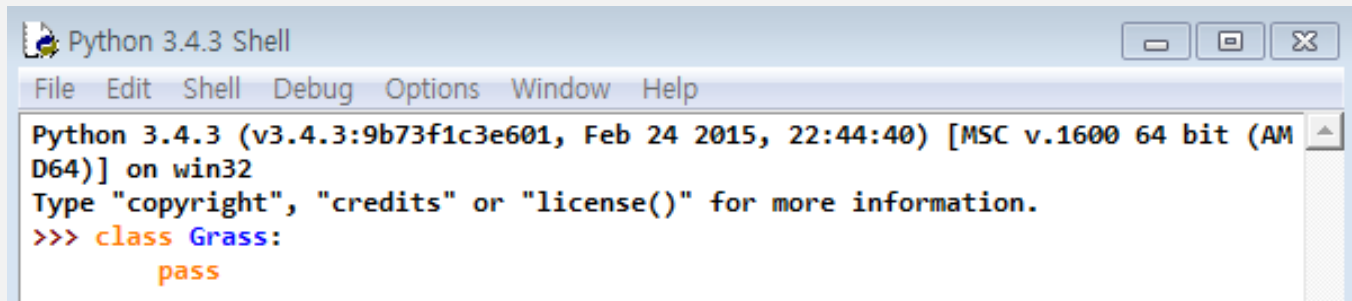
- 객체를 생성하려면?

- 클래스라는 틀을 이용하여 붕어빵 찍어내듯이 객체를 생성하게 됨.
- 찍어내는 과정을 Object Instantiation 이라고 함.

- 인스턴스(Instance)

- 생성된 각각의 객체
- 모든 객체는 어떤 클래스로부터 생성된 인스턴스.

잔디 클래스 만들기



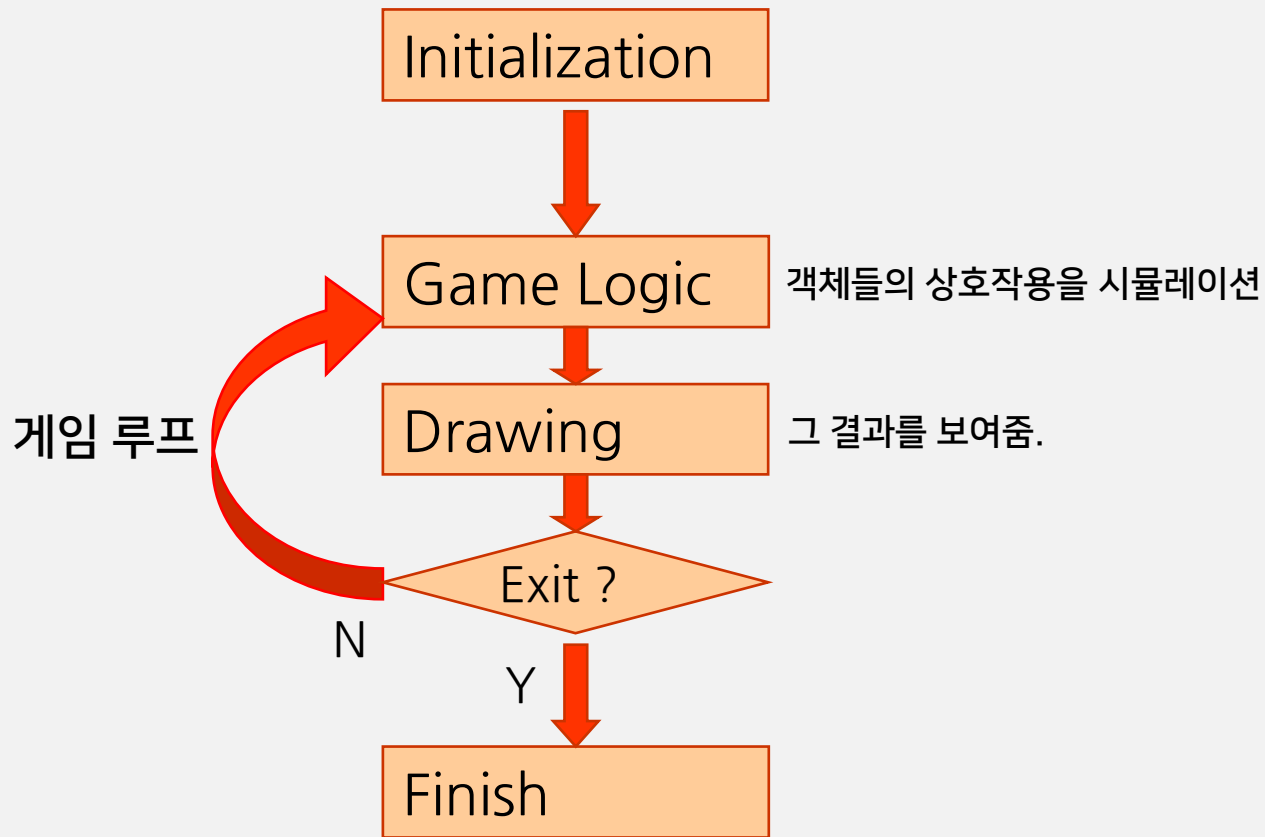
```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:44:40) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> class Grass:
    pass
```


