

Lecture #4. 2D 렌더링

2D 게임 프로그래밍

이대현 교수



한국공학대학교
TECH UNIVERSITY OF KOREA

학습 내용

- 2D 게임의 정의
- 2D 게임의 기본 요소
- Pico2d 설치
- 캐릭터 이미지의 렌더링과 이동

2D 게임?

■ 게임이란?

- “가상 월드에 존재하는 여러 객체들의 상호작용”

■ 게임의 기본 구성 요소

- 배경
- 캐릭터, 오브젝트
- UI - GUI, 입력(키, 마우스, 터치, ...)
- AI
- 사운드

■ 2D 게임?

- 현재 진행 중인 게임 가상 월드의 내용을 화면에 2D 그림으로 보여주는 것
- 배경, 캐릭터(오브젝트)의 표현(렌더링)을 2D 이미지들의 조합으로 구성함!

2D 게임의 기본 요소



2D 게임 개발 접근법

- 플랫폼 종속적 방법
 - Direct X
 - OpenGL
 - Simple Frame Buffer
- 플랫폼 독립적 방법, Cross Platform
 - Unity
 - Unreal
 - COCOS2D
 - SDL
 - 그 외의 범용 2D 렌더링 라이브러리

SDL(Simple DirectMedia Layer)

■ SDL이란?

- 크로스 플랫폼 멀티미디어 라이브러리.
- 비디오, 오디오 및 사용자 입력을 처리하는 API로 구성.
- 기본적으로 2D 그래픽 라이브러리. 3D는 OpenGL을 통해서 지원.

■ SDL이 지원하는 플랫폼

- PC: Windows, Linux, Mac OS
- Phone: Android, iOS,

■ 라이선싱(SDL 2.0)

- zlib license
- 자유롭게 상용 게임을 개발할 수 있음.
- SDL1.2 → GNU LGPL 라이선싱

■ 홈페이지

- www.libsdl.org



2D 게임 개발 환경 구성

- 필수 환경

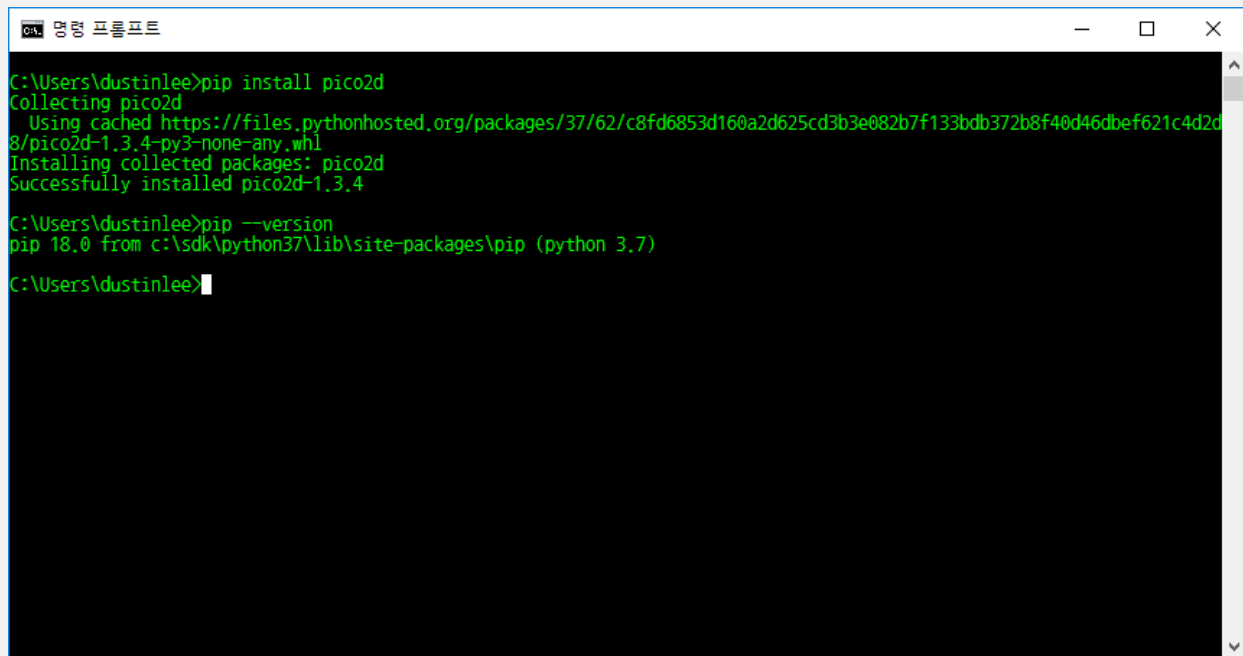
- Windows 10 64 bit
- Python 3.7.0+
- Git / TortoiseGit

