Lecture #13. 게임 프레임웍

2D 게임 프로그래밍

이대현 교수

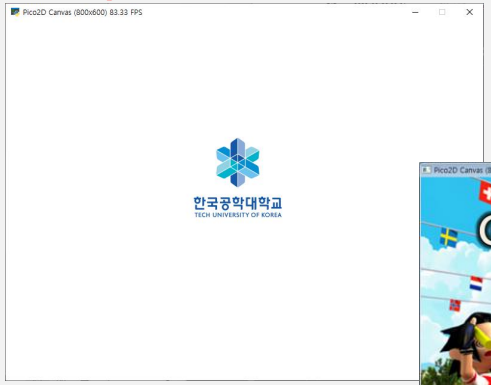
한국공학대학교
TECH UNIVERSITY OF KOREA

학습 내용

- 게임 모드
- 게임 프레임웍
- 로고 화면의 구현
- 타이틀 화면의 구현
- 메인 게임 구현

오늘 만들어 볼 것

로고화면



타이틀화면

메인플레이

리팩토링

리팩토링 (1) – 기존 코드를 play_mode.py 로 정리

control_boy.py

```python
from pico2d import *

import game_world
from grass import Grass
from boy import Boy

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)


def create_world():
    global running
    global grass
    global boy

    running = True

    grass = Grass()
    game_world.add_object(grass, 0)

    boy = Boy()
    game_world.add_object(boy, 1)


open_canvas()
create_world()
while running:
    handle_events()
    game_world.update()
    clear_canvas()
    game_world.render()
    update_canvas()
    delay(0.01)
close_canvas()
```

수정

play_mode.py

```python
# 앞부분 생략

def init():
    global running
    global grass
    global boy

    running = True

    grass = Grass()
    game_world.add_object(grass, 0)

    boy = Boy()
    game_world.add_object(boy, 1)


def finish():
    pass


def update():
    game_world.update()


def draw():
    clear_canvas()
    game_world.render()
    update_canvas()

open_canvas()
init()
while running:
    handle_events()
    update()
    draw()
    delay(0.01)
finish()
close_canvas()
```

리팩토링 (2) – play_mode.py 에서 메인 코드 분리하여 main.py 생성

play_mode.py

```
# 앞부분 생략

def init():
    global running
    global grass
    global boy

    running = True

    grass = Grass()
    game_world.add_object(grass, 0)

    boy = Boy()
    game_world.add_object(boy, 1)


def finish():
    pass


def update():
    game_world.update()


def draw():
    clear_canvas()
    game_world.render()
    update_canvas()

open_canvas()
init()
while running:
    handle_events()
    update()
    draw()
    delay(0.01)
finish()
close_canvas()
```

분리

main.py

```
import pico2d
import play_mode

pico2d.open_canvas()

play_mode.init()

while play_mode.running:
    play_mode.handle_events()
    play_mode.update()
    play_mode.draw()
    pico2d.delay(0.01)

play_mode.finish()

pico2d.close_canvas()
```

