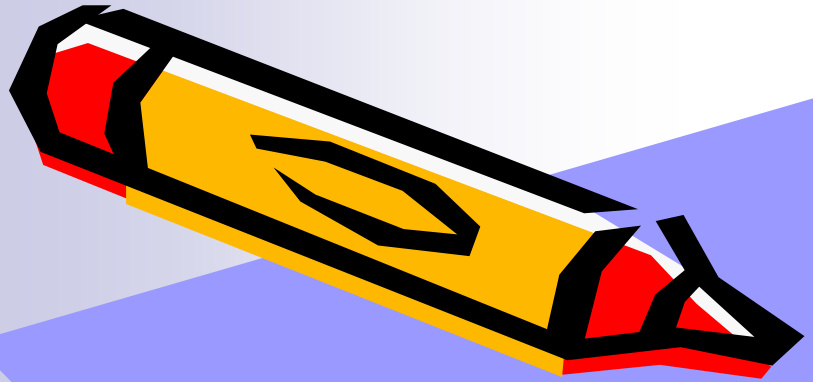


2D 게임 프로그래밍

제12강 타일링

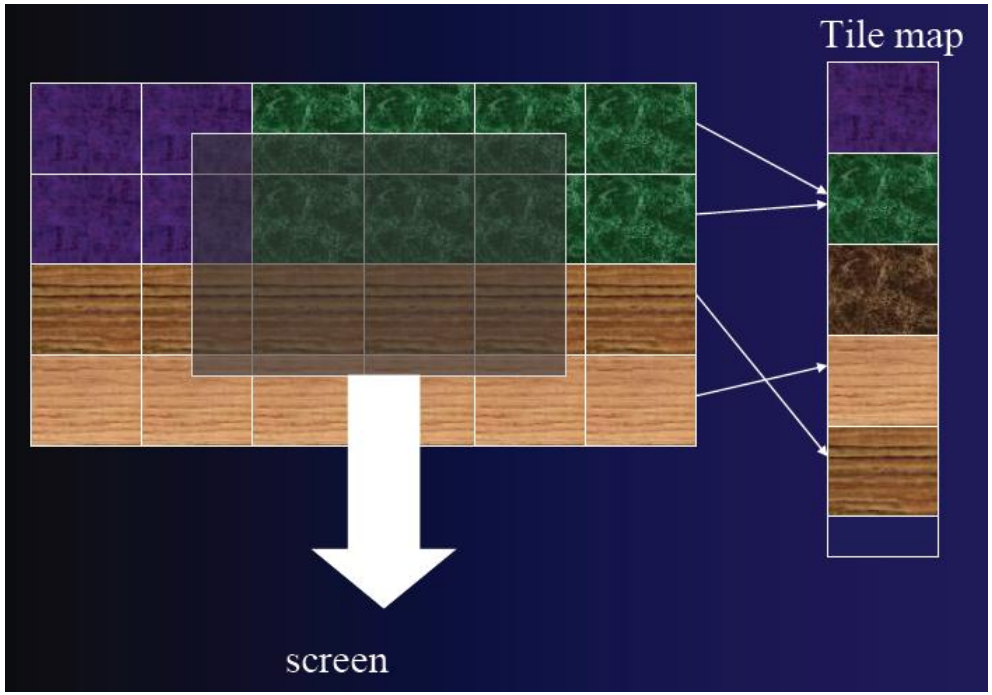
이대현
한국산업기술대학교

학습 내용

- 타일링
- 타일셋 구현
