게임프로그래밍

첫 번째 예제

박종승

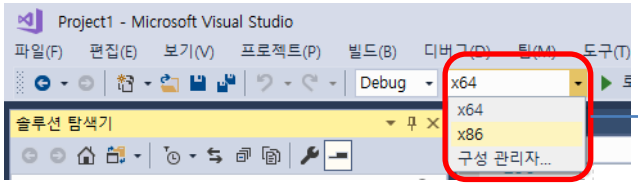
Dept. of CSE, Incheon Nat. Univ.
jong@inu.ac.kr
http://ecl.inu.ac.kr

목차

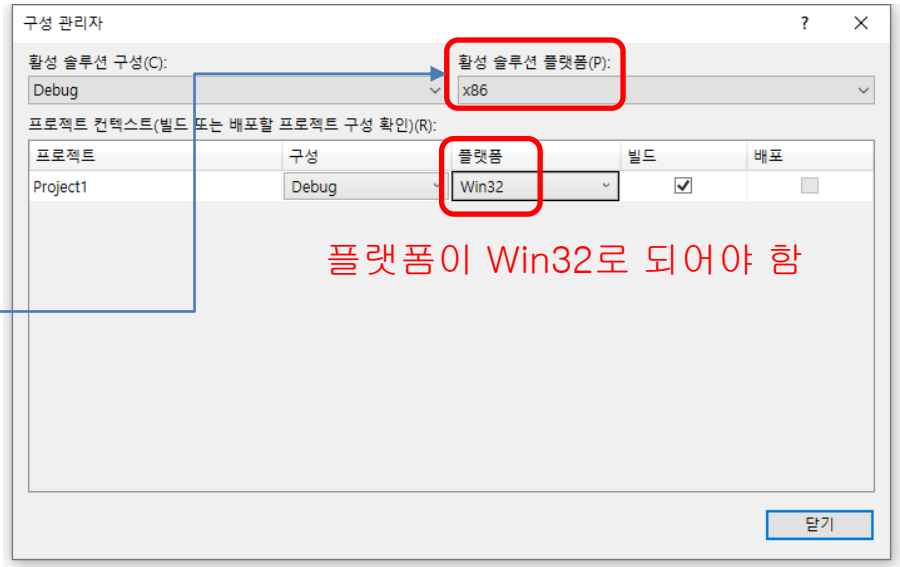
소스코드 빌드 환경 참고

- 샘플 소스코드는 32비트용 Win32 함수들을 사용하고 있음
  - SetWindowLongPtrW, GetWindowLongPtrW

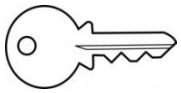
- 따라서 32비트용으로 빌드해야 함
  - 툴바의 "Debug"로 되어 있는 "솔루션 구성" 드롭다운리스트의 바로 오른쪽에 있는 "솔루션 플랫폼" 드롭다운리스트에서 "x64"가 아니라 "x86"을 선택하면 됨



활성 솔루션 플랫폼 설정



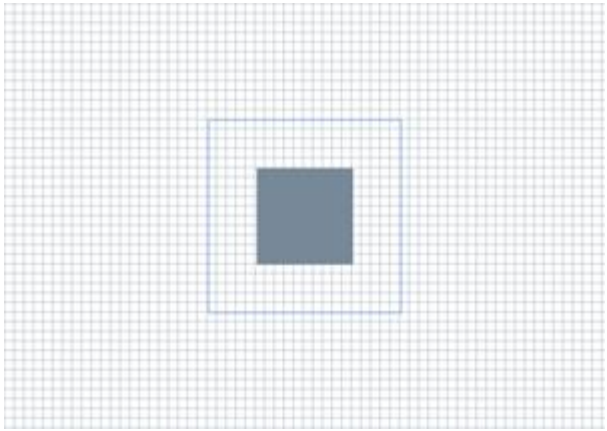
플랫폼이 Win32로 되어야 함



단순한 D2D 응용 만들기

01.DrawRectangle

- 기능
  - 한 격자와 두 사각형을 D2D로 그리는 응용
- 응용 만들기
  - Visual Studio로 Win32 프로젝트를 생성
  - 소스파일(cpp 파일과 h 파일)을 수정