grass = Grass()



게임 오브젝트

게임 기본 구조



boy_grass_object.py



```
from pico2d import *

# 게임 오브젝트 클래스의 정의를 여기에

def handle_events():
    ... 생략 ...

open_canvas()

# 초기화 코드

# 게임 루프 코드

# 종료 코드

close_canvas()
```

```
def reset_world():  
    global running  
  
    running = True  
  
def update_world():  
    pass  
  
def render_world():  
    clear_canvas()  
    update_canvas()
```

```
open_canvas()  
  
reset_world()  
  
while running:  
    handle_events()  
    update_world()  
    render_world()  
    delay(0.05)  
  
close_canvas()
```



잔디 클래스

생성자(Constructor) 함수 `__init__`
객체가 생성될 때, 맨처음 자동 호출되는 함수
객체의 초기 상태를 설정 - 속성(멤버변수) 설정

```
class Grass:
    def __init__(self):
        self.image = load_image('grass.png')
```

`self` : 생성된 객체를 가리키는 더미 변수

```
    def draw(self):
        self.image.draw(400, 30)
```

멤버 함수는 항상 첫번째 인자가 `self` 이어야 함.

```
    def update(self): pass
```

잔디 객체 생성

```
def reset_world():  
    global running  
    global grass  
  
    running = True  
    grass = Grass()
```

잔디 객체 업데이트와 렌더링

```
def update_world():  
    grass.update()  
    pass
```

```
def render_world():  
    clear_canvas()  
    grass.draw()  
    update_canvas()
```


소년 클래스

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

    def update(self):
        self.frame = (self.frame + 1) % 8
        self.x += 5

    def draw(self):
        self.image.clip_draw(self.frame*100, 0, 100, 100, self.x, self.y)
```

완성된 코드

```
def reset_world():
    global running
    global grass
    global boy

    running = True
    grass = Grass()
    boy = Boy()