- 2D 그래픽 라이브러리

- pico2d - 내부에 SDL라이브러리와 PySDL2 라이브러리를 포함.

pico2d 의 설치 – pip 이용

- cmd 창에서, “pip install pico2d” 를 입력
 - 경우에 따라서, pip 자체를 update 할 필요가 있음.
 - pip가 실행되지 않는 경우는, python을 다시 설치해야 함.

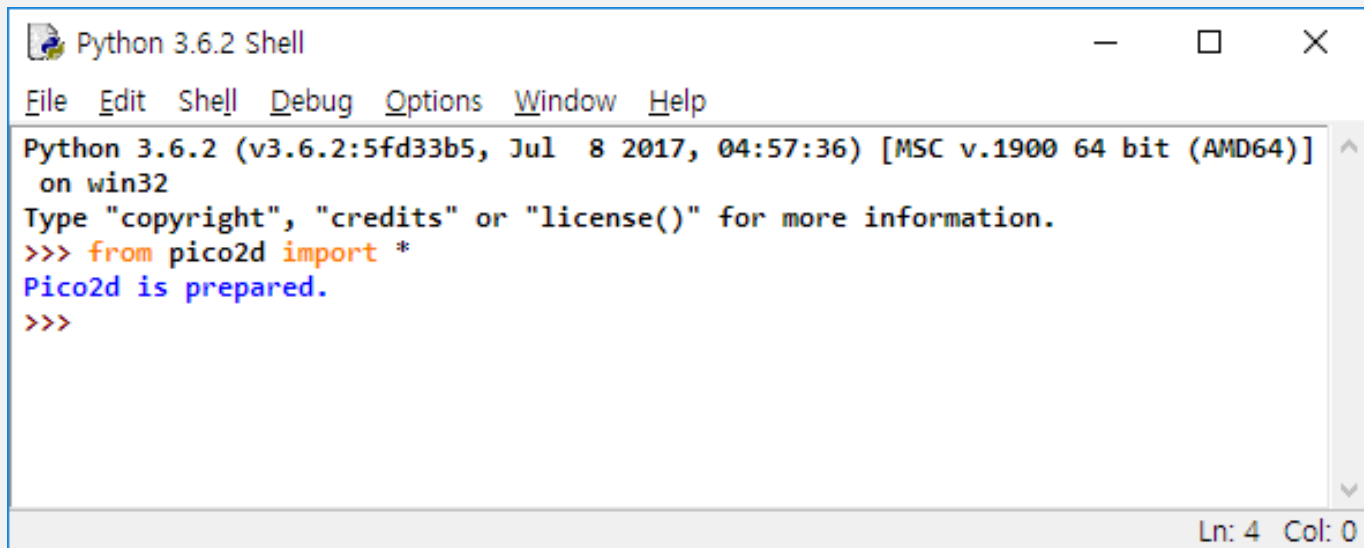


```
C:\Users\dustinlee>pip install pico2d
Collecting pico2d
  Using cached https://files.pythonhosted.org/packages/37/62/c8fd6853d160a2d625cd3b3e082b7f133bdb372b8f40d46dbef621c4d2d8/pico2d-1.3.4-py3-none-any.whl
Installing collected packages: pico2d
Successfully installed pico2d-1.3.4

C:\Users\dustinlee>pip --version
pip 18.0 from c:\sdk\python37\lib\site-packages\pip (python 3.7)

C:\Users\dustinlee>
```


