26. 콜리젼 클래스

CollisionManager

```h
#pragma once

#include "../Game.h"

class CCollisionManager
{
	DECLARE_SINGLE( CCollisionManager )
};
```

```cpp
#include "CollisionManager.h"

DEFINITION_SINGLE( CCollisionManager )

CCollisionManager::CCollisionManager()
{

}

CCollisionManager::~CCollisionManager()
{

}
```

관리

Collider

```h
#pragma once

#include "../Ref.h"
class CCollider : public CRef
{
protected:
	friend class CObject;

protected:
	CCollider();
	CCollider( const CCollider& coll );
	virtual ~CCollider() = 0;

protected:
	COLLIDER_TYPE	m_eCollType;

public:
	COLLIDER_TYPE GetColliderType()const
	{
		return m_eCollType;
	}

public:
	virtual bool Init() = 0;
	virtual void Input( float fDeltaTime );
	virtual int Update( float fDeltaTime );
	virtual int LateUpdate( float fDeltaTime );
	virtual void Collision( float fDeltaTime );
	virtual void Render( HDC hDC, float fDeltaTime );
	virtual CCollider* Clone() = 0;
};
```

```cpp
#include "Collider.h"

CCollider::CCollider()
{
}

CCollider::CCollider( const CCollider& coll )
{
	*this = coll;
}

CCollider::~CCollider()
{
}

void CCollider::Input( float fDeltaTime )
{
}

int CCollider::Update( float fDeltaTime )
{
	return 0;
}

int CCollider::LateUpdate( float fDeltaTime )
{
	return 0;
}

void CCollider::Collision( float fDeltaTime )
{
}

void CCollider::Render( HDC hDC, float fDeltaTime )
{
}
```

상속

ColliderRect

```h
#pragma once
#include "Collider.h"

class CColliderRect : public CCollider
{
protected:
	friend class CObject;

protected:
	CColliderRect();
	CColliderRect( const CColliderRect& coll );
	virtual ~CColliderRect();

public:
	virtual bool Init();
	virtual void Input( float fDeltaTime );
	virtual int Update( float fDeltaTime );
	virtual int LateUpdate( float fDeltaTime );
	virtual void Collision( float fDeltaTime );
	virtual void Render( HDC hDC, float fDeltaTime );
	virtual CColliderRect* Clone();
};
```

```cpp
#include "ColliderRect.h"

CColliderRect::CColliderRect()
{
	m_eCollType = COLLIDER_TYPE::CT_RECT;
}

CColliderRect::CColliderRect( const CColliderRect& coll )
	:
	CCollider( coll )
{
}

CColliderRect::~CColliderRect()
{
}

bool CColliderRect::Init()
{
	return true;
}

void CColliderRect::Input( float fDeltaTime )
{
}

int CColliderRect::Update( float fDeltaTime )
{
	return 0;
}

int CColliderRect::LateUpdate( float fDeltaTime )
{
	return 0;
}

void CColliderRect::Collision( float fDeltaTime )
{
}

void CColliderRect::Render( HDC hDC, float fDeltaTime )
{
}

CColliderRect* CColliderRect::Clone()
{
	return new CColliderRect(*this);
}
```

Flag.h

```h
#pragma once

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

// Direction
enum MOVE_DIR
{
	MD_BACK = -1,
	MD_NONE,
	MD_FRONT
};

// Collider Type
enum COLLIDER_TYPE
{
	CT_RECT,
	CT_SPHERE,
	CT_LINE,
	CT_POINT,
	CT_PIXEL,
	CT_END
};
```

콜리젼 타입

Object

```h
#pragma once

#include "../Game.h"
#include "../Ref.h"
#include "../Scene/Layer.h"

class CObject : public CRef
{
protected:
	friend class CScene;

protected:
	CObject();
	CObject( const CObject& obj );
	virtual ~CObject();

private:
	static list<CObject*>	m_ObjList;

public:
	static void AddObj( CObject* pObj );
	static CObject* FindObject( const string& strTag );
	static void EraseObj( CObject* pObj );
	static void EraseObj( const string& strTag );
	static void EraseObj();

protected:
	class CScene* m_pScene;
	class CLayer* m_pLayer;

protected:
	string			m_strTag;
	POSITION		m_tPos;
	_SIZE			m_tSize;
	POSITION		m_tPivot;
	class CTexture* m_pTexture;
	list<class CCollider*> m_ColliderList;
```

오브젝트가 여러 콜라이더를 지닐 수 있도록 해줌.

```cpp
CObject::CObject( const CObject& obj )
{
	*this = obj;

	if( m_pTexture )
	{
		m_pTexture->AddRef();
	}

	m_ColliderList.clear();

	list<CCollider*>::const_iterator iter;
	list<CCollider*>::const_iterator iterEnd = m_ColliderList.end();

	for( iter = obj.m_ColliderList.begin(); iter != iterEnd; ++iter )
	{
		CCollider* pColl = (*iter)->Clone();

		m_ColliderList.push_back( pColl );
	}
}
```