- 타일맵 렌더링

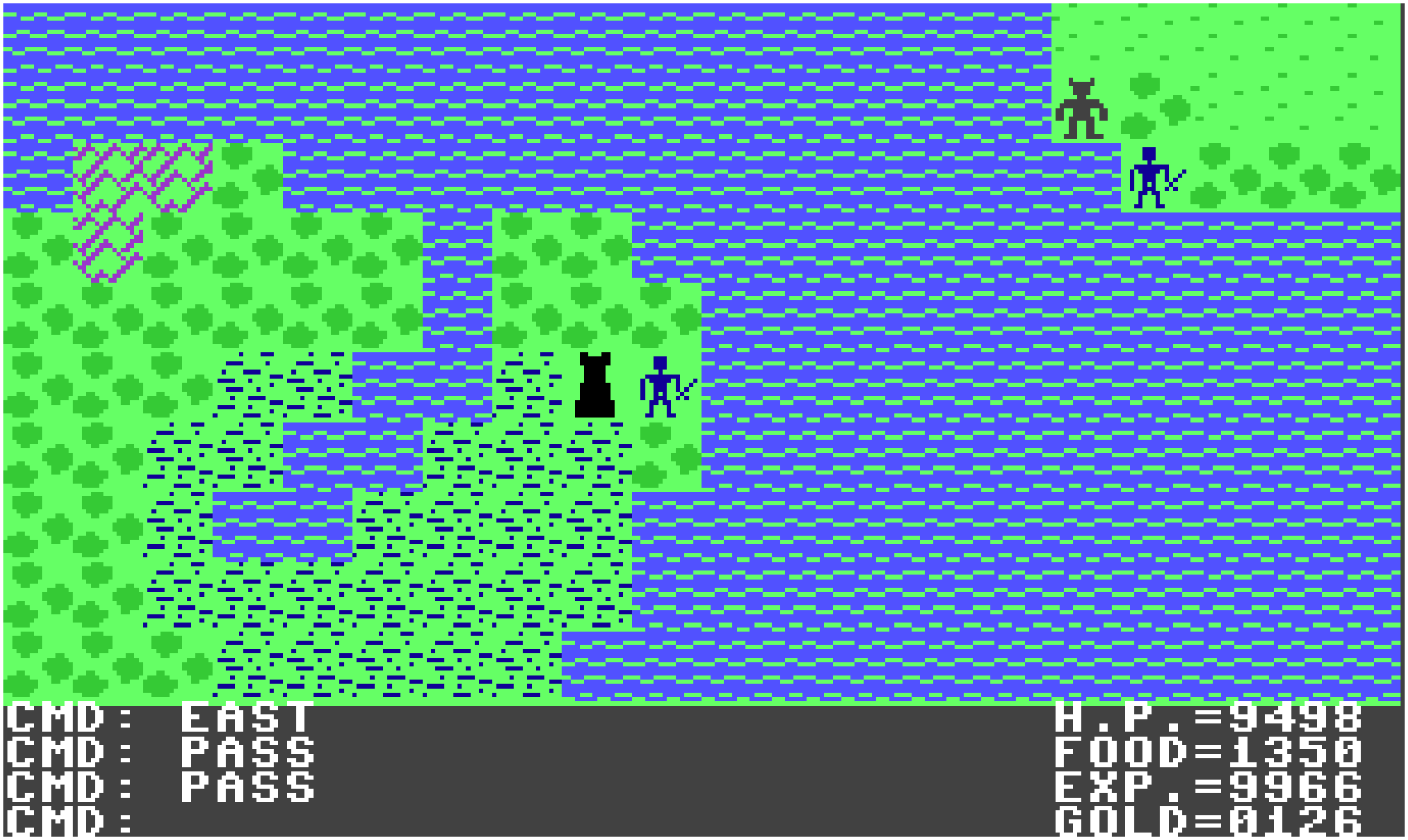
타일링(Tiling)

- 게임의 배경을 단일 이미지로 구성하지 않고, 여러 개의 타일을 이용하여 구성하는 방법. 거의 모든 RPG 게임에 사용됨.