Pico2d 라이브러리 설치 완료 확인

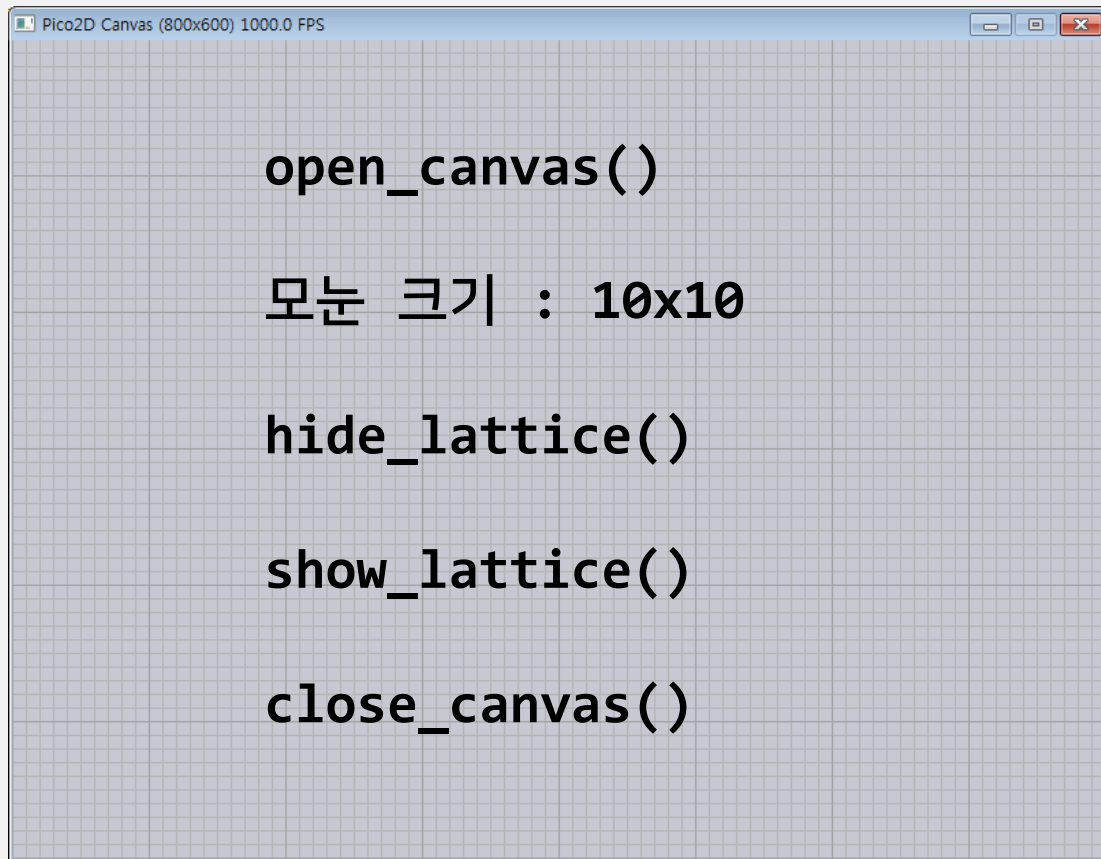


A screenshot of a Python 3.6.2 Shell window. The window title is "Python 3.6.2 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area shows the following output:

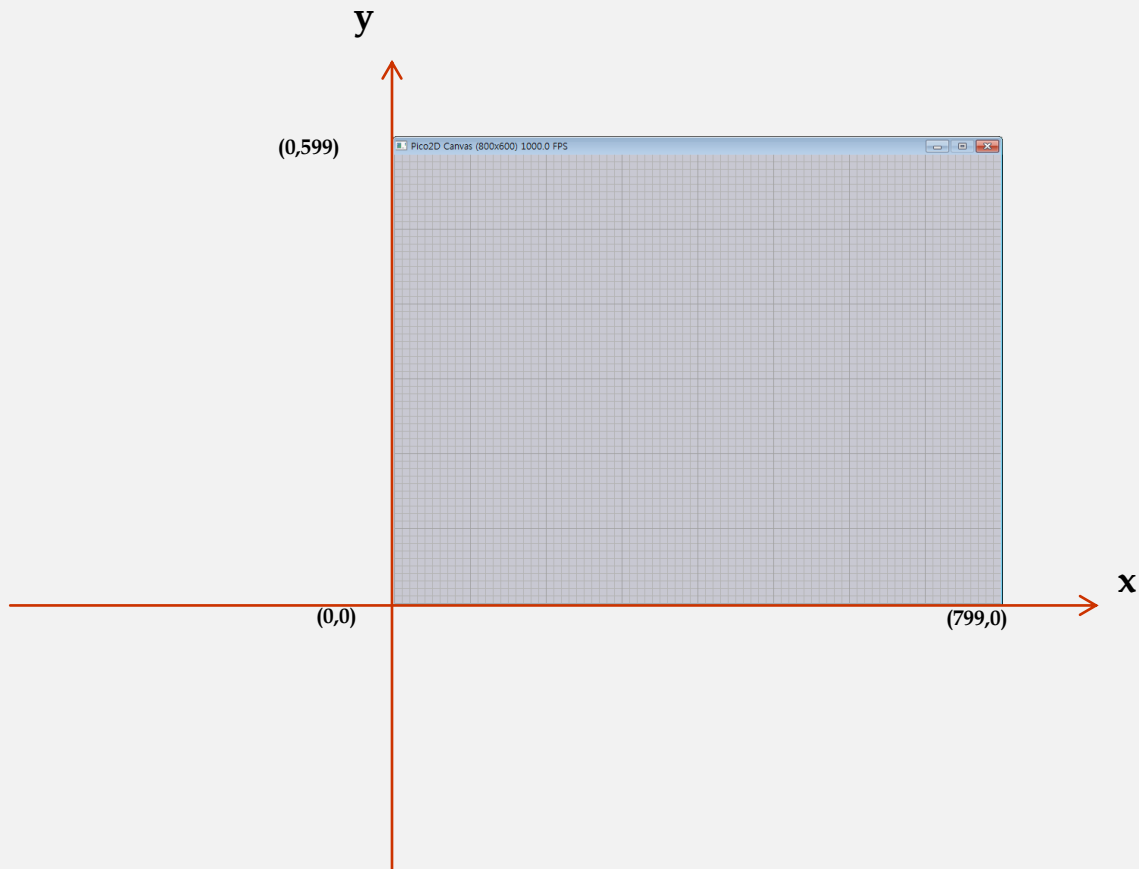
```
Python 3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:57:36) [MSC v.1900 64 bit (AMD64)]  
on win32  
Type "copyright", "credits" or "license()" for more information.  
>>> from pico2d import *  
Pico2d is prepared.  
>>>
```

The status bar at the bottom right indicates "Ln: 4 Col: 0".

캔버스 열기 - open_canvas(800,600)

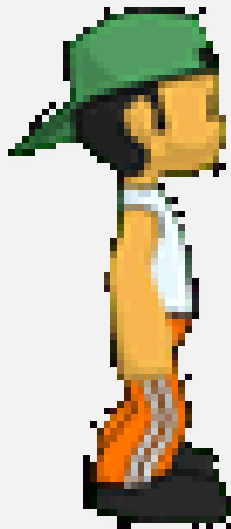


캔버스의 좌표계



JPG vs PNG

우리의 주인공

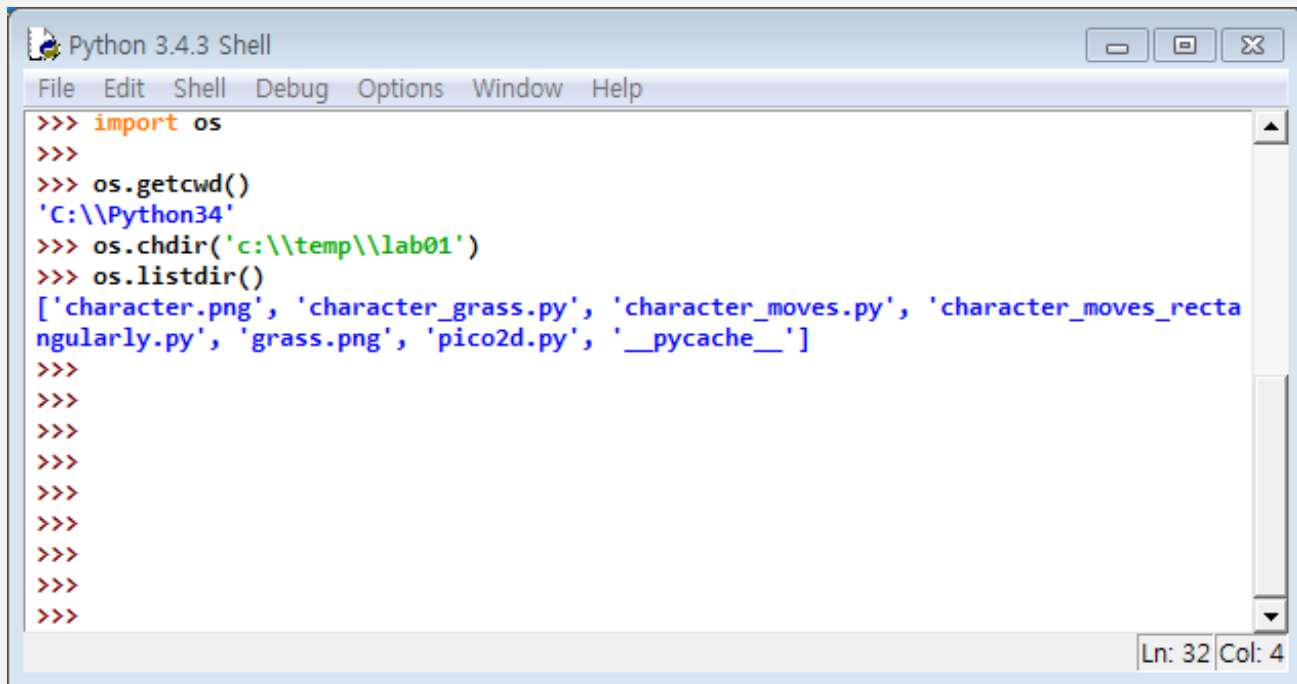


```
>>> image = load_image('character.png')  
>>> image.draw_now(400, 300)
```