로고 모드 구현

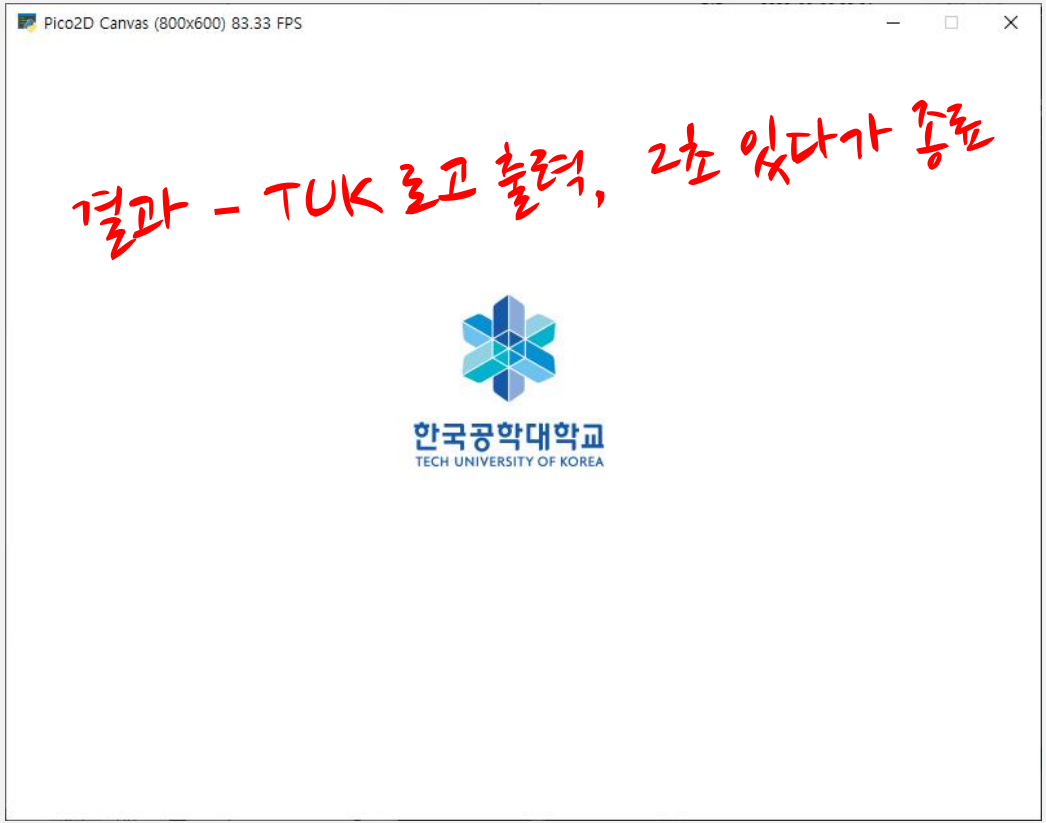
로고 모드 구현: logo_mode.py

```
from pico2d import load_image, delay, clear_canvas, update_canvas, get_events, get_time

def init():
    global image
    global running
    global logo_start_time

    image = load_image('tuk_credit.png')
    running = True
    logo_start_time = get_time()

def finish():
    global image
    del image

def update():
    global running
    global logo_start_time
    if get_time() - logo_start_time >= 2.0:
        logo_start_time = get_time()
        running = False

def draw():
    clear_canvas()
    image.draw(400, 300)
    update_canvas()

def handle_events():
    events = get_events()
```

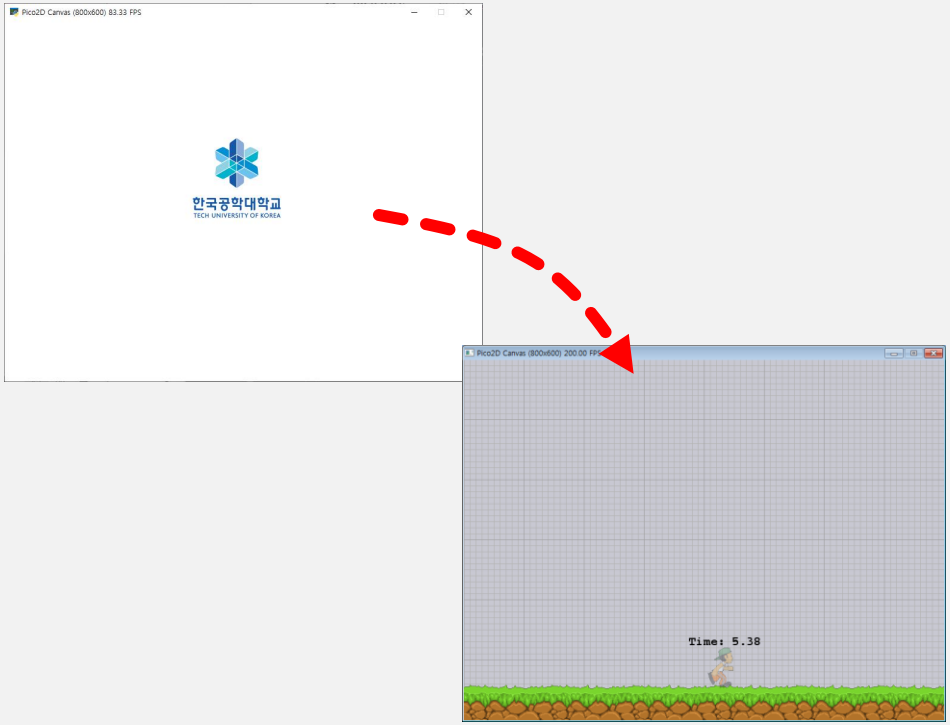
main.py


```
from pico2d import open_canvas, delay, close_canvas
import logo_mode

open_canvas()

logo_mode.init()

while logo_mode.running:
    logo_mode.handle_events()
    logo_mode.update()
    logo_mode.draw()

logo_mode.finish()

close_canvas()
```

실행 – main.py 를 실행



로고 화면 후에 플레이 모드로 가려면?