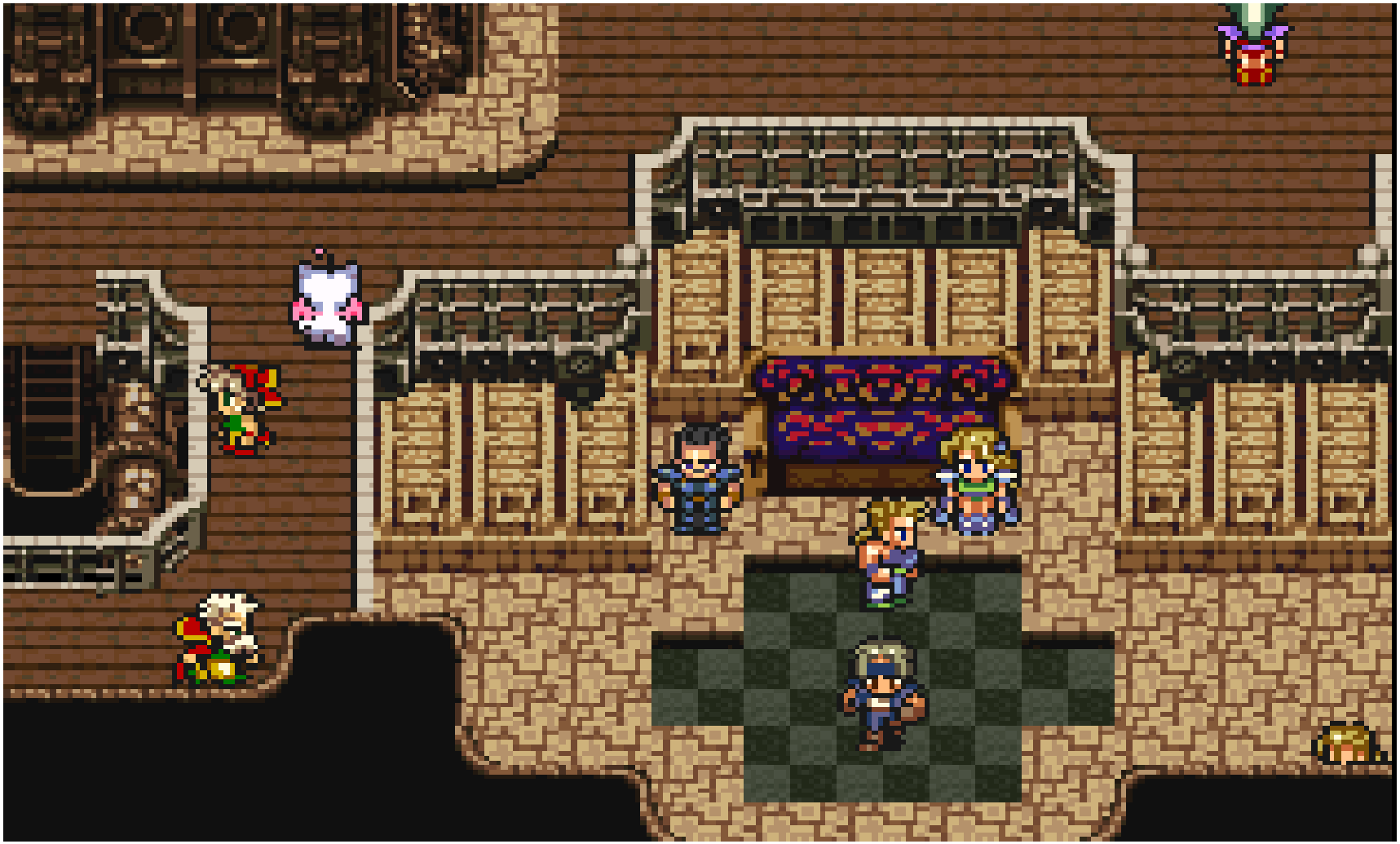


- 장점
  - 맵의 크기를 대폭 줄일 수 있다.
  - 사용자의 구미에 맞게 맵을 편집하여 사용할 수 있다.
- 단점
  - 정교한 배경을 만드는데 많은 시간이 소요되고 어렵다.

타일 기반 게임의 진화: Ultima2(1982)