- 단계 1: 헤더파일 작성하기
- 단계 2: 클래스 기초작업 구현
- 단계 3: D2D 자원을 생성
- 단계 4: 내용을 그리기

단계 1: 헤더파일 작성하기

- 1.1: DemoApp.h 파일에 자주 사용하는 헤더를 포함

```
// Windows Header Files:
#include <windows.h>

// C RunTime Header Files:
#include <stdlib.h>
#include <malloc.h>
#include <memory.h>
#include <wchar.h>
#include <math.h>

#include <d2d1.h>
#include <d2d1helper.h>
#include <dwrite.h>
#include <wincodec.h>
```

단계 1: 헤더파일 작성하기'

- 1.2: 자원을 반납하는 함수, 매크로, 에러 처리 함수 등을 선언

```
#define SAFE_RELEASE(p) { if(p) { (p)->Release(); (p)=NULL; } }

template<typename Interface>
inline void SafeRelease( Interface **ppInterfaceToRelease )
{
    if (*ppInterfaceToRelease != NULL) {
        (*ppInterfaceToRelease)->Release();
        (*ppInterfaceToRelease) = NULL;
    }
}

#ifndef Assert
#if defined( DEBUG ) || defined( _DEBUG )
#define Assert(b) do {if (!(b)) {OutputDebugStringA("Assert: " #b "₩n");}} while(0)
#else
#define Assert(b)
#endif //DEBUG || _DEBUG
#endif

#ifndef HINST_THISCOMPONENT
EXTERN_C IMAGE_DOS_HEADER __ImageBase;
#define HINST_THISCOMPONENT ((HINSTANCE)&__ImageBase)
#endif
```

단계 1: 헤더파일 작성하기”

- 1.3: 클래스의 함수들을 선언함

```
class DemoApp
{
public:
    DemoApp();
    ~DemoApp();
    HRESULT Initialize(); // 윈도우 클래스를 등록, 그리기 자원들을 생성하는 함수들을 호출
    void RunMessageLoop(); // 메시지를 처리

private:
    HRESULT CreateDeviceIndependentResources(); // 장치 독립 자원들을 초기화
    HRESULT CreateDeviceResources(); // 장치 의존 자원들을 초기화
    void DiscardDeviceResources(); // 장치 의존 자원들을 반납
    HRESULT OnRender(); // 내용을 그리기
    void OnResize( UINT width, UINT height ); // 렌더타겟을 resize
    static LRESULT CALLBACK WndProc( HWND hWnd, UINT message, // 윈도우 프로시져
                                     WPARAM wParam, LPARAM lParam );
};
```

단계 1: 헤더파일 작성하기'''

- 1.4: 클래스의 변수들을 선언함

```
private:
HWND m_hwnd;
ID2D1Factory* m_pDirect2dFactory;
ID2D1HwndRenderTarget* m_pRenderTarget;
ID2D1SolidColorBrush* m_pLightSlateGrayBrush;
ID2D1SolidColorBrush* m_pCornflowerBlueBrush;
```

단계 2: 클래스 기초작업 구현

- 2.1: 클래스 생성자와 소멸자를 구현

```
// 생성자에서는 클래스 변수들을 NULL로 초기화
DemoApp::DemoApp() :
    m_hwnd(NULL),
    m_pDirect2dFactory(NULL),
    m_pRenderTarget(NULL),
    m_pLightSlateGrayBrush(NULL),
    m_pCornflowerBlueBrush(NULL)
{
}

// 소멸자에서는 모든 인터페이스들을 반납
DemoApp::~DemoApp()
{
    SafeRelease(&m_pDirect2dFactory);
    SafeRelease(&m_pRenderTarget);
    SafeRelease(&m_pLightSlateGrayBrush);
    SafeRelease(&m_pCornflowerBlueBrush);
}
```

