12. SceneManager와 Timer

Scene.h

```h
#pragma once

#include "../Game.h"

class CScene
{
protected:

	friend class CSceneManager;

protected:
	CScene();
	virtual ~CScene() = 0; // 순수 가상함수. 추상클래스.

public:
	virtual bool Init();
};
```

추상 클래스이므로
객체 생성 불가.

업캐스팅 통한 관리자
역할 수행

Scene.cpp

```cpp
#include "Scene.h"

CScene::CScene()
{
}

CScene::~CScene()
{
}

bool CScene::Init()
{
    return false;
}
```

상속

InGameScene.h

```h
#pragma once

#include "Scene.h"

class CInGameScene : public CScene
{
private:
	friend class CSceneManager;

private: // SceneManger를 통해서만 생성하도록 함.
	CInGameScene();
	virtual ~CInGameScene() override;

public:
	virtual bool Init() override;
};
```

CSceneManger 클래스 통해서만
Scene을 생성할 수 있도록 열어줌.

InGameScene.cpp

```cpp
#include "InGameScene.h"

CInGameScene::CInGameScene()
{
}

CInGameScene::~CInGameScene()
{
}

bool CInGameScene::Init()
{
    // 부모 Scene 클래스의 초기화 함수를 호출해줌.
    if( !CScene::Init() )
    {
        return false;
    }

    // 본인 기능 수행.

    return true;
}
```

SceneManager.h

```h
#pragma once

#include "../Game.h"

class CSceneManager
{
private:
	class CScene* m_pScene;
	class CScene* m_pNextScene;

public:
	bool Init();

public:
	template<typename T>
	T* CreateScene( SCENE_CREATE sc )
	{
		T* pScene = new T;

		if( !pScene->Init() )
		{
			SAFE_DELETE( pScene );
			return nullptr;
		}

		switch( sc )
		{
		case SC_CURRENT:
			SAFE_DELETE( m_pScene );
			m_pScene = pScene;
			break;
		case SC_NEXT:
			SAFE_DELETE( m_pNextScene );
			m_pNextScene = pScene;
			break;
		}

		return pScene;
	}

	DECLARE_SINGLE( CSceneManager )
};
```

기존 씬 삭제.
생성된 씬으로
교체.

Flag.h

```h
#pragma once

// Scene Type
enum SCENE_CREATE
{
	SC_CURRENT,
	SC_NEXT
};
```

SceneManger.cpp

```cpp
#include "SceneManager.h"
#include "InGameScene.h"

DEFINITION_SINGLE(CSceneManager)

CSceneManager::CSceneManager()
	:
	m_pScene(nullptr),
	m_pNextScene(nullptr)
{
}

CSceneManager::~CSceneManager()
{
	SAFE_DELETE( m_pScene );
}

bool CSceneManager::Init()
{
	CreateScene<CInGameScene>( SC_CURRENT );

	return true;
}
```

Timer.h

```h
#pragma once

#include "../Game.h"

class CTimer
{
private:
	LARGE_INTEGER	m_tSecond;
	LARGE_INTEGER	m_tTime;
	float			m_fDeltaTime;
	float			m_fFPS;
	float			m_fFPSTime;
	int				m_iFrameMax;
	int				m_iFrame;

public:
	float GetDeltaTime() const
	{
		return m_fDeltaTime;
	}

	float GetFPS() const
	{
		return m_fFPS;
	}

public:
	bool Init();
	void Update();

	DECLARE_SINGLE( CTimer )
};
```

Timer.cpp

```cpp
#include "Timer.h"

DEFINITION_SINGLE(CTimer)

bool CTimer::Init()
{
	QueryPerformanceFrequency( &m_tSecond );
	QueryPerformanceCounter( &m_tTime );

	m_fDeltaTime = 0.f;
	m_fFPS = 0.f;
	m_fFPSTime = 0.f;
	m_iFrameMax = 60;
	m_iFrame = 0;

	return true;
}

void CTimer::Update()
{
	LARGE_INTEGER tTime;
	QueryPerformanceCounter( &tTime );

	m_fDeltaTime = (tTime.QuadPart - m_tTime.QuadPart) / (float)m_tSecond.QuadPart;
}
```

DeltaTime 갱신

Core.cpp

```cpp
#include "Core.h"
#include "Scene\SceneManager.h"
#include "Core/Timer.h"

CCore* CCore::m_pInst;
bool CCore::m_bLoop = true;

CCore::CCore()
{
}

CCore::~CCore()
{
	DESTROY_SINGLE( CSceneManager );
	DESTROY_SINGLE( CTimer );
}

bool CCore::Init( HINSTANCE hInst )
{
	m_hInst = hInst;

	// 윈도우 클래스 등록
	MyRegisterClass();

	// 해상도 설정
	m_tRs.iW = 1280;
	m_tRs.iH = 720;

	// 윈도우 창 생성
	Create();

	// 타이머 초기화
	if( !GET_SINGLE( CTimer )->Init() )
	{
		return false;
	}

	// 장면 관리자 초기화
	if( !GET_SINGLE( CSceneManager )->Init() )
	{
		return false;
	}

	return true;
}
```

관리자, 코어 기능 클래스 객체 삭제

init

```cpp
int CCore::Run()
{
	MSG msg;

	// Main message loop:
	while( m_bLoop )
	{
		if( PeekMessage( &msg, nullptr, 0, 0, PM_REMOVE ) )
		{
			TranslateMessage( &msg );
			DispatchMessage( &msg );
		}

		// 윈도우 데드 타임일 경우
		else
		{
			Logic();
		}
	}

	return (int)msg.wParam;
}
```

윈도우 데드타임에 게임 로직 수행

```cpp
void CCore::Logic()
{
	// 타이머 갱신
	GET_SINGLE( CTimer )->Update();

	float fDeltaTime = GET_SINGLE( CTimer )->GetDeltaTime();
}
```

로직 수행하기 위해 최신 delta time 얻어오기

```cpp
ATOM CCore::MyRegisterClass()
{
	WNDCLASSEXW wcex;

	wcex.cbSize = sizeof( WNDCLASSEX );

	wcex.style = CS_HREDRAW | CS_VREDRAW;
	wcex.lpfnWndProc = WndProc;
	wcex.cbClsExtra = 0;
	wcex.cbWndExtra = 0;
	wcex.hInstance = m_hInst;
	wcex.hIcon = LoadIcon( m_hInst, MAKEINTRESOURCE( IDI_ICON1 ) );
	wcex.hCursor = LoadCursor( nullptr, IDC_ARROW );
	wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
	wcex.lpszMenuName = NULL; //MAKEINTRESOURCEW(IDC_ASSORTROCKWINAPI);
	wcex.lpszClassName = L"WINAPIFRAMEWORK";
	wcex.hIconSm = LoadIcon( wcex.hInstance, MAKEINTRESOURCE( IDI_ICON1 ) );

	return RegisterClassExW( &wcex );
}

BOOL CCore::Create()
{
	// 메인 윈도우를 생성하고 윈도우 핸들에 저장
```