Working Directory

- 현재 프로그램의 수행 작업이 이루어지고 있는 폴더를 말함.
- 프로그램에서 파일을 참조할 때 기본 위치(ROOT)로 사용됨.
- 일반적으로 프로그램 실행 파일이 들어있는 그 폴더 자체가 Working Directory 임.

OS 모듈을 이용한 Working Directory 설정

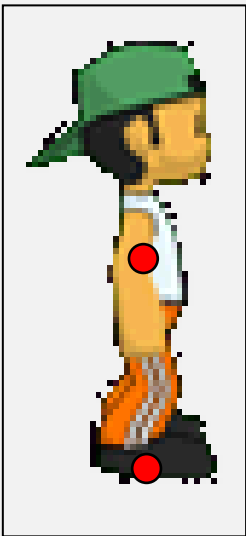


The screenshot shows a Python 3.4.3 Shell window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help) and a command prompt interface. The user has entered several commands to change the working directory and list its contents. The output shows the current directory is 'C:\\Python34' and the new directory is 'C:\\temp\\lab01', which contains several files including character-related images and Python scripts.

```
>>> import os
>>>
>>> os.getcwd()
'C:\\Python34'
>>> os.chdir('c:\\temp\\lab01')
>>> os.listdir()
['character.png', 'character_grass.py', 'character_moves.py', 'character_moves_recta
ngularly.py', 'grass.png', 'pico2d.py', '__pycache__']
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
```

Ln: 32 Col: 4

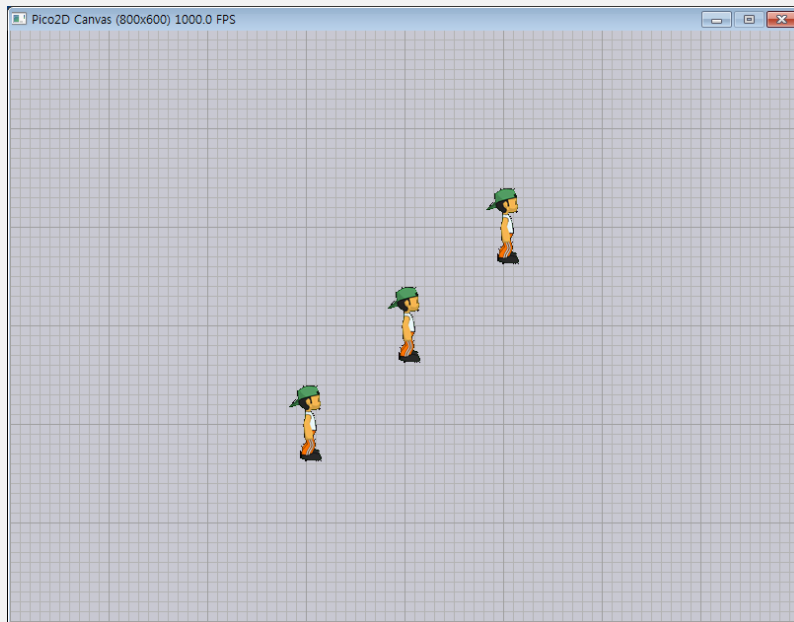
피봇(Pivot)



여기가 피봇입니다.