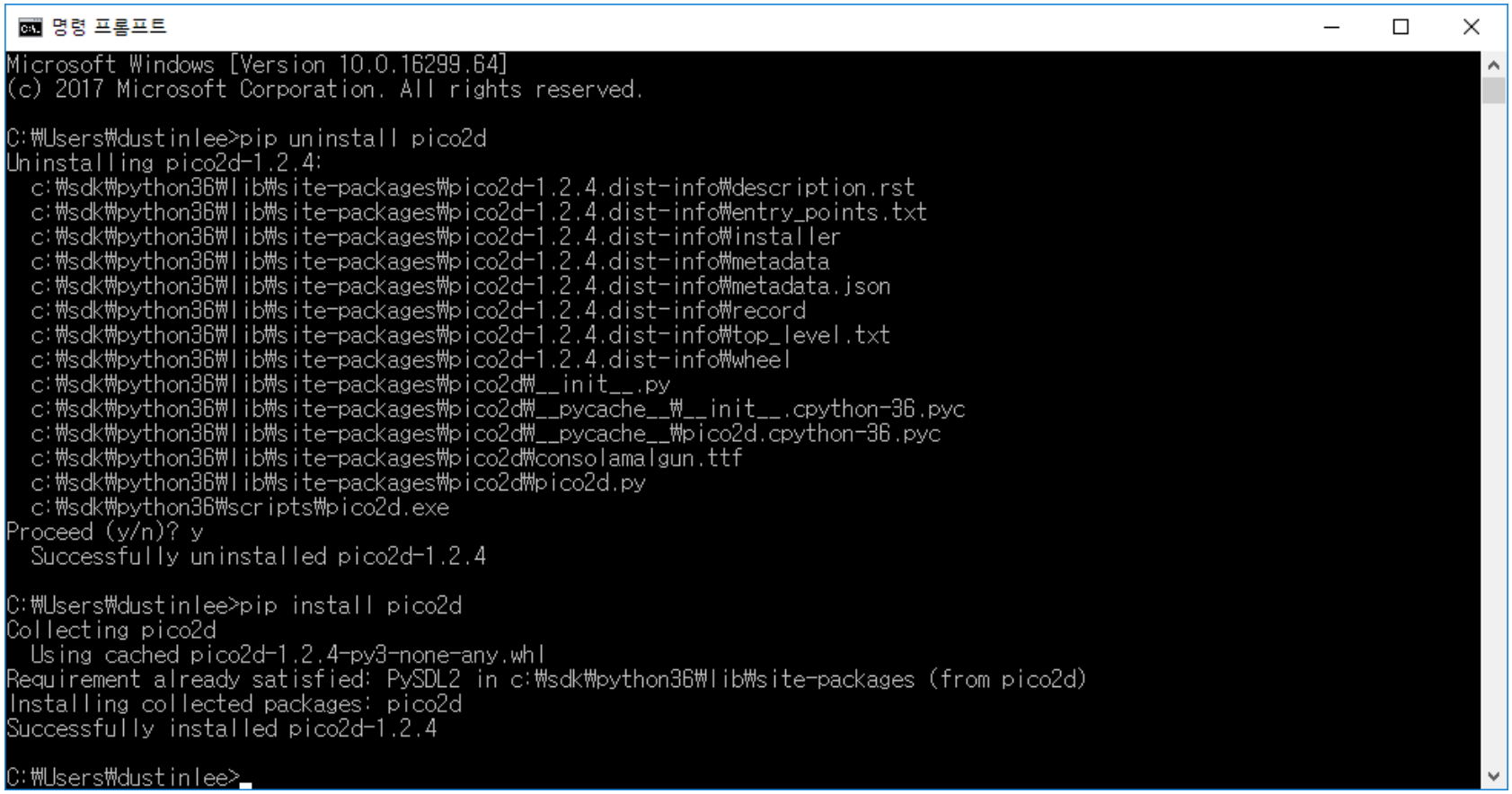
타일 기반 게임의 진화: Final Fantasy VI (1994)



타일 기반 게임의 진화: Civilization II (1996)

타일 기반 게임의 진화: Age of Wonders (2003)

