18. Bullet

Scene

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

private:
	static unordered_map<string, class CObject*> m_mapPrototype;

public:
	static void ErasePrototype( const string& strTag );
	static void ErasePrototype();
	static CObject* FindPrototype( const string& strKey );

protected:
	list<class CLayer*> m_LayerList;

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

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 );

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

	template <typename T>
	static T* CreatePrototype( const string& strTag )
	{
		T* pObj = new T;

		pObj->SetTag( strTag );

		if( !pObj->Init() )
		{
			SAFE_RELEASE( pObj );
			return nullptr;
		}

		pObj->AddRef();
		m_mapPrototype.insert( make_pair( strTag, pObj ) );

		return pObj;
	}
};
```

기존에 Object 클래스에서 프로토타입을 만들어 주었는데, 이를 Scence 클래스로 옮겨주었다.

InGameScene

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

    CLayer* pLayer = FindLayer( "Default" );

    CPlayer* pPlayer = CObject::CreateObj<CPlayer>( "Player", pLayer );

    SAFE_RELEASE( pPlayer );

    CMinion* pMinion = CObject::CreateObj<CMinion>( "Minion", pLayer );

    SAFE_RELEASE( pMinion );

    // 총알 프로토타입을 만들어 준다.
    CBullet* pBullet = CScene::CreatePrototype<CBullet>( "Bullet" );

    pBullet->SetSize( 50.f, 50.f );

    SAFE_RELEASE( pBullet );

    return true;
}
```

총알 객체 생성

Object

```
CObject* CObject::CreateCloneObj( const string& strPrototypeKey, const string& strTag, CLayer* pLayer )
{
	CObject* pProto = CScene::FindPrototype( strPrototypeKey );

	if( !pProto ) return nullptr;

	CObject* pObj = pProto->Clone();

	pObj->SetTag( strTag );

	if( pLayer )
	{
		pLayer->AddObject( pObj );
	}

	AddObj( pObj );

	return pObj;
}
```

프로토타입 클론

Scene - InGameScene 에서 Player, Minion 오브젝트 생성. Bullet의 프로토타입 생성.

↓

Player, Minion 에서 Bullet의 프로토타입을 클론하여 Bullet 객체 생성.

MoveObject

```
void CMoveObject::MoveAngle()
{
        m_tPos.x += cosf( m_fAngle ) * m_fSpeed;
        m_tPos.y += sinf( m_fAngle ) * m_fSpeed;
}
```

각도에 따라 움직임.

```
void CMoveObject::MoveAngle( float fDeltaTime )
{
        m_tPos.x += cosf( m_fAngle ) * m_fSpeed * fDeltaTime;
        m_tPos.y += sinf( m_fAngle ) * m_fSpeed * fDeltaTime;
}
```

Player

```
void CPlayer::Input( float fDeltaTime )
{
    CMoveObject::Input( fDeltaTime );

    if( GetAsyncKeyState( 'W' ) & 0x8000 )
    {
        MoveYAsSpeed( fDeltaTime, MD_BACK );
    }
    if( GetAsyncKeyState( 'S' ) & 0x8000 )
    {
        MoveYAsSpeed( fDeltaTime, MD_FRONT );
    }
    if( GetAsyncKeyState( 'A' ) & 0x8000 )
    {
        MoveXBySpeed( fDeltaTime, MD_BACK );
    }
    if( GetAsyncKeyState( 'D' ) & 0x8000 )
    {
        MoveXBySpeed( fDeltaTime, MD_FRONT );
    }
    if( GetAsyncKeyState( VK_SPACE ) & 0x8000 )
    {
        Fire();
    }
}
```

```
void CPlayer::Fire()
{
    CObject* pBullet = CMoveObject::CreateCloneObj( "Bullet", "PlayerBullet", m_pLayer );

    pBullet->SetPos( m_tPos.x + m_tSize.x, (m_tPos.y + m_tPos.y + m_tSize.y) / 2.f - pBullet->GetSize().y / 2.f );
}
```

Minion

```
        if( m_fFireTime >= m_fFireLimitTime )
        {
                m_fFireTime -= m_fFireLimitTime;
                Fire();
        }

        return 0;
}

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

void CMinion::Collision( float fDeltaTime )
{
        CMoveObject::Collision( fDeltaTime );
}

void CMinion::Render( HDC hDC, float fDeltaTime )
{
        CMoveObject::Render( hDC, fDeltaTime );
        Rectangle( hDC, m_tPos.x, m_tPos.y, m_tPos.x + m_tSize.x, m_tPos.y + m_tSize.y );
}

CMinion* CMinion::Clone()
{
        return new CMinion( *this );
}

void CMinion::Fire()
{
        CObject* pBullet = CMoveObject::CreateCloneObj( "Bullet", "MinionBullet", m_pLayer );

        ((CMoveObject*)pBullet)->SetAngle( PI );

        pBullet->SetPos( m_tPos.x - pBullet->GetSize().x, (m_tPos.y + m_tPos.y + m_tSize.y) / 2.f - pBullet->GetSize().y / 2.f );
}
```

일정시간 마다 Fire

Bullet

```cpp
#include "Bullet.h"

CBullet::CBullet()
	:
	m_fLimitDist(500.f),
	m_fDist(0.f)
{
}

CBullet::CBullet( const CBullet& bullet )
	:
	CMoveObject( bullet )
{
	m_fLimitDist = bullet.m_fLimitDist;
	m_fDist = bullet.m_fDist;
}

CBullet::~CBullet()
{
}

bool CBullet::Init()
{
	SetSpeed( 500.f );

	return true;
}

int CBullet::Update( float fDeltaTime )
{
	CMoveObject::Update( fDeltaTime );

	MoveAngle( fDeltaTime );

	m_fDist += GetSpeed() * fDeltaTime;

	if( m_fDist >= m_fLimitDist )
	{
		Die();
	}

	return 0;
}

int CBullet::LateUpdate( float fDeltaTime )
{
	CMoveObject::LateUpdate( fDeltaTime );

	return 0;
}
```

최대 거리 다다르면 소멸함.

```cpp
void CBullet::Collision( float fDeltaTime )
{
	CMoveObject::Collision( fDeltaTime );
}

void CBullet::Render( HDC hDC, float fDeltaTime )
{
	CMoveObject::Render( hDC, fDeltaTime );
	Ellipse( hDC, m_tPos.x, m_tPos.y, m_tPos.x + m_tSize.x, m_tPos.y + m_tSize.y );
}

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