이 점을 피봇으로 삼기도 합니다

몇 명 더 그려 봅시다~



```
>>> image.draw_now(300,200)
```

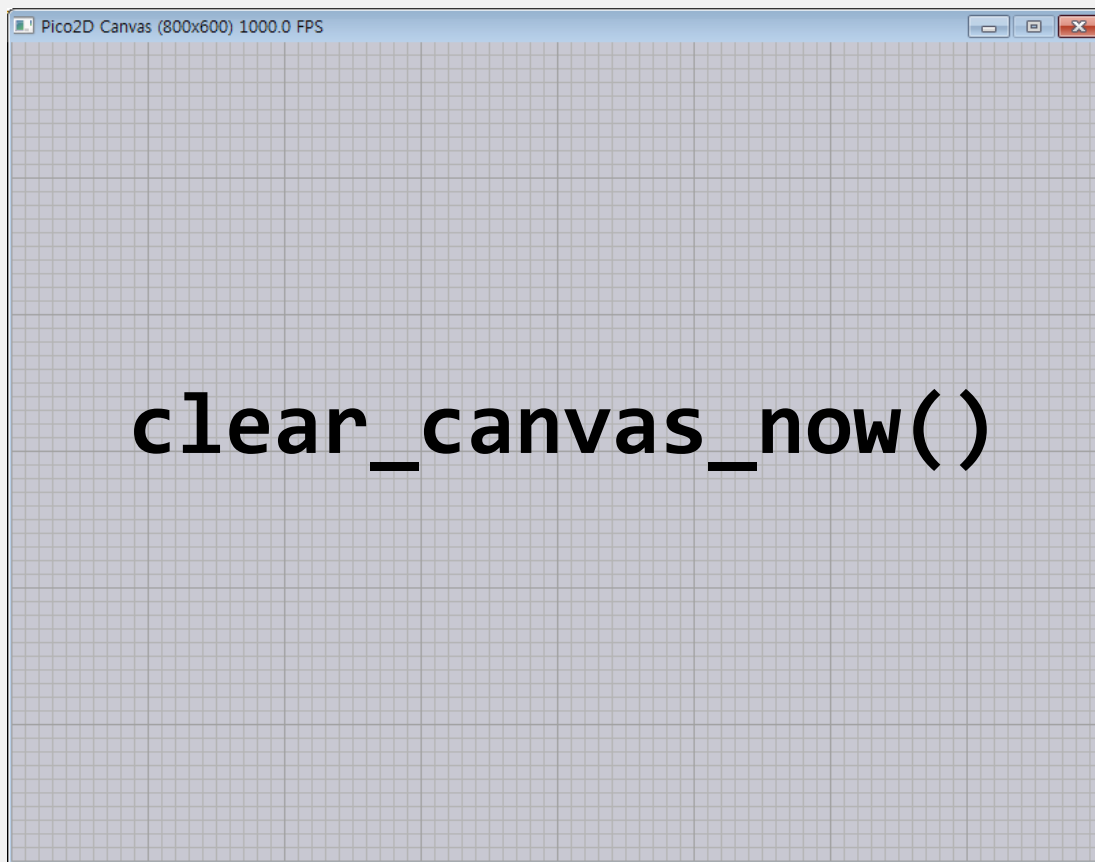
```
>>> image.draw_now(500,400)
```

테로 그리기

```
>>> for x in range(0,9):  
        for y in range (0, 7):  
            img.draw_now(x * 100, y * 100)
```

캐릭터 떼!