게임 모드의 이해 (1)

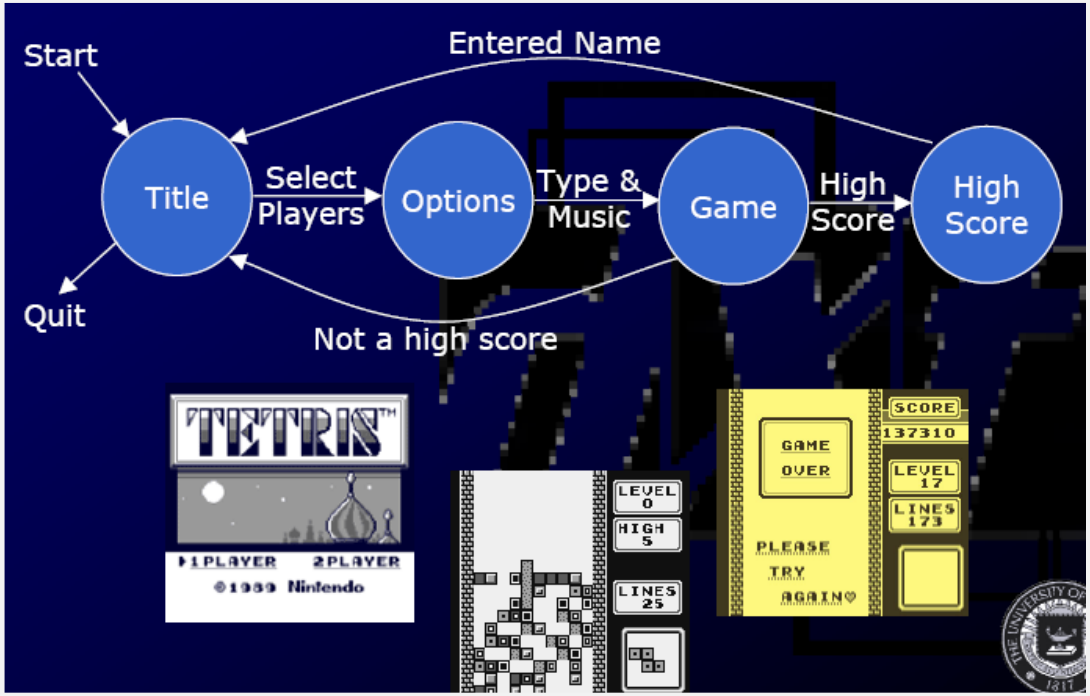
- **게임 모드란?**
  - 게임 프로그램 실행 중에 지속적으로 머물러 있는 특정 상황, 씬, ….
  - 사용자 입력(키보드 또는 마우스 입력)에 대한 대응 방식은 게임 모드에 따라 달라짐.
  - 작은 게임 루프로 볼 수 있음.

- 맵 선택 모드.
- 방향키는 맵 선택을 처리.

- 게임 메인 플레이 모드.
- 방향키는 캐릭터의 이동을 처리.

게임 모드의 이해 (2)

