10. 기본 프레임워크 완성

```h
#pragma once

#include "Game.h"

class CCore
{
private:
	static CCore*	m_pInst;

public:
	static CCore* GetInst()
	{
		if( !m_pInst )
			m_pInst = new CCore;
		return m_pInst;
	}

	static void DestroyInst()
	{
		SAFE_DELETE( m_pInst );
	}

private:
	CCore();
	~CCore();

private:
	static bool m_bLoop;

private:
	HINSTANCE	m_hInst;
	HWND		m_hWnd;
	RESOLUTION	m_tRs;

public:
	bool Init( HINSTANCE hInst );
	int Run();

private:
	ATOM MyRegisterClass();
	BOOL Create();

public:
	static LRESULT CALLBACK WndProc( HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam );
};
```

```cpp
#include "Core.h"

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

CCore::CCore()
{
}

CCore::~CCore()
{
}

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

	// 윈도우 클래스 등록
	MyRegisterClass();

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

	// 윈도우 창 생성
	Create();

	return true;
}

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
		{

		}
	}

	return (int)msg.wParam;
}

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
	m_hWnd = CreateWindowW( L"WINAPIFRAMEWORK", L"WINAPIFRAMEWORK",
		WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, nullptr, nullptr, m_hInst, nullpt

	if( !m_hWnd )
	{
		return FALSE;
	}

	// 실제 윈도우 타이틀바나 메뉴를 포함한 윈도우 크기를 구해옴.
	RECT rc = { 0,0,m_tRs.iW,m_tRs.iH };
	AdjustWindowRect( &rc, WS_OVERLAPPEDWINDOW, FALSE );

	// 위에서 구해온 전체 윈도우 크기를 통해 클라이언트 영역의 크기를 원하는 크기로 맞춰줘야 함.
	SetWindowPos( m_hWnd, HWND_TOPMOST, 100, 100, rc.right - rc.left, rc.bottom - rc.top, SWP_NOMO

	ShowWindow( m_hWnd, SW_SHOW );
	UpdateWindow( m_hWnd );

	return TRUE;
}

LRESULT CCore::WndProc( HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam )
{
	switch( message )
	{
		case WM_PAINT:
		{
			PAINTSTRUCT ps;
			HDC hdc = BeginPaint( hWnd, &ps );
			// TODO: Add any drawing code that uses hdc here...

			EndPaint( hWnd, &ps );
		}
		break;
		// 윈도우를 종료시킬 때 들어오는 메시지.
		case WM_DESTROY:
			m_bLoop = false;
			PostQuitMessage( 0 );
			break;
		default:
			return DefWindowProc( hWnd, message, wParam, lParam );
		}
	return 0;
}
```

윈도우 클래스를 등록하는 함수에서
윈도우 클래스에 윈도우 프로시져를
넘겨줄 때 전역 함수인 윈도우 프로시져만
등록할 수 있다.
따라서 전역 함수로 취급되는 static 으로
선언해 주었다.
또 m_bLoop의 경우 윈도우 프로시져
함수에서 조작하기 때문에 마찬가지로
전역 변수 취급하는 static 으로 선언해주었다.

* static 함수를 단순히 클래스 네임스페이스를
통해 접근할 수 있는 이유는 static 멤버 함수
호출시에 내부적으로 this 포인터가 넘어가지
않기 때문.
그런데 일반 멤버 변수의 경우 내부적으로
this 포인터를 사용하기 때문에 static 멤버 함수
에서 사용할 수 없는 이유이다.

resource.h

```h
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by APIFramework.rc
//
#define IDI_ICON1                       101

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        102
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1001
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

프로젝트에서 리소스를 추가하면
자동으로 이 헤더에 해당 리소스들이
추가됨.
그리고 등록된 자원을 활용할 수 있음.

main.h

```cpp
#include <Windows.h>

#include "Core.h"

int APIENTRY wWinMain( _In_ HINSTANCE hInstance,
    _In_opt_ HINSTANCE hPrevInstance,
    _In_ LPWSTR    lpCmdLine,
    _In_ int       nCmdShow )
{
    // ÃÊ±âÈ ½ÇÆÐ½Ã 초기화 과정에서 오류가 난 경우
    if( !CCore::GetInst()->Init( hInstance ) )
    {
        CCore::DestroyInst();
        return 0;
    }

    int iRev = CCore::GetInst()->Run();

    CCore::DestroyInst();

    return iRev;
}
```

types.h

```h
#pragma once

typedef struct _tagResolution
{
	unsigned int	iW;
	unsigned int	iH;
}RESOLUTION, *PRESOLUTION;
```

따로 헤더 만들어서
커스텀 타입들 관리