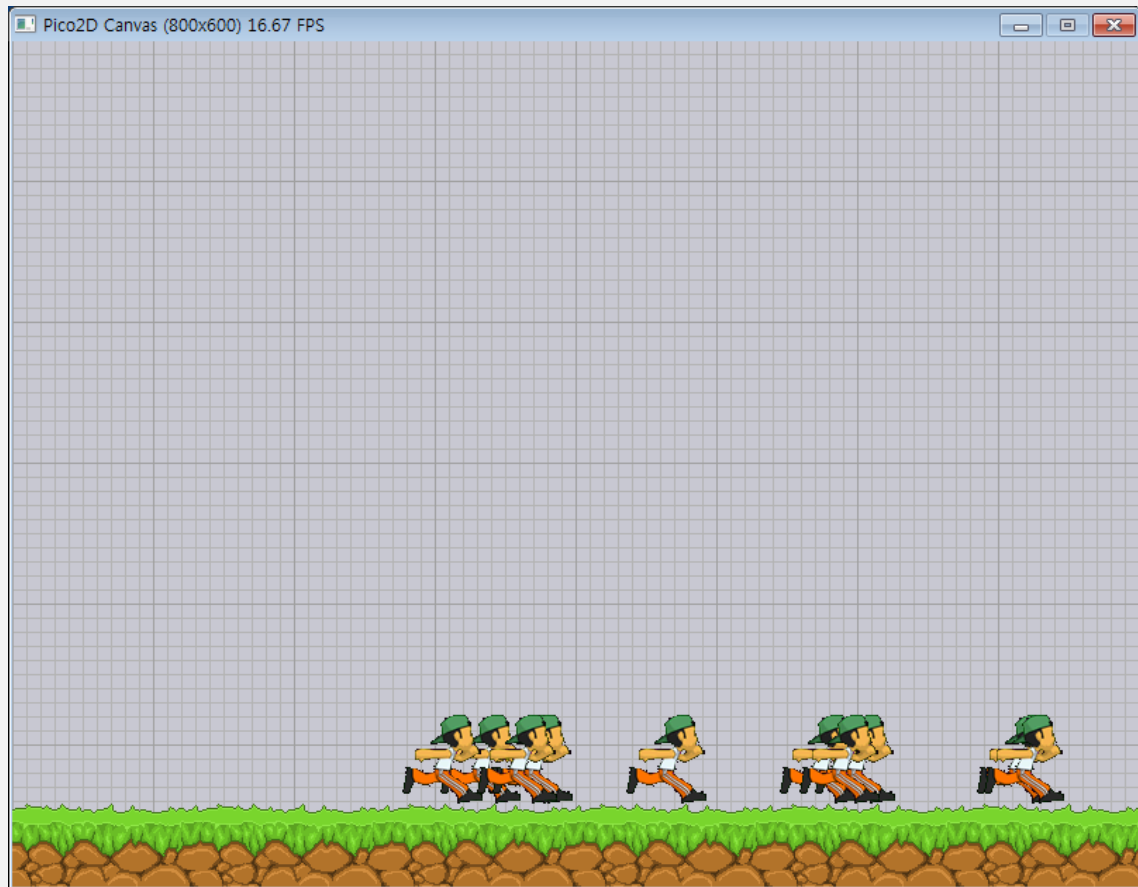
def update_world():
    grass.update()
    boy.update()
    pass

def render_world():
    clear_canvas()
    grass.draw()
    boy.draw()
    update_canvas()
```



소년 축구단 만들기

소년 축구단을 만들어보자~



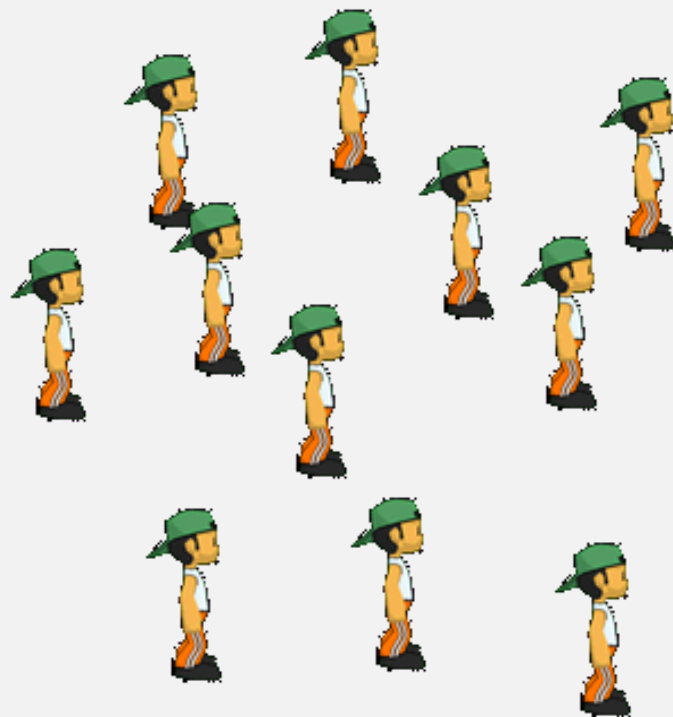
11명의 소년 만들기(생성)!

boy = Boy()



```
boy1 = Boy()  
boy2 = Boy()  
boy3 = Boy()  
boy4 = Boy()  
boy5 = Boy()  
boy6 = Boy()  
boy7 = Boy()  
boy8 = Boy()  
boy9 = Boy()  
boy10 = Boy()  
boy11 = Boy()
```

팀(team) 만들기 ?