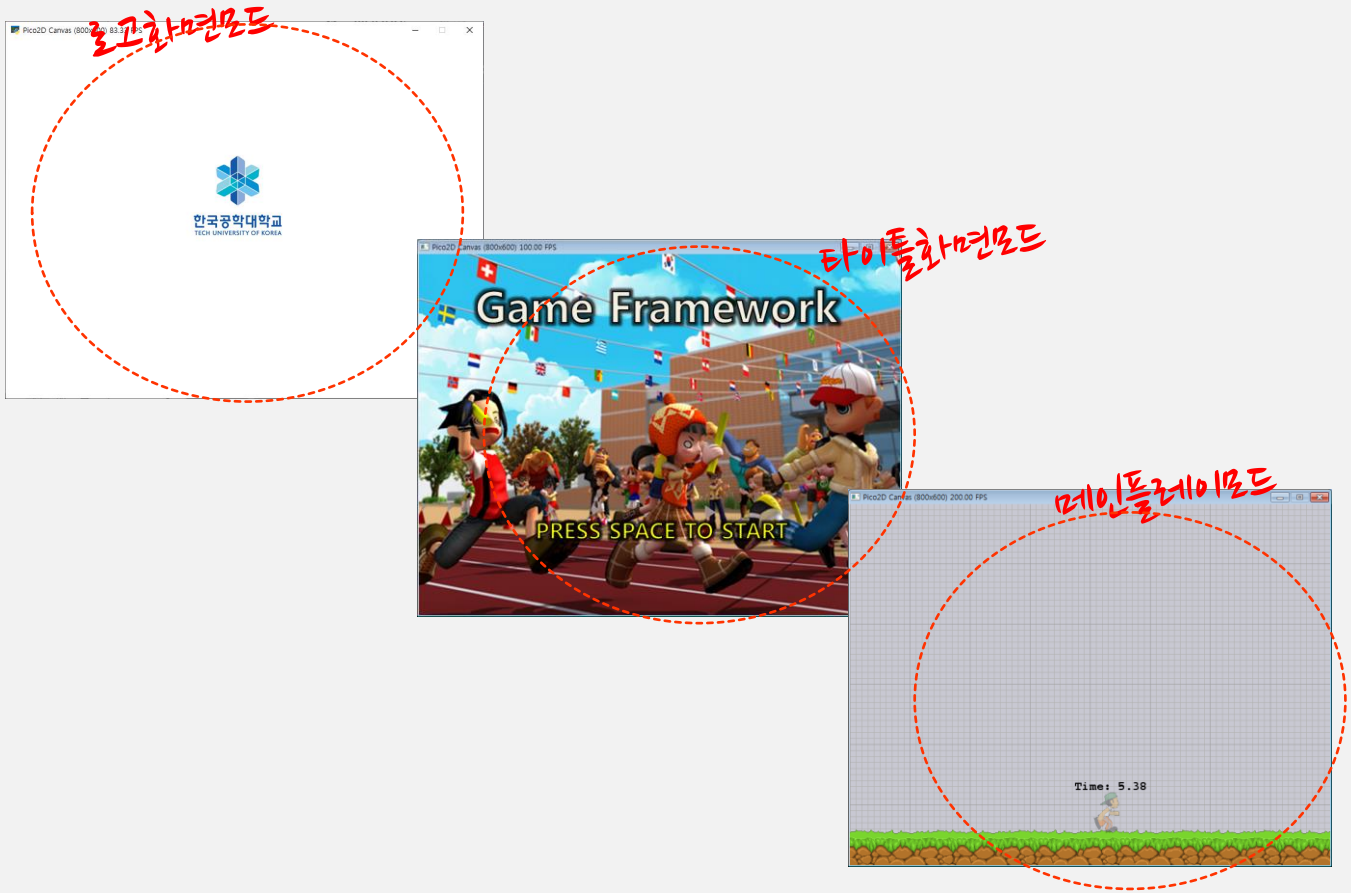
- 게임 프로그램은 여러 개의 게임 모드들의 연결로 구현됨.
  - 예) 테트리스 게임



게임 프레임워크

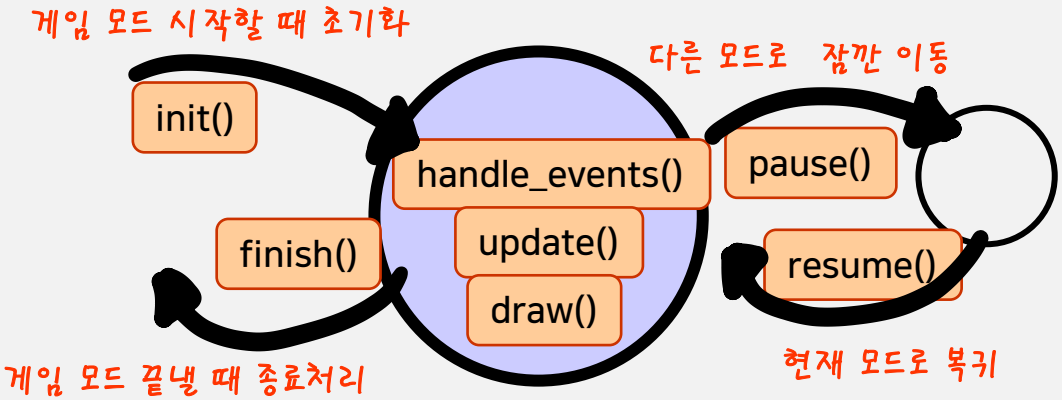
- 게임 모드들을 효과적으로 연결하는 소프트웨어 구조.
- 일종의 Task Switching System
- 디자인 패턴 중, State Pattern 혹은 Strategy Pattern에 해당됨.

게임 프레임워크 활용 순서 #1. 각각의 모드를 구현



게임 프레임워크 활용 순서 #2. 모드 간의 이동을 구현.