단계 2: 클래스 기초작업 구현'

- 2.2: 메시지루프 함수 구현

```
void DemoApp::RunMessageLoop()
{
    MSG msg;

    while (GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
}
```

단계 2: 클래스 기초작업 구현”

- 2.3: 초기화 함수 구현.

```
HRESULT DemoApp::Initialize() {
    HRESULT hr = CreateDeviceIndependentResources(); //장치 독립적 자원들을 초기화함
    if (SUCCEEDED(hr)) {
        WNDCLASSEX wcex = { sizeof(WNDCLASSEX) };
        wcex.style=CS_HREDRAW|CS_VREDRAW;  wcex.lpfnWndProc=DemoApp::WndProc;
        wcex.cbClsExtra=0;  wcex.cbWndExtra=sizeof(LONG_PTR);
        wcex.hInstance=HINST_THISCOMPONENT;  wcex.hbrBackground=NULL;
        wcex.lpszMenuName=NULL;  wcex.hCursor=LoadCursor(NULL, IDI_APPLICATION);
        wcex.lpszClassName = L"D2DDemoApp";
        RegisterClassEx(&wcex); // 윈도우 클래스를 등록함
        FLOAT dpiX, dpiY;
        m_pDirect2dFactory->GetDesktopDpi(&dpiX, &dpiY); //현재의 시스템 DPI를 얻음
        m_hwnd = CreateWindow( // 윈도우를 생성함
            L"D2DDemoApp", L"Direct2D Demo App",
            WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT,
            static_cast<UINT>(ceil(640.f * dpiX / 96.f)),
            static_cast<UINT>(ceil(480.f * dpiY / 96.f)),
            NULL, NULL, HINST_THISCOMPONENT, this );
        hr = m_hwnd ? S_OK : E_FAIL;
        if (SUCCEEDED(hr)) {
            ShowWindow(m_hwnd, SW_SHOWNORMAL);
            UpdateWindow(m_hwnd);
        }
    }
    return hr;
}
```

단계 2: 클래스 기초작업 구현'''

- 2.4: WinMain 함수 구현.

```
int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    LPSTR lpCmdLine, int nCmdShow )
{
    // 이 프로세스가 사용하는 힙에 에러가 발생하면 이 프로세스를 종료하도록 명시함.
    HeapSetInformation(NULL, HeapEnableTerminationOnCorruption, NULL, 0);

    if (SUCCEEDED(CoInitialize(NULL))) {
        {
            DemoApp app;
            if (SUCCEEDED(app.Initialize())) {  // DemoApp 클래스의 instance 초기화
                app.RunMessageLoop();  // 메시지루프 시작
            }
        }
        CoUninitialize();
    }
    return 0;
}
```

단계 3: D2D 자원을 생성

- 3.1: CreateDeviceIndependentResources 함수 구현.

```
HRESULT DemoApp::CreateDeviceIndependentResources()
{
    // 장치 독립적 자원들을 생성함

    HRESULT hr = S_OK;

    // Create a Direct2D factory.
    hr = D2D1CreateFactory(D2D1_FACTORY_TYPE_SINGLE_THREADED, &m_pDirect2dFactory);

    return hr;
}
```

단계 3: D2D 자원을 생성'