```
team = [boy1, boy2, boy3, boy4, boy5,  
        boy6, boy7, boy8, boy9, boy10, boy11]
```



```
team = [Boy()] * 11
```



List Comprehension의 이용

```
team = [Boy() for i in range(11)]
```

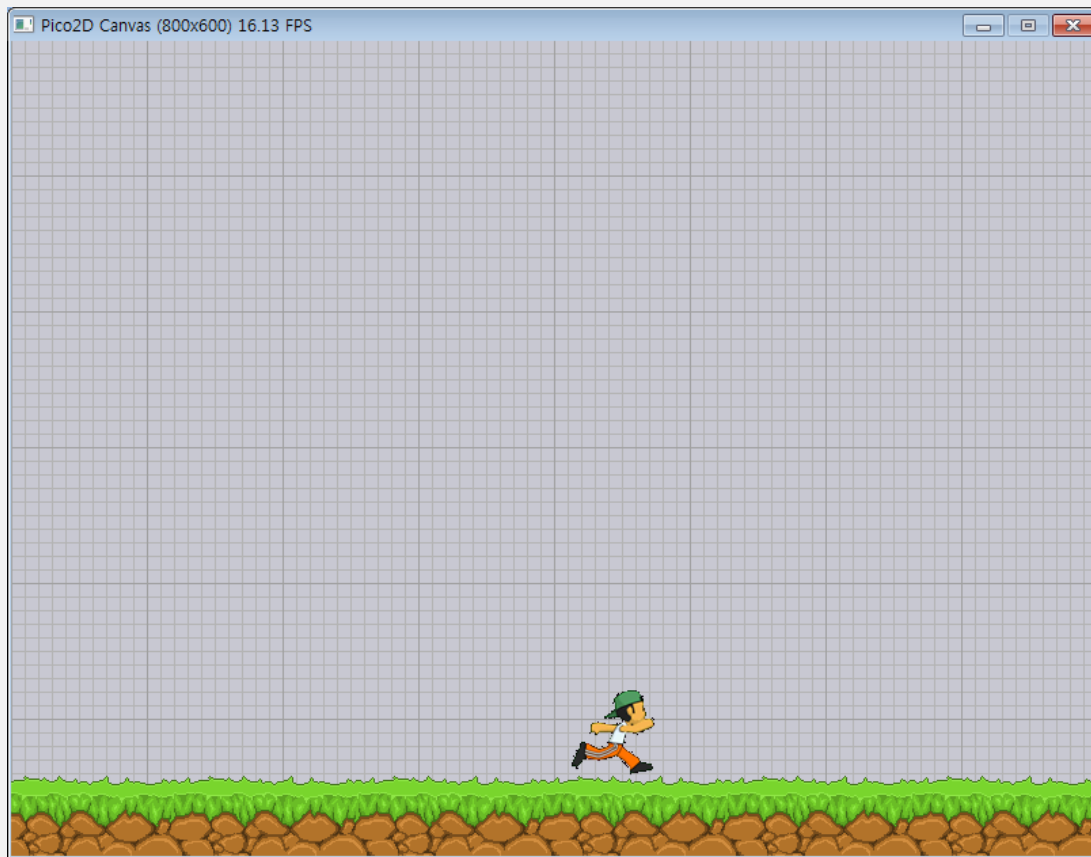


팀 생성, 업데이트, 렌더링

```
def reset_world():  
    global running  
    global grass  
    global team  
  
    running = True  
    grass = Grass()  
    team = [Boy() for i in range(10)]
```

```
def update_world():  
    grass.update()  
    for boy in team:  
        boy.update()  
  