게임 모드의 구현



상태간의 전환: game_framework을 이용

```
run(mode):
state를 시작 게임 모드로 하여, 게임 실행을 시작함.

quit(): 게임을 중단

change_mode(mode):
게임 모드를 mode로 이동. 이전 게임 모드를 완전히 나옴.

push_mode(mode):
게임 상태를 mode로 이동. 이전 게임 모드 데이타는 남아 있음.

pop_mode(): 이전 게임 모드로 복귀
```

로고 화면 구현 (2)

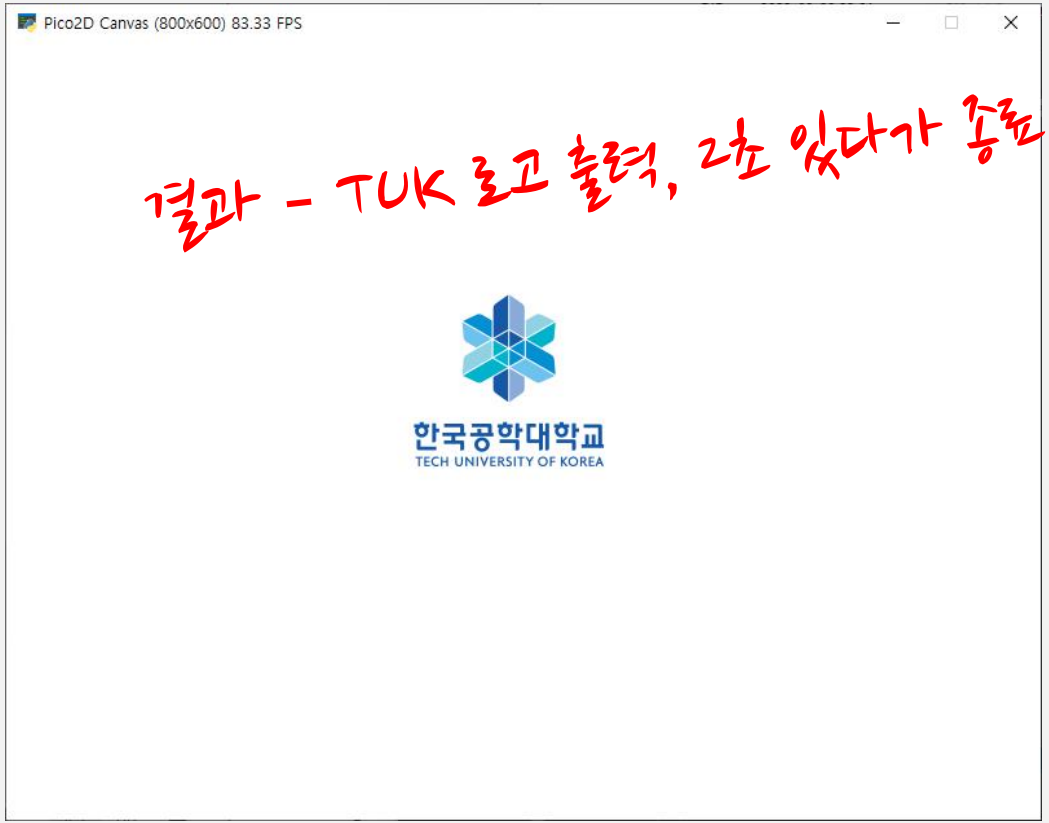
로고 모드 구현: logo_mode.py 수정


```
import game_framework
from pico2d import load_image, delay, clear_canvas, update_canvas, get_events, get_time

def init():
    global image
    global running
    global logo_start_time

    image = load_image('tuk_credit.png')
    running = True
    logo_start_time = get_time()

def finish():
    global image
    del image

def update():
    global logo_start_time
    if get_time() - logo_start_time >= 2.0:
        logo_start_time = get_time()
        game_framework.quit()
```

main.py 구현


```
from pico2d import open_canvas, close_canvas
import game_framework

import logo_mode

open_canvas()
game_framework.run(logo_mode)
close_canvas()
```

