게임프로그래밍

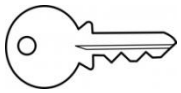
윈도우 프로그래밍

박종승

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

목차

- Win32 프로그래밍
- 문자열 표시하기
- Win32 프로그래밍 팁
- GDI 사용하기



윈도우 프로그래밍

- 윈도우 프로그래밍
  - 윈도우 데스크톱 응용 프로그램을 만드는 과정
  - Win32 API(저수준 C 함수) vs. MFC API(고수준 C++ 클래스)

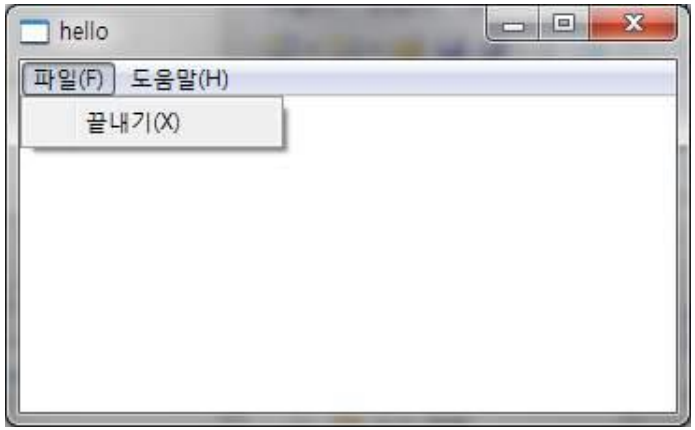
- Win32는 왜 알아야 하나?
  - 어떤 윈도우 application을 작성하기 위해서는 Win32/MFC를 알아야 함
  - DirectX를 이용하기 위한 최소한의 code 이해가 필요함

- Window 응용 프로그램의 모드
  - 윈도우 모드
    - 다른 프로그램과 리소스들을 공유
  - 전체화면 모드
    - 윈도우로부터 하드웨어를 완전히 독점

디폴트 Win32 응용프로그램 만들기

01.WindowsProject1

- 새 프로젝트 만들기: 디폴트 Win32 응용프로그램
  - 1. 프로젝트 생성 시에 Windows 응용 프로그램을 생성함
    - 프로젝트 형식을 'Win32/Win32 프로젝트'으로 선택하여 프로젝트 생성
    - 새 프로젝트를 생성하면 Visual Studio가 기본적인 프레임에 대한 디폴트 파일들을 자동 생성함
      - 소스(cpp/h) 파일: 'hello.cpp', 'hello.h', 'stdafx.cpp', 'stdafx.h', 'Resource.h'
      - 리소스 파일: 'hello.ico', 'hello.rc', 'small.ico'
    - 특히 'hello. cpp'에는 상당히 긴 소스코드들이 추가되어 있음
      - 기본적인 메뉴바와 도움말 대화상자를 구성하기 위함
  - 2. 응용에 필요한 코드를 추가
- 실습: 윈도우 응용 프로그램을 위한 프로젝트를 만들어보자.
  - 템플릿 'Windows 데스크톱 응용 프로그램 ' 으로 프로젝트 생성
  - 생성된 모든 파일들을 살펴보자.



최소한의 Win32 응용프로그램 만들기

02.HelloEmpty

- 실습: 빈 프로젝트에서부터 만들어보기
  - '빈 프로젝트' 템플릿을 선택하여 빈 프로젝트를 만들자.
  - 생성된 모든 파일들을 살펴보자.

- 최소한의 코드를 추가해보자.
  - 새 파일 추가 : HelloEmpty.cpp

- 윈도우 응용 프로그램의 필수 함수
  - Entry point: WinMain( )
  - Window procedure: 예: WinProc( )

최소한의 Win32 응용프로그램 만들기'

02.HelloEmpty

```
#include <windows.h>
HWND MainWindowHandle = 0;

Bool InitWindowsApp(HINSTANCE instanceHandle, int show);
int  Run();
LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam);

int WINAPI WinMain(
        HINSTANCE hInstance, HINSTANCE hPrevInstance,
        PSTR pCmdLine, int nShowCmd)
{
        if (! InitWindowsApp(hInstance, nShowCmd))
        {
                ::MessageBox(0, "Init - Failed", "Error", MB_OK);
                return 0;
        }
        return Run();
}
```

최소한의 Win32 응용프로그램 만들기”

```
Bool InitWindowsApp(HINSTANCE instanceHandle, int show)
{
        WNDCLASS wc;
        wc.style          = CS_HREDRAW | CS_VREDRAW;
        wc.lpfnWndProc    = WndProc;
        wc.cbClsExtra     = 0;
        wc.cbWndExtra     = 0;
        wc.hInstance      = instanceHandle;
        wc.hIcon          = ::LoadIcon(0, IDI_APPLICATION);
        wc.hCursor        = ::LoadCursor(0, IDC_ARROW);
        wc.hbrBackground  = static_cast<HBRUSH>(::GetStockObject(WHITE_BRUSH));
        wc.lpszMenuName   = 0;
        wc.lpszClassName  = "Hello";
        if (!::RegisterClass(&wc)) {
                ::MessageBox(0, "RegisterClass - Failed", 0, 0);
                return false;
        }
        MainWindowHandle = ::CreateWindow(
                "Hello", "Hello", WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,
                0, 0, instanceHandle, 0);
        if (MainWindowHandle == 0) {
                ::MessageBox(0, "CreateWindow - Failed", 0, 0);
                return false;
        }
        ::ShowWindow(MainWindowHandle, show);
        ::UpdateWindow(MainWindowHandle);
        return true;
}
```

최소한의 Win32 응용프로그램 만들기'''

```
int Run()
{
        MSG msg;
        ::ZeroMemory(&msg, sizeof(MSG));

        while (::GetMessage(&msg, 0, 0, 0) ) {
                ::TranslateMessage(&msg);
                ::DispatchMessage(&msg);
        }
        return msg.wParam;
}

LRESULT CALLBACK WndProc(HWND   windowHandle,
        UINT msg, WPARAM wParam, LPARAM lParam)
{
        switch (msg) {
        case WM_KEYDOWN:
                if ( wParam == VK_ESCAPE )
                        ::DestroyWindow(MainWindowHandle);
                return 0;
        case WM_DESTROY:
                ::PostQuitMessage(0);
                return 0;
        }
        return ::DefWindowProc(windowHandle, msg, wParam, lParam);
}
```

Note: GetMessage vs. PeekMessage

문자열 표