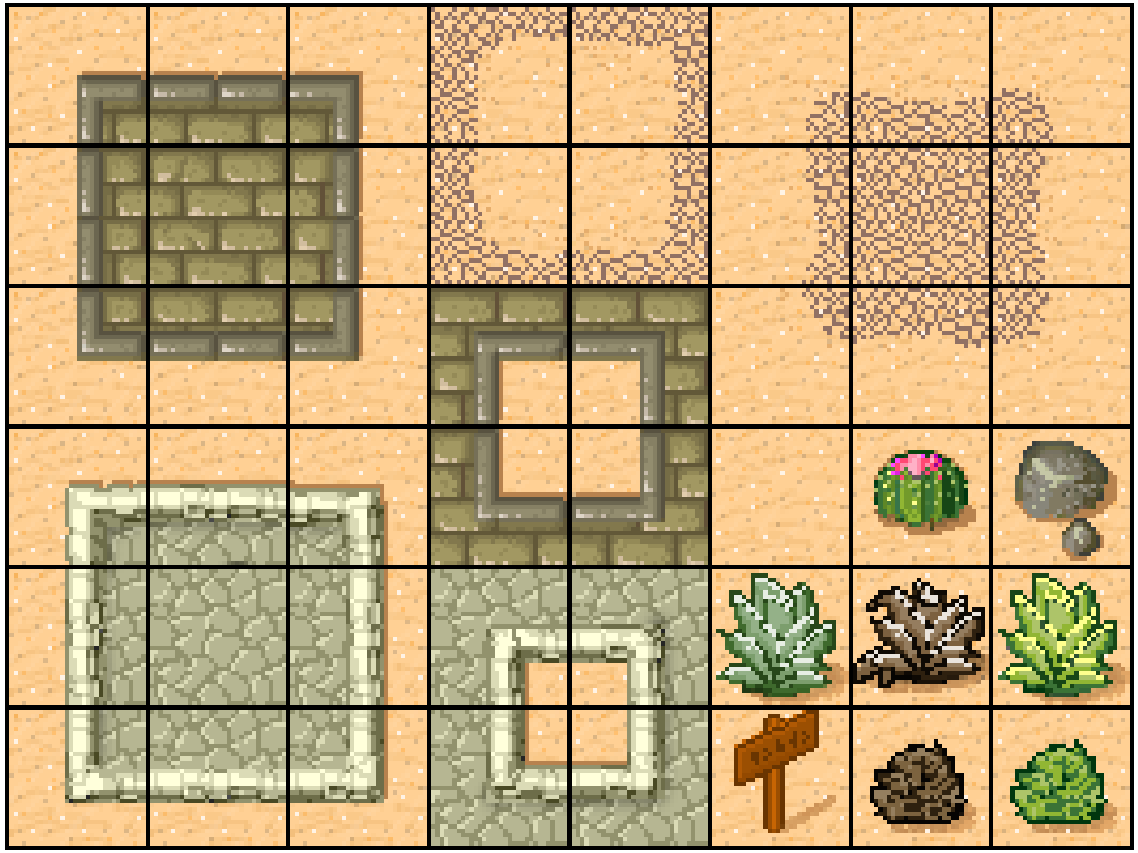
pico2d 업데이트



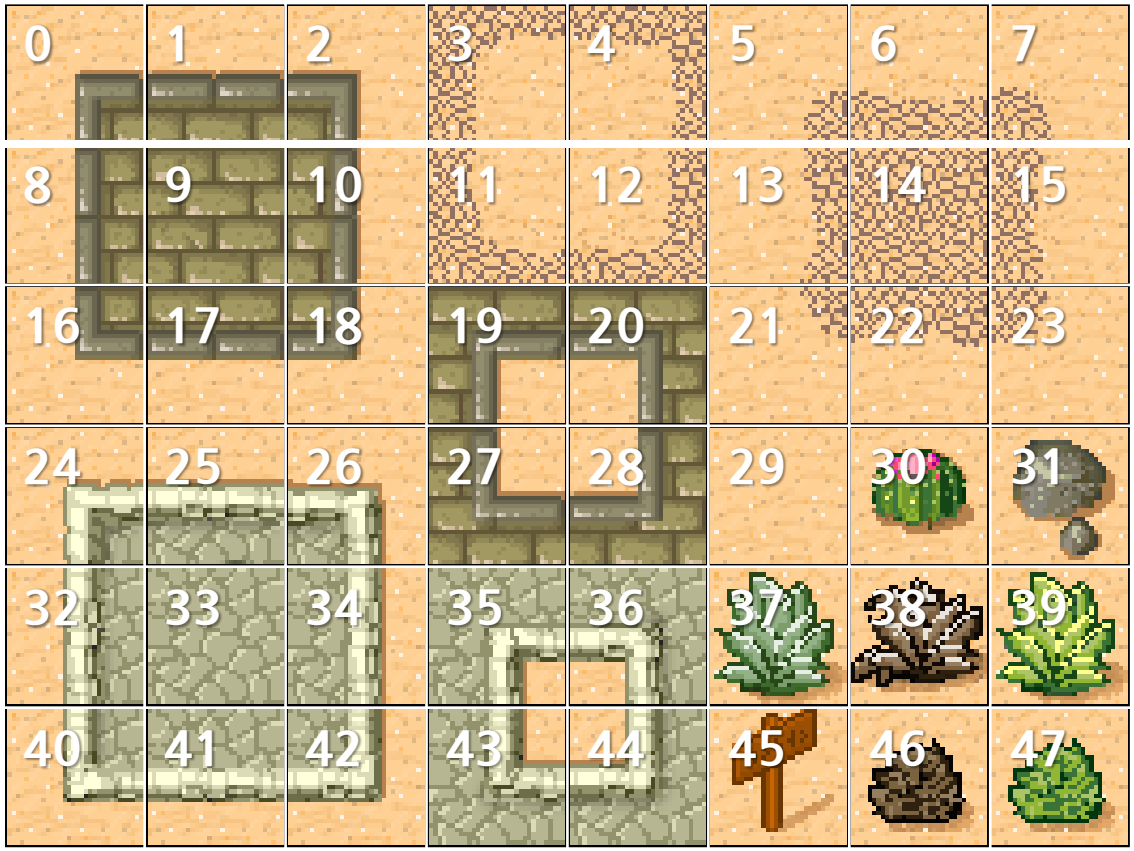
실습

타일셋 구현

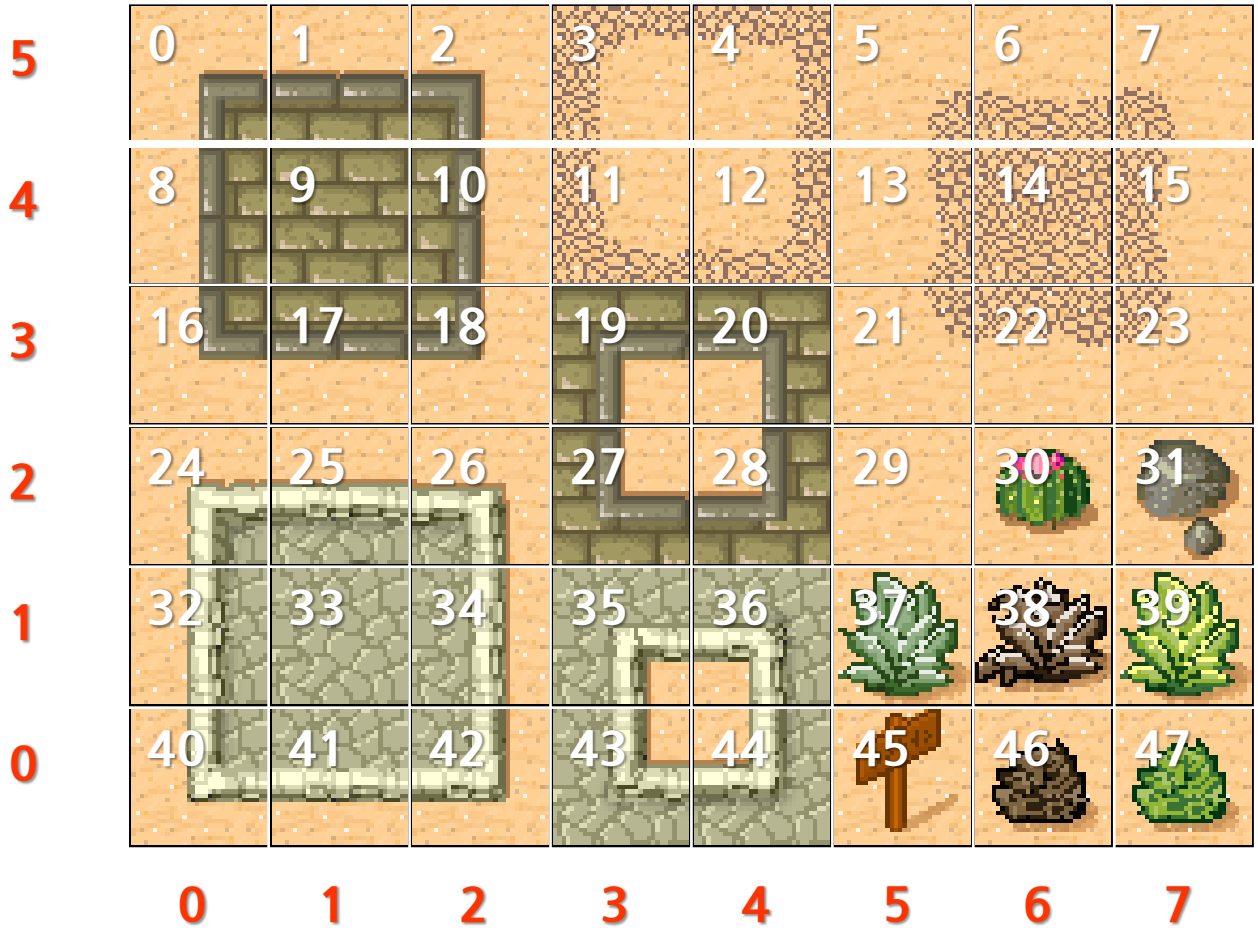
타일 이미지: desert_tiles.png



타일별 ID



ID로부터 이미지 영역 계산



locals()와 update() 함수

Python 3.6.2 Shell

File Edit Shell Debug Options Window Help

```
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.
>>>
>>>
>>> a
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    a
NameError: name 'a' is not defined
>>> b
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    b
NameError: name 'b' is not defined
>>>
>>> data = {'a': 3, 'b':4}
>>>
>>> locals
<built-in function locals>
>>> locals()
{'__name__': '__main__', '__doc__': None, '__package__': None, '__loader__': <class '_frozen_importlib.BuiltinImporter'>, '__spec__': None, '__annotations__': {}, '_
_builtins__': <module 'builtins' (built-in)>, 'data': {'a': 3, 'b': 4}}
>>> locals().update(data)
>>> a
3
>>> b
4
>>> locals()
{'__name__': '__main__', '__doc__': None, '__package__': None, '__loader__': <class '_frozen_importlib.BuiltinImporter'>, '__spec__': None, '__annotations__': {}, '_