실행 – main.py 를 실행

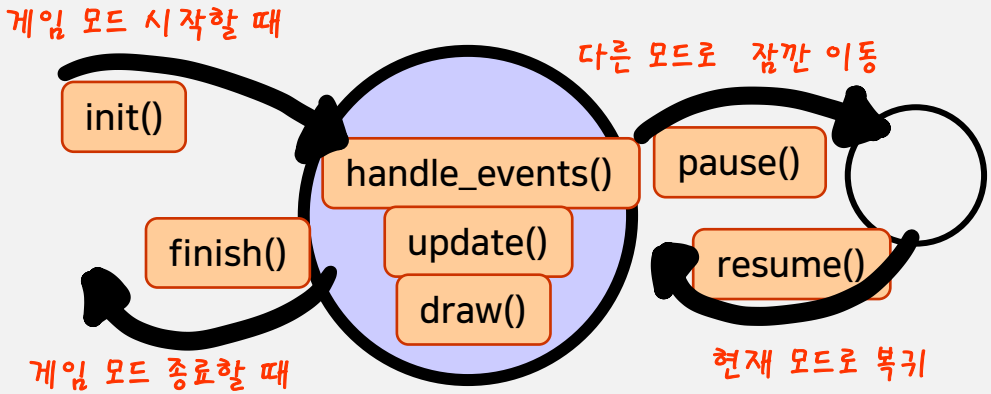


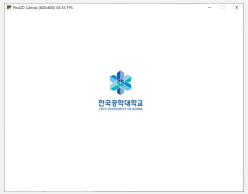
게임 모드의 뼈대

```
def init(): pass

def finish():  pass

def update(): pass

def draw(): pass

def handle_events(): pass

def pause(): pass

def resume(): pass
```

OOOO_mode 의 구현과 활용

1. OOOO_mode.py 를 만든다
2. OOOO_mode.py의 내부 함수들을 작성한다.
3. 다른 소스에서 import OOOO_mode 를 해서 활용한다.





게임의 구성과 시작 – 게임프레임워크 활용

- game_framework 를 import 한다.
- 시작 게임 모드를 import 한다.

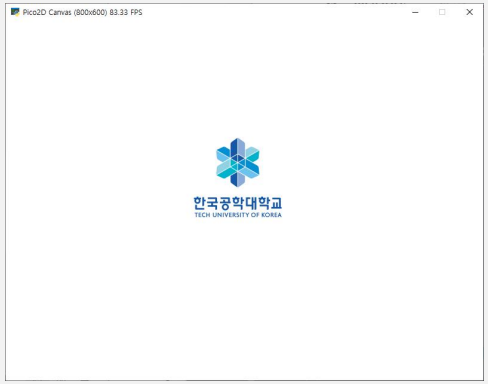
- 시작 게임 모드를 지정한 후, game_framework 를 시작한다.

```
import game_framework
import pico2d

import start_mode

pico2d.open_canvas()
game_framework.run(start_mode)
pico2d.close_canvas()
```

타이틀 모드 추가

로고 화면에 이어지는 타이틀 화면

logo_mode.py



title_mode.py

title_mode.py


```
def init():
    global image
    image = load_image('title.png')

def finish():
    global image
    del image

def handle_events():
    events = get_events()
    for event in events:
        if event.type == SDL_QUIT:
            game_framework.quit()
        elif event.type == SDL_KEYDOWN and event.key == SDLK_ESCAPE:
            game_framework.quit()

def draw():
    clear_canvas()
    image.draw(400,300)
    update_canvas()
```

logo_mode.py 의 수정


```
import title_mode

# ... 중략 ...

def update():
    global logo_start_time
    if get_time() - logo_start_time >= 2.0:
        logo_start_time = get_time()
        game_framework.change_mode(title_mode)
```

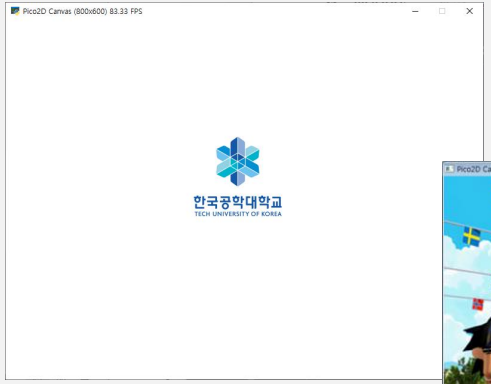
main.py 수정


```
from pico2d import open_canvas, close_canvas
import game_framework

import logo_mode as start_mode

open_canvas()
game_framework.run(start_mode)
close_canvas()
```