- 3.2: CreateDeviceResources 함수 구현.

```
HRESULT DemoApp::CreateDeviceResources()
{
    // 장치 의존적 자원들, 하나의 렌더타겟, 두개의 브러시를 생성함
    HRESULT hr = S_OK;
    if  (!m_pRenderTarget) { // 렌더타겟이 이미 유효하면 실행하지 않음
        // 클라이언트 영역의 크기를 얻음.
        RECT rc;  GetClientRect(m_hwnd, &rc);
        D2D1_SIZE_U size = D2D1::SizeU( rc.right - rc.left, rc.bottom - rc.top );

        // D2D 렌더타겟을 생성함.
        hr = m_pDirect2dFactory->CreateHwndRenderTarget(
            D2D1::RenderTargetProperties(),
            D2D1::HwndRenderTargetProperties(m_hwnd, size), //클라이언트 영역과 동일 크기로.
            &m_pRenderTarget );
        if (SUCCEEDED(hr)) {
            hr = m_pRenderTarget->CreateSolidColorBrush(  // 회색 브러시 생성
                D2D1::ColorF(D2D1::ColorF::LightSlateGray), &m_pLightSlateGrayBrush );
        }
        if (SUCCEEDED(hr)) {
            hr = m_pRenderTarget->CreateSolidColorBrush(  //파랑색 브러시 생성
                D2D1::ColorF(D2D1::ColorF::CornflowerBlue), &m_pCornflowerBlueBrush );
        }
    }
    return hr;
}
```

단계 3: D2D 자원을 생성”

- 3.3: DiscardDeviceResources 함수 구현.

```
void DemoApp::DiscardDeviceResources()
{
    // CreateDeviceResources 함수에서 생성한 모든 자원들을 반납
    SafeRelease(&m_pRenderTarget);
    SafeRelease(&m_pLightSlateGrayBrush);
    SafeRelease(&m_pCornflowerBlueBrush);
}
```

단계 4: 내용을 그리기

- 4.1: 윈도우 메시지 처리를 위한 WndProc 함수 구현

```
LRESULT CALLBACK DemoApp::WndProc(HWND hwnd, UINT message,
                                  WPARAM wParam, LPARAM lParam)
{
    LRESULT result = 0;
    if (message == WM_CREATE) {
        LPCREATESTRUCT pcs = (LPCREATESTRUCT)lParam;
        DemoApp *pDemoApp = (DemoApp *)pcs->lpCreateParams;
        ::SetWindowLongPtrW( hwnd, GWLP_USERDATA, PtrToUlong(pDemoApp) );
        result = 1;
    } else {
        DemoApp *pDemoApp = reinterpret_cast<DemoApp *>(static_cast<LONG_PTR>(
            ::GetWindowLongPtrW( hwnd, GWLP_USERDATA )));
        bool wasHandled = false;
        if (pDemoApp) {
            switch (message) {
            /*case WM_SIZE: .... 이 부분은 다음 장에.... */
            }//switch
        }
        if (! wasHandled)
            result = DefWindowProc(hwnd, message, wParam, lParam);
    }
    return result;
}
```

단계 4: 내용을 그리기'

- 4.1: WndProc 함수 구현'

```
switch (message) {
case WM_SIZE:
    {  UINT width = LOWORD(lParam);  // 새 창의 크기
       UINT height = HIWORD(lParam);
       pDemoApp->OnResize(width, height);  // 창의 크기를 다시 조절
    }
    result = 0;  wasHandled = true;  break;
case WM_DISPLAYCHANGE:
    {  InvalidateRect(hwnd, NULL, FALSE); //WM_PAINT를 발생시킴
    }
    result = 0;  wasHandled = true;  break;
case WM_PAINT:
    {  pDemoApp->OnRender();  // 창을 그림
       ValidateRect(hwnd, NULL);
    }
    result = 0;  wasHandled = true;  break;
case WM_DESTROY:
    {  PostQuitMessage(0);
    }
    result = 1;  wasHandled = true;  break;
}//switch
```

단계 4: 내용을 그리기”