def render_world():  
    clear_canvas()  
    grass.draw()  
    for boy in team:  
        boy.draw()  
    update_canvas()
```

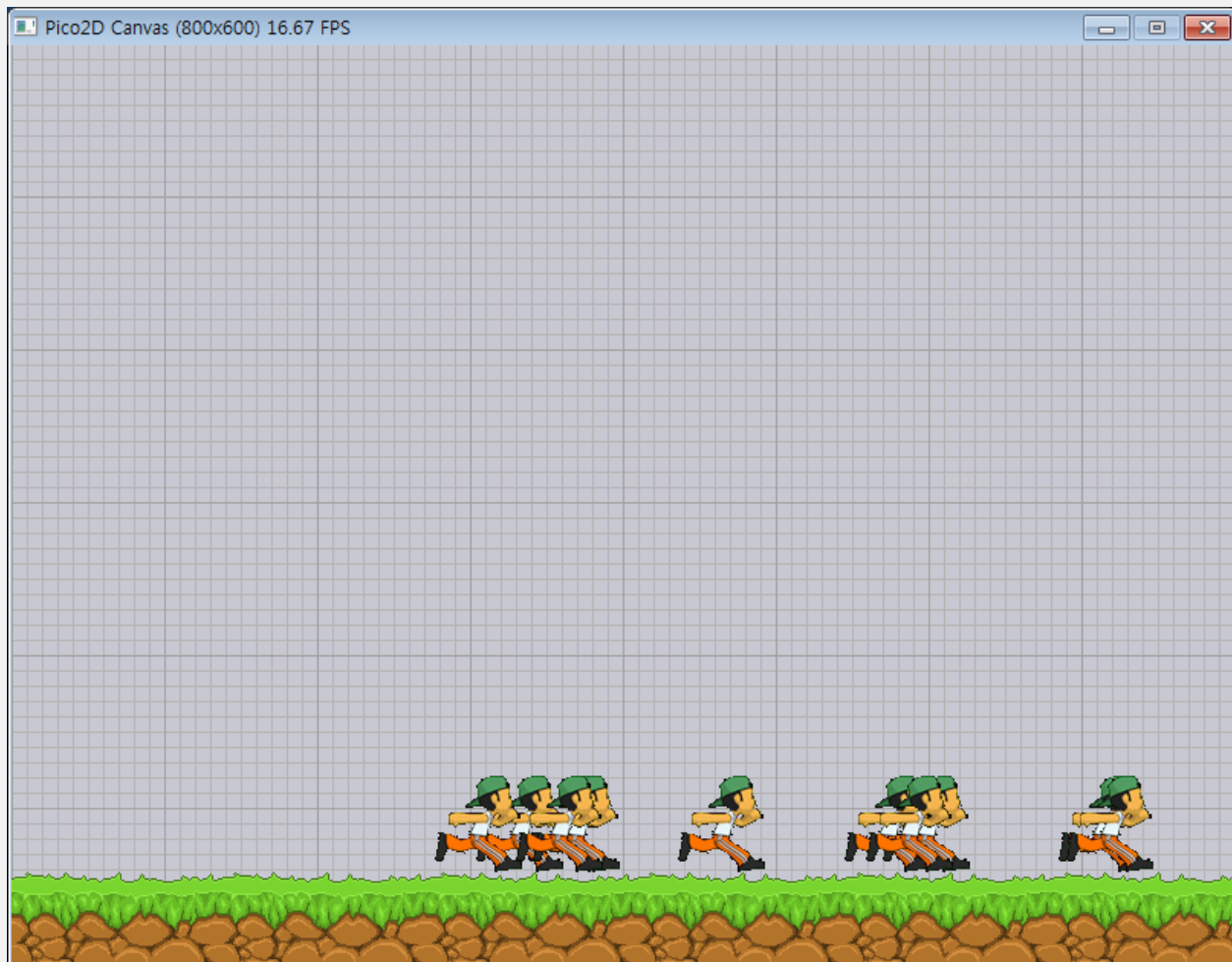
잉? 왜 한명만????



소년별로 시작 위치를 다르게...

```
import random
```

```
self.x, self.y = random.randint(100, 700), 90
```



애니메이션 싱크가 안되게...

```
self.frame = random.randint(0, 7)
```

world list 를 이용한 Refactoring

```
def reset_world():  
    global running  
    global grass  
    global team  
    global world  
  
    running = True  
    world = []  
  
    grass = Grass()  
    world.append(grass)  
  
    team = [Boy() for i in range(10)]  
    world += team
```

```
def update_world():  
    for o in world:  
        o.update()  
  
def render_world():  
    clear_canvas()  
    for o in world:  
        o.draw()  
    update_canvas()
```