13. Layer 클래스와 POSITION 구조체

```cpp
typedef struct _tagPosition
{
	float x, y;

	_tagPosition()
		:
		x( 0.f ),
		y( 0.f )
	{

	}

	_tagPosition(float _x, float _y)
		:
		x( _x ),
		y( _y )
	{

	}

	_tagPosition(const _tagPosition& pos)
	{
		x = pos.x;
		y = pos.y;
	}

	_tagPosition( const POINT& pt )
	{
		x = pt.x;
		y = pt.y;
	}

	void operator = ( const _tagPosition& pos )
	{
		x = pos.x;
		y = pos.y;
	}

	void operator = ( const POINT& pt )
	{
		x = pt.x;
		y = pt.y;
	}

	void operator = ( float f[2] )
	{
		x = f[0];
		y = f[1];
	}
```

```cpp
// +
_tagPosition operator + ( const _tagPosition& pos )
{
	_tagPosition tPos;
	tPos.x += pos.x;
	tPos.y += pos.y;
	return tPos;
}

_tagPosition operator + ( float f[2]  )
{
	_tagPosition tPos;
	tPos.x += f[0];
	tPos.y += f[1];
	return tPos;
}

_tagPosition operator + ( float f )
{
	_tagPosition tPos;
	tPos.x += f;
	tPos.y += f;
	return tPos;
}
```

이런식으로 향후 사용할 구조체를 정의해 줌.

```cpp
template<typename T>
void Safe_Delete_VecList( T& p )
{
	typename T::iterator iter;
	typename T::iterator iterEnd = p.end();

	for( iter = p.begin(); iter != iterEnd; ++iter )
	{
		SAFE_DELETE( (*iter) );
	}

	p.clear();
```

중첩 의존 이름에 대해 명시적으로 타입임을 알려주기 위해 typename을 앞에 붙여준다.

CLayer

```h
#pragma once

#include "../Game.h"

class CLayer
{
private:
	friend class CScene;

private:
	CLayer();

public:
	~CLayer();

private:
	class CScene*	m_pScene;
	string			m_strTag;
	int				m_iZOrder;

public:
	void SetScene( class CScene* pScene )
	{
		m_pScene = pScene;
	}

	void SetTag( const string& strTag )
	{
		m_strTag = strTag;
	}

	void SetZOrder( int iZOrder )
	{
		m_iZOrder = iZOrder;
	}

	CScene* GetScene()
	{
		return m_pScene;
	}

	string GetTag() const
	{
		return m_strTag;
	}

	int GetZOrder() const
	{
		return m_iZOrder;
	}
};
```

생성과 조작은 오직 CScene 클래스를 통해 하도록 한정.

Safe_Delete_VecList를 통해 CScene 클래스에서 레이어를 소멸시키는데 해당 함수가 Game.h에 선언되어 클래스 외부에서 접근하므로 public 선언.

레이어 출력 순서(위치) 결정.

2D 게임은 나중에 출력한 이미지가 이전에 출력한 이미지를 덮게됨.

```cpp
#include "Layer.h"

CLayer::CLayer()
	:
	m_strTag(""),
	m_iZOrder(0),
	m_pScene(nullptr)
{
}

CLayer::~CLayer()
{
}
```

CObject

```h
#pragma once

#include "../Game.h"

class CObject
{
protected:
	CObject();
	virtual ~CObject();

protected:
	POSITION		m_tPos;
	_SIZE			m_tSize;
};
```

```cpp
#include "Object.h"

CObject::CObject()
{
}

CObject::~CObject()
{
}
```

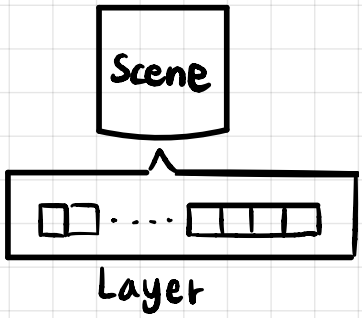
CScene

```h
#pragma once

#include "../Game.h"

class CScene
{
protected:

	friend class CSceneManager;

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

protected:
	list<class CLayer*> m_LayerList;

public:
	class CLayer* CreateLayer( const string& strTag, int iZOrder = 0 );

public:
	virtual bool Init();

public:
	static bool LayerSort( CLayer* pL1, CLayer* pL2 );
};
```

Sort 함수의 인자는 '전역함수'

```cpp
#include "Scene.h"
#include "Layer.h"

CScene::CScene()
{
    CLayer* pLayer = CreateLayer( "UI", INT_MAX );
    pLayer = CreateLayer( "Default" );
}

CScene::~CScene()
{
    Safe_Delete_VecList( m_LayerList );
}

CLayer* CScene::CreateLayer( const string& strTag, int iZOrder )
{
    CLayer* pLayer = new CLayer;

    pLayer->SetTag( strTag );
    pLayer->SetZOrder( iZOrder );
    pLayer->SetScene( this );

    m_LayerList.push_back( pLayer );

    if( m_LayerList.size() >= 2 )
    {
        m_LayerList.sort( CScene::LayerSort );
    }

    return pLayer;
}

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

bool CScene::LayerSort( CLayer* pL1, CLayer* pL2 )
{
    return pL1->GetZOrder() < pL2->GetZOrder();
}
```

레이어 생성과 동시에 zOrder 값에 따라 정렬



각 씬은 기본적으로 UI 레이어, 기본 레이어를 들고 있으며, 게임 내 모든 오브젝트들은 Object 클래스를 상속받게 됨.