- 4.2: OnRender함수 구현

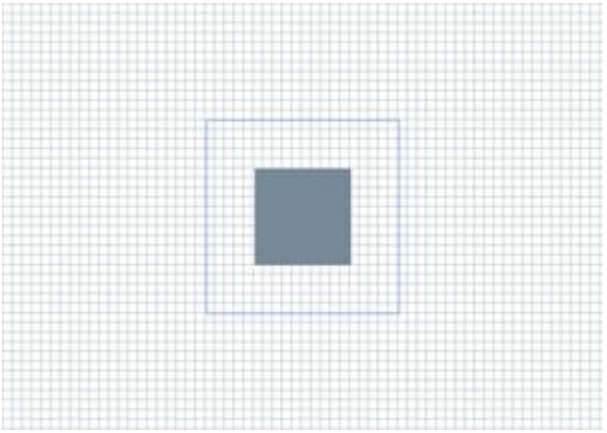
```
HRESULT DemoApp::OnRender()
{
    HRESULT hr = S_OK;
    hr = CreateDeviceResources(); // 항상 호출되며, 렌더타겟이 유효하면 아무일도 하지 않음.
    if (SUCCEEDED(hr)) { //렌더타겟이 유효함.
        m_pRenderTarget->BeginDraw(); //그리기 시작.
        m_pRenderTarget->SetTransform(D2D1::Matrix3x2F::Identity()); //변환행렬을 항등행렬로.
        m_pRenderTarget->Clear(D2D1::ColorF(D2D1::ColorF::White)); // 창을 클리어.

        D2D1_SIZE_F rtSize = m_pRenderTarget->GetSize(); //그리기 영역의 크기를 얻음.

        // 배경 격자를 그림.
        int width = static_cast<int>(rtSize.width);
        int height = static_cast<int>(rtSize.height);
        for (int x = 0; x < width; x += 10) {
            m_pRenderTarget->DrawLine(  D2D1::Point2F(static_cast<FLOAT>(x), 0.0f),
D2D1::Point2F(static_cast<FLOAT>(x), rtSize.height),  m_pLightSlateGrayBrush, 0.5f );
        }
        for (int y = 0; y < height; y += 10) {
            m_pRenderTarget->DrawLine(  D2D1::Point2F(0.0f, static_cast<FLOAT>(y)),
D2D1::Point2F(rtSize.width, static_cast<FLOAT>(y)),  m_pLightSlateGrayBrush, 0.5f );
        }
        // 계속....
```

단계 4: 내용을 그리기‴

- 4.2: OnRender함수 구현′



```
        // 화면 중간에 두 사각형을 그림.
        D2D1_RECT_F rectangle1 = D2D1::RectF(
            rtSize.width/2 - 50.0f, rtSize.height/2 - 50.0f,
            rtSize.width/2 + 50.0f, rtSize.height/2 + 50.0f );
        D2D1_RECT_F rectangle2 = D2D1::RectF(
            rtSize.width/2 - 100.0f, rtSize.height/2 - 100.0f,
            rtSize.width/2 + 100.0f, rtSize.height/2 + 100.0f );
        // 첫번째 사각형의 내부를 회색 브러시로 채워서 그림.
        m_pRenderTarget->FillRectangle(&rectangle1, m_pLightSlateGrayBrush);
        // 두번째 사각형의 외곽선을 파랑색 브러시로 그림.
        m_pRenderTarget->DrawRectangle(&rectangle2, m_pCornflowerBlueBrush);
        hr = m_pRenderTarget->EndDraw();  //그리기를 수행함. 성공하면 S_OK를 리턴함.
    }

    if (hr == D2DERR_RECREATE_TARGET) { // 렌더타겟을 재생성해야 함.
        // 실행중에 D3D 장치가 소실된 경우,
        hr = S_OK;
        DiscardDeviceResources(); // 장치 의존 자원들을 반납함.
        // 다음번 OnRender 함수가 호출될 때에 재생성함.
    }
    return hr;
}
```

단계 4: 내용을 그리기""

- 4.3: OnResize 함수 구현

```
void DemoApp::OnResize(UINT width, UINT height)
{
    // 창이 resize될 경우에 렌더타겟도 이에 맞도록 resize함.
    if (m_pRenderTarget) {
        m_pRenderTarget->Resize(D2D1::SizeU(width, height));
        // Resize함수가 실패할 수도 있으나, 예외처리하지 않았음.
        // 왜냐하면, 실패하는  경우에, 다음번 EndDraw 호출에서도 실패할 것이므로.
    }
}
```