게임 플레이 모드 추가

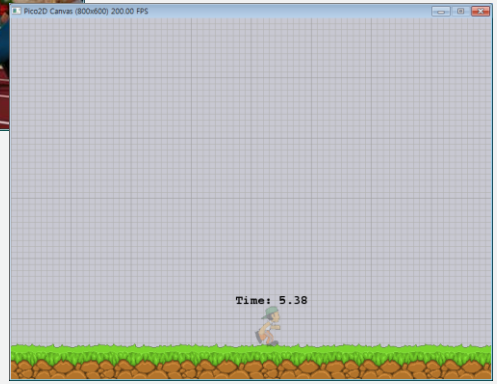
로고 화면에 이어지는 타이틀 화면

logo_mode.py



title_mode.py

play_mode.py



play_mode.py 의 수정


```
def handle_events():
    events = get_events()
    for event in events:
        if event.type == SDL_QUIT:
            game_framework.quit()
        elif event.type == SDL_KEYDOWN and event.key == SDLK_ESCAPE:
            game_framework.change_mode(title_mode)
        else:
            boy.handle_event(event)
```

title_mode.py 의 수정


```
def handle_events():
    events = get_events()
    for event in events:
        if event.type == SDL_QUIT:
            game_framework.quit()
        elif event.type == SDL_KEYDOWN and event.key == SDLK_ESCAPE:
            game_framework.quit()
        elif (event.type, event.key) == (SDL_KEYDOWN, SDLK_SPACE):
            game_framework.change_mode(play_mode)
```

game world clear 필요

play_mode.py

```
def finish():
    game_world.clear()
    pass
```

game_framework.py

```
def clear():
    for layer in objects:
        layer.clear()
```

아이템 모드 구현

boy.py 수정


```
class Boy:
    def __init__(self):
        self.x, self.y = 400, 90
        self.frame = 0
        self.action = 3
        self.face_dir = 1
        self.dir = 0
        self.image = load_image('animation_sheet.png')
        self.state_machine = StateMachine(self)
        self.state_machine.start()
        self.item = None

    def fire_ball(self):

        if self.item ==   'Ball':
            ball = Ball(self.x, self.y, self.face_dir*10)
            game_world.add_object(ball)
        elif self.item == 'BigBall':
            ball = BigBall(self.x, self.y, self.face_dir*10)
            game_world.add_object(ball)
```

play_mode.py


```
def handle_events():
    events = get_events()
    for event in events:
        if event.type == SDL_QUIT:
            game_framework.quit()
        elif event.type == SDL_KEYDOWN and event.key == SDLK_ESCAPE:
            game_framework.change_mode(title_mode)
        elif event.type == SDL_KEYDOWN and event.key == SDLK_i:
            game_framework.push_mode(item_mode)
        else:
            boy.handle_event(event)
```

play_mode.py - pause 와 resume 추가

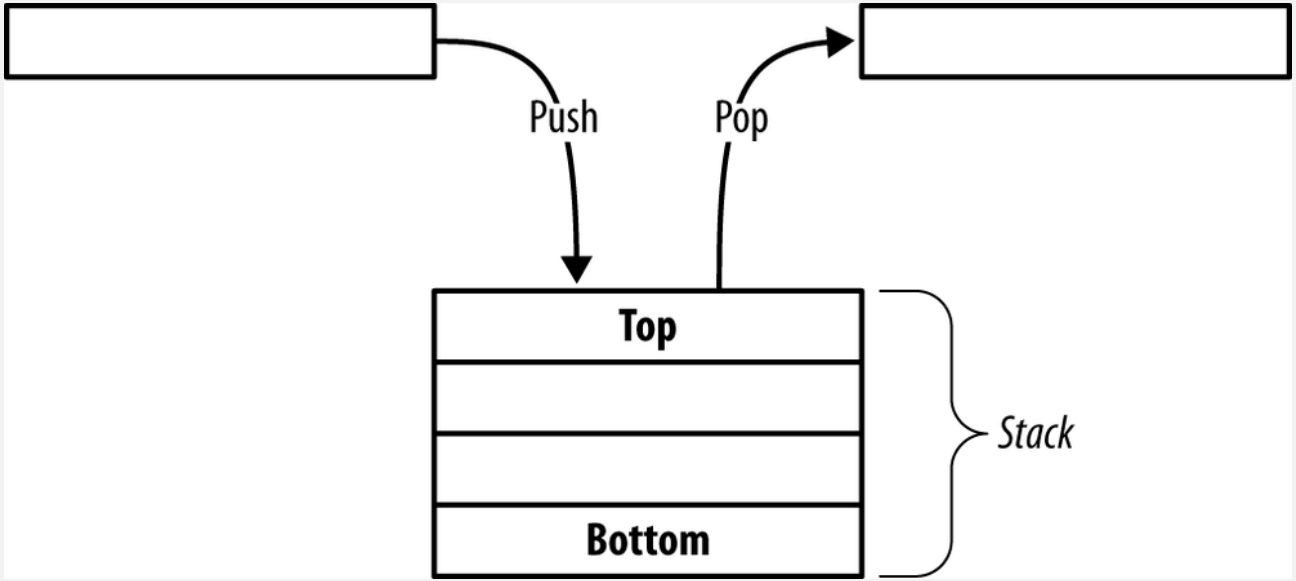
- push_state 와 pop_state 를 호출하면, pause와 resume 이 call back 되므로 실제 내용은 없더라도 뼈대는 만들어줘야 함.

```
def pause():
    pass

def resume():
    pass
```

