20. 플레이어 텍스쳐 출력

Texture.h

```h
#pragma once

#include "../Ref.h"

class CTexture : public CRef
{
private:
	friend class CResourcesManager;

private:
	CTexture();
	~CTexture();

private:
	HDC			m_hMemDC;
	HBITMAP		m_hBitmap;
	HBITMAP		m_hOldBitmap;
	BITMAP		m_tInfo;

public:
	bool LoadTexture( HINSTANCE hInst, HDC hDC, const string& strKey,
		const wchar_t* pFileName, const string& strPathKey = TEXTURE_PATH );
	HDC GetDC() const
	{
		return m_hMemDC;
	}
};
```

Texture.cpp

```cpp
#include "Texture.h"
#include "../Core/PathManager.h"

CTexture::CTexture()
	:
	m_hMemDC( nullptr )
{
}

CTexture::~CTexture()
{
	SelectObject( m_hMemDC, m_hOldBitmap );
	DeleteObject( m_hBitmap );
	DeleteDC( m_hMemDC );
}

bool CTexture::LoadTexture( HINSTANCE hInst, HDC hDC,
	const string& strKey, const wchar_t* pFileName, const string& strPathKey )
{
	// Make memory DC
	m_hMemDC = CreateCompatibleDC( hDC );

	// Make a whole path
	const wchar_t* pPath = GET_SINGLE( CPathManager )->FindPath( strPathKey );

	wstring strPath;
	if( pPath )
		strPath = pPath;

	strPath += pFileName;

	m_hBitmap = (HBITMAP)LoadImage( hInst, strPath.c_str(),
		IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE );

	// Set DC with Bitmap tool
	m_hOldBitmap = (HBITMAP)SelectObject( m_hMemDC, m_hBitmap );

	GetObject( m_hBitmap, sizeof( m_tInfo ), &m_tInfo );

	return true;
}
```

원래 도구로 바꾸어주고 화면 DC와 달리 메모리 DC는 DeleteDC 함수를 통해 해제함.

경로+파일 이름

기존 dc의 도구를 리턴함.

이미지를 로드할 때 이미지 픽셀 정보를 저장해둘 공간이 필요하다. 따라서 이를 메모리 dc에 저장하게 된다.

메모리 dc란, 화면 dc를 이용해 이와 같은 성질을 갖는 메모리 상의 dc를 만들어 이 메모리 dc에 픽셀 정보를 담아 도장 처럼 만들어 찍어내는 것이다.

dc들은 각자 자신의 그리기 도구를 가지고 있다.
그리고 이 dc에 어떤 그리기 도구를 세팅하느냐에 따라 보여지는 것이 달라지는 것이다.

여기서 사용하는 그리기 도구가 HBITMAP 이며, 원래 dc가 가지고 있던 도구를 새 도구로 변경할 경우 dc 해제 시에 원래 도구로 바꾸어주고 해제해야 한다.

Game.h

```h
#pragma once

#include <Windows.h>
#include <list>
#include <vector>
#include <unordered_map>
#include <crtdbg.h>

using namespace std;

#include "resource.h"
#include "Macro.h"
#include "Types.h"
#include "Flag.h"

#pragma comment(lib, "msimg32")
```

→ window에서 기본적으로 제공해주는 이미지 파일 라이브러리를 코드를 통해 링킹해줌.

ResourceManager.h

```h
#pragma once

#include "../Game.h"

class CResourcesManager
{
private:
	unordered_map<string, class CTexture*> m_mapTexture;
	HINSTANCE	m_hInst;
	HDC			m_hDC;

public:
	bool Init( HINSTANCE hInst, HDC hDC );
	class CTexture* LoadTexture( const string& strKey,
		const wchar_t* pFileName, const string& strPathKey = TEXTURE_PATH );
	CTexture* FindTexture( const string& strKey );

	DECLARE_SINGLE( CResourcesManager )
};
```

ResourceManager.cpp

```cpp
CResourcesManager::CResourcesManager()
{
}

CResourcesManager::~CResourcesManager()
{
	Safe_Release_Map( m_mapTexture );
}

bool CResourcesManager::Init( HINSTANCE hInst, HDC hDC )
{
	m_hInst = hInst;
	m_hDC = hDC;

	return true;
}

CTexture* CResourcesManager::LoadTexture( const string& strKey, const wchar_t* pFileName, const string& s
{
	CTexture* pTexture = FindTexture( strKey );

	if( pTexture != nullptr ) return pTexture;

	pTexture = new CTexture;

	if( !pTexture->LoadTexture( m_hInst, m_hDC, strKey, pFileName, strPathKey ) )
	{
		SAFE_RELEASE( pTexture );
		return nullptr;
	}

	pTexture->AddRef();
	m_mapTexture.insert( make_pair( strKey, pTexture ) );

	return pTexture;
}

CTexture* CResourcesManager::FindTexture( const string& strKey )
{
	unordered_map<string, CTexture*>::iterator iter = m_mapTexture.find( strKey );

	if( iter == m_mapTexture.end() )
		return nullptr;

	iter->second->AddRef();

	return iter->second;
}
```

텍스쳐 클래스에서 필요하기 때문에 미리 받아줌.

텍스쳐 객체를 리턴해 외부에서 받기 때문에 미리 레퍼런스 카운트를 증가시켜줌.

Core.cpp

```cpp
CCore::~CCore()
{
	DESTROY_SINGLE( CSceneManager );
	DESTROY_SINGLE( CResourcesManager );
	DESTROY_SINGLE( CPathManager );
	DESTROY_SINGLE( CTimer );

	ReleaseDC( m_hWnd, m_hDC );
}
```

```cpp
// 리소스 관리자 초기화
if( !GET_SINGLE( CResourcesManager )->Init( m_hInst, m_hDC ) )
{
	return false;
}
```

리소스 관리자 초기화 호출

Player.cpp

```cpp
void CPlayer::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 );
}
```

Object.cpp

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

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

shallow copy 하고
texture에 대한 레퍼런스 카운트 증가

호출

```cpp
void CObject::SetTexture( CTexture* pTexture )
{
        SAFE_RELEASE( m_pTexture );
        m_pTexture = pTexture;

        if( pTexture )
                pTexture->AddRef();
}

void CObject::SetTexture( const string& strKey, const wchar_t* pFileName, const string& strPathKey )
{
        SAFE_RELEASE( m_pTexture );
        m_pTexture = GET_SINGLE( CResourcesManager )->LoadTexture( strKey, pFileName, strPathKey );
        i[illegible] = 0;
}
```

```cpp
void CObject::Render( HDC hDC, float fDeltaTime )
{
       if( m_pTexture )
       {
               BitBlt( hDC, m_tPos.x, m_tPos.y, m_tSize.x, m_tSize.y, m_pTexture->GetDC(), 0, 0, SRCCOPY
       }
}
```

이미지 출력 함수