_builtins__': <module 'builtins' (built-in)>, 'data': {'a': 3, 'b': 4}, 'a': 3, 'b': 4}
>>>
>>>
```

Ln: 30 Col: 4

self.__dict__와 update()

Python 3.6.2 Shell

File Edit Shell Debug Options Window Help

```
>>> class Test:
        def __init__(self):
                print(self.__dict__)
                data = {'x': 10, 'y':20}
                self.__dict__.update(data)
                print(self.__dict__)
                print(self.x)
                print(self.y)


>>> t = Test()
{}
{'x': 10, 'y': 20}
10
20
>>>
>>>
>>>
>>>
>>>
>>>
>>>
```

Ln: 42 Col: 4

desert_tileset.json


```
{
 "type": "tileset",
 "name": "Desert",
 "image": "desert_tiles.png",
 "imageheight": 192,
 "imagewidth": 256,
 "tileheight": 32,
 "tilewidth": 32,
 "columns": 8,
 "tilecount": 48
}
```

TileSet.py


```
class TileSet:

    def load(self, file_name):
        f = open(file_name)
        data = json.load(f)
        f.close()
        self.__dict__.update(data)
        print(self.__dict__)
        self.base_image = load_image(self.image)
        self.tile_images = []
        for i in range(self.tilecount):
            col, row = i % self.columns, i // self.columns
            left = col * self.tilewidth
            bottom = self.base_image.h - (row + 1) * self.tileheight
            image = self.base_image.clip_image(left, bottom,
                                               self.tilewidth, self.tileheight)
            self.tile_images.append(image)
```

TileSet.py


```
def load_tile_set(file_name):

    tile_set = TileSet()
    tile_set.load(file_name)

    return tile_set


if __name__ =='__main__':

    open_canvas(800, 600)

    tile_set = load_tile_set('desert_tileset.json')

    for i in range(tile_set.tilecount):
        col = i % tile_set.columns
        row = i // tile_set.columns
        tile_set.tile_images[i].draw_to_origin(400 + col * tile_set.tilewidth,
                                               300 + row * tile_set.tileheight)

    update_canvas()
    delay(5)
    close_canvas()
```

