🍗

[Code Convention]

#변수

- 변수의 경우 Camel Style로 표기합니다.
- int myFirstField // Camel style 첫 글자만 소문자, 이후 새로운 단어가 등장하면 대문자로 구분합니다.

#주석

- 한 행의 주석은 해당행의 끝에 작성한다.
- 함수는 함수위에 작성한다.
- 주석은 가능한 한 같은 열에 정렬되도록 합니다.
- 주석 근데 걍 대충 알아볼 수 만 있으면 됨ㅎ 굳이 규칙 막 지키려고 안해도됩니다 라는 뜻

```
class CorrectExampleClass
{
private:
    int exampleField = 1; // Line comment

    //Block comment
    void Foo()
    {
        //Code
    }
}
```

#함수

- 대문자 시작.
- 동사로 시작.
- bool타입은 is로 시작하면서 긍정적인 이름으로한다. (isNotDo() 이런식으로 하지 않기)
- {} 괄호는 내려쓰기.(BSD 스타일)

#private, protected

- 변수, 함수규칙을 따르며 앞에 _붙이기 ex) int _num;
- public은 아무것도 붙이지 않는다 ex) int num;

#상수 이름

- 상수 이름은 c로 시작하고 카멜스타일로 작성한다.

#전역 변수

- G로 시작하고 GEngine 이런식으로 작성한다.

#EnginePch.h

```
#pragma once

#include <windows.h>
#include <string>
#include <iostream>
#include <vector>
#include <map>
#include <time.h>
#include <array>
#include "Vector2.h"
#include <d2d1helper.h> // D2D1  쓰려구..
using namespace std;

#pragma comment(lib, "JW2DEngine.lib")

// typedef
using int8  = __int8;
using int16 = __int16;
using int32 = __int32;
using int64 = __int64;
using uint8 = unsigned __int8;
using uint16 = unsigned __int16;
using uint32 = unsigned __int32;
using uint64 = unsigned __int64;
using Matrix = D2D1::Matrix3x2F;

struct WindowInfo
{
  HWND  hWnd; // 출력 윈도우
  int32 width; // 너비
  int32 height; // 높이
  bool  windowed; // 창모드 or 전체화면
};

// Rigidbody bodytype 설정
enum class BODY_TYPE
{
  Dynamic,
  Kinematic,
  Static
};

// define
#define SCREEN_WIDTH 1920
#define SCREEN_HEIGHT 1080

#define DECLARE_SINGLE(type)    \
private:              \
  type() {}           \
  ~type() {}            \
public:               \
  static type* GetInstance()    \
  {               \
    static type instance;   \
    return &instance;     \
  }               \

#define GET_SINGLE(type)  type::GetInstance()

#define INPUT       GET_SINGLE(Input)
#define TIMER       GET_SINGLE(Timer)
#define SCENE_MANAGER   GET_SINGLE(SceneManager)
#define SOUND_MANAGER   GET_SINGLE(SoundManager)
#define EVENT_MANAGER   GET_SINGLE(EventManager)
#define DELTA_TIME      GET_SINGLE(Timer)->GetDeltaTime()

extern class JW2DEngine* GEngine;
```

- 필요한 라이브러리 + stl(string, vector, map... 등), usingnamespace를 여기서 작성합니다.

/

- 여러방면으로 알려줘야할 필요가 있는 struct, enum을 여기서 작성합니다.
- #define은 대문자로 작성하되 다른 단어가 나오면 언더바*() 로 구분해줍니다.*
  ex) #define SCREEN_WIDTH
- 싱글톤을 만드려면 클래스 첫부분에 DECLARE_SINGLE(클래스 이름); 해주면 된다
  싱글톤을 만들었으면 #define INPUT   GET_SINGLE(Input) 처럼 define을 해준다.

- Engine.cpp 에 엔진을 동적할당 했으며 extern으로 GEngine을 알려준다.
  엔진이 필요하다면 cpp 쪽에 #include<JW2DEngine.h> 을 해주고 쓰면 됩니다.

#초기화

```
Player::Player(GameObject* gameObject)
  : MonoBehaviour(gameObject),
    _state(State::IDLE), _speed(200.f), _jumpPower(10.0f), _isJumping(false), _h(0.f), _v(0.f), _attackRange(nullptr),
    _isAttack(false), _scale(_transform->GetLocalScale()), _isDash(false), _isFlipped(false), _dashTime(0.f), _waitingTime(3), _dash(8
    _transform(gameObject->GetComponent<Transform>()),
    _rigidbody(gameObject->GetComponent<Rigidbody>()),
    _animator(gameObject->GetComponent<Animator>()),
    _text(gameObject->GetComponent<Text>())
{}
```

- 초기화 리스트를 이용해서 초기화 합니다.
- 컴포넌트를 상속받는 클래스들에는 규칙이 있습니다.

  ```
  Rigidbody::Rigidbody(GameObject* gameObject)
    : Component(gameObject, COMPONENT_TYPE::RIGIDBODY),
      _bodyType(BODY_TYPE::Dynamic), _mass(1), _linearDrag(0), _gravityScale(2),
      _gravity({0.0f, 9.8f}), _isGravity(false)
  ```

  - 컴포넌트를 받는 모든 클래스의 생성자는 매개변수로 GameObject 를 받습니다.
  - Component 초기화를 꼭 해주어야 합니다
    : Component(gameObject, COMPONENT_TYPE::RIGIDBODY) // gameObject, 타입
  - MonoBehaviour은 컴포넌트를 상속받기때문에 여기 자체에서 타입을 정해주었고
    스크립트에서는 이제 위에 플레이어 코드처럼 작성하면 됩니다.