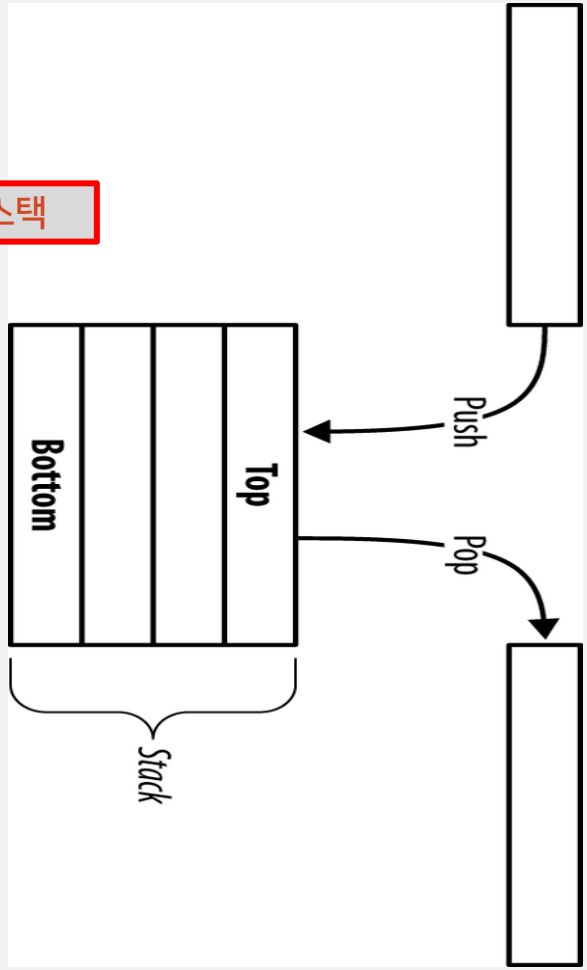
item_mode.py


```
def handle_events():
    events = get_events()
    for event in events:
        if event.type == SDL_QUIT:
            game_framework.quit()
        elif event.type == SDL_KEYDOWN:
            match event.key:
                case pico2d.SDLK_ESCAPE:
                    game_framework.pop_mode()
                case pico2d.SDLK_0:
                    play_mode.boy.item = None
                    game_framework.pop_mode()
                case pico2d.SDLK_1:
                    play_mode.boy.item = 'Ball'
                    game_framework.pop_mode()
                case pico2d.SDLK_2:
                    play_mode.boy.item = 'BigBall'
                    game_framework.pop_mode()
```

Stack 자료 구조



list 를 이용한 스택



game_framework.py 분석(1)

```
def run(start_state):
    global running, stack
    running = True

    stack = [start_state]
    start_state.enter()

    while (running):
        stack[-1].handle_events()
        stack[-1].update()
        stack[-1].draw()

    # repeatedly delete the top of the stack
    while (len(stack) > 0):
        stack[-1].exit()
        stack.pop()
```

start_state 를 담고 있는 스택을 생성

현재 게임 상태(다시 말하면, stack top에 있는 게임 상태)에 대한 게임 루프를 진행

스택에 남아있는 모든 게임 상태들을 차례로 제거

game_framework.py 분석 (2)

```
def change_state(state):
    global stack
    if (len(stack) > 0):
        stack[-1].exit()
        stack.pop()
    stack.append(state)
    state.enter()


def pop_state():
    global stack
    if (len(stack) > 0):
        stack[-1].exit()
        stack.pop()
    if (len(stack) > 0):
        stack[-1].resume()
```

현재 상태를 삭제한 후,
새로운 상태를 추가하고, enter로 들어간다.

Stack Top의 상태를 exit() 한 후, 상태를 제거.
이제 Stack Top에는 이전 상태가 있으므로, 이 내용을 다시 가져옴(resume)

game_framework.py 분석 (3)

```
def push_state(state):
    global stack
    if (len(stack) > 0):
        stack[-1].pause()
    stack.append(state)
    state.enter()


def quit():
    global running
    running = False
```

현재 상태를 저장하고(Pause),
새로운 상태로 들어감.