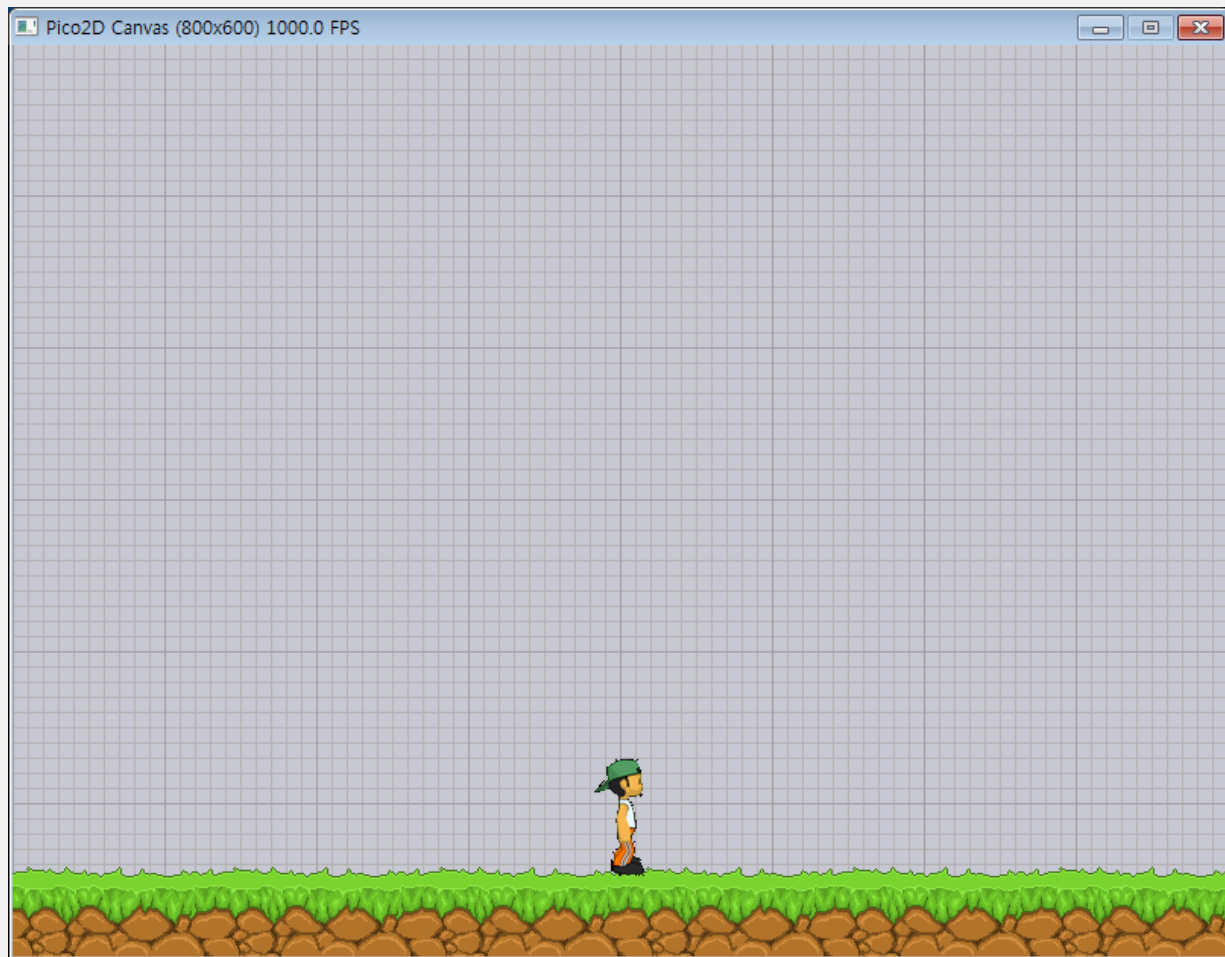




캐릭터 이동



```
from pico2d import *  
  
open_canvas()  
  
grass = load_image('grass.png')  
character = load_image('character.png')  
  
grass.draw_now(400, 30)  
character.draw_now(400, 90)  
  
delay(5)  
  
close_canvas()
```





```
from pico2d import *

open_canvas()

grass = load_image('grass.png')
character = load_image('character.png')

x = 0
while (x < 800):
    clear_canvas_now()
    grass.draw_now(400, 30)
    character.draw_now(x, 90)
    x = x + 2
    delay(0.01)

close_canvas()
```


게임 루프

```
x = 0
```

```
while (x < 800):
```

```
    clear_canvas_now()  
    grass.draw_now(400, 30)  
    character.draw_now(x, 90)
```

Game Rendering

```
    x = x + 2
```

Game Logic

```
    delay(0.01)
```