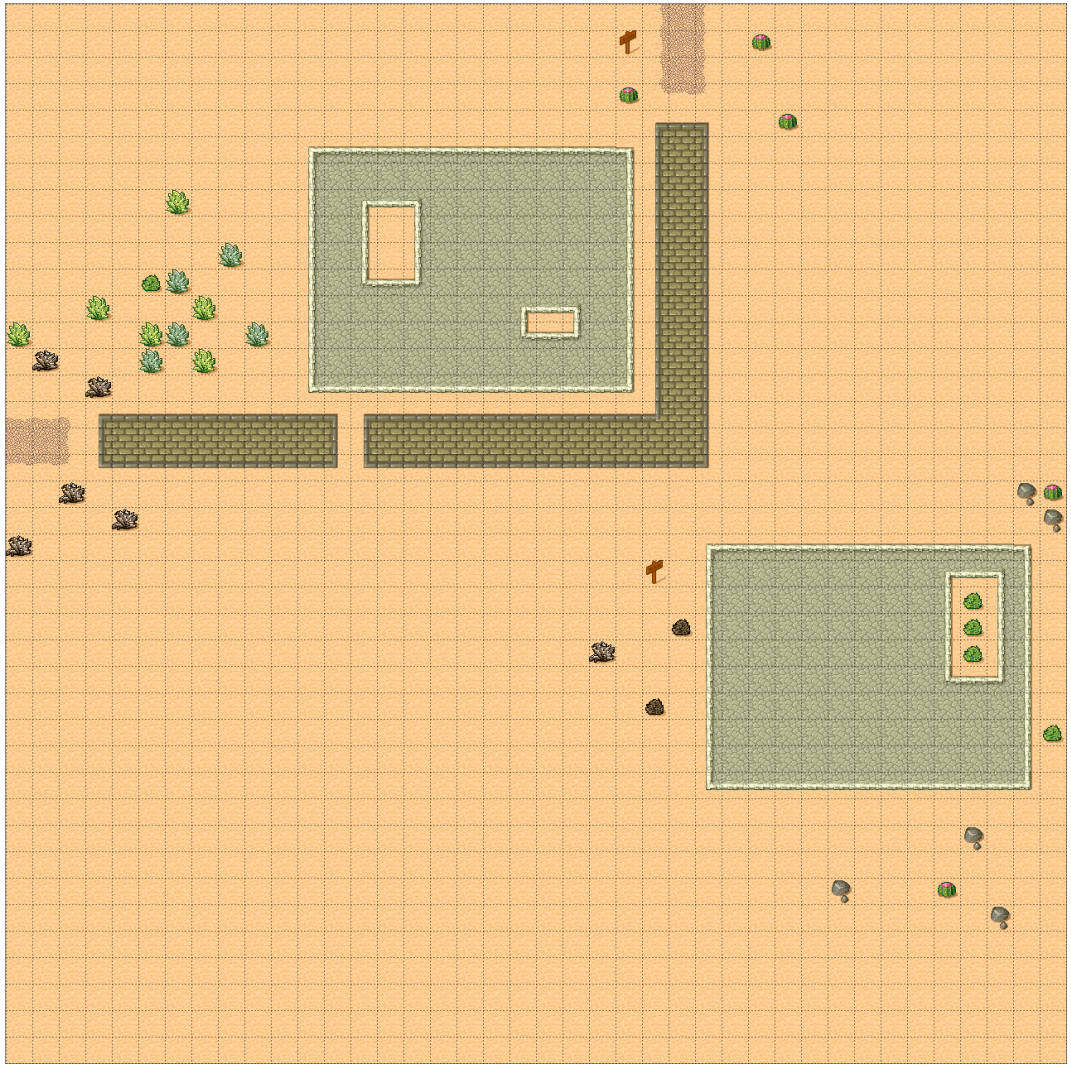
행의 반전

print(self.__dict__)

실습

타일맵 렌더링

목표 맵 (40 x 40)



desert_map.json


```
{
 "orientation":"orthogonal",
 "width":40,
 "height":40,
 "tilewidth":32,
 "tileheight":32,
 "tilesets":[ { "firstgid":1,"source":"desert_tileset.json"} ],
 "layers":[
        {
         "name":"Ground",
         "width":40,
         "height":40,
         "data":[30, 30, 30, 30, 30, ……],
         "opacity":1,
         "type":"tilelayer",
         "visible":true,
         "x":0,
         "y":0
        }]
}
```

TileMap.py


```
def load(self, name):
    f = open(name)
    info = json.load(f)
    f.close()

    self.__dict__.update(info)
    print(self.tilesets[0])
    self.tile_set = load_tile_set(self.tilesets[0]['source'])
    self.firstgid = self.tilesets[0]['firstgid']
    self.data = self.layers[0]['data']

    new_data = []
    for row in reversed(range(self.height)):
        new_data.append(self.data[row * self.width : row * self.width + self.width])
    self.data = new_data
```

TileMap.py


```
def clip_draw_to_origin(self, l, b, w, h, dx, dy):

    tl = l // self.tilewidth
    tb = b // self.tileheight
    tw = (l + w) // self.tilewidth - tl + 1
    th = (b + h) // self.tileheight - tb + 1

    lo = l % self.tilewidth
    bo = b % self.tileheight

    for x in range(tl, min(tl + tw, self.width)):
        for y in range(tb, min(tb + th, self.height)):
            self.tile_set.tile_images[self.data[y][x]-self.firstgid].
              draw_to_origin((x-tl) * self.tilewidth - lo, (y-tb)* self.tileheight - bo)
```

background.py


```
from TileMap import load_tile_map

class FixedTileBackground:

    def __init__(self):
        self.tile_map = load_tile_map('desert_map.json')
        self.canvas_width = get_canvas_width()
        self.canvas_height = get_canvas_height()
        self.w = self.tile_map.width * self.tile_map.tilewidth
        self.h = self.tile_map.height * self.tile_map.tileheight
```

scroll_state.py


```
from boy import FreeBoy as Boy

from background import FixedTileBackground as Background

name = "scroll_state"

boy = None
background = None
```

실행 화면

Tiled Map Editor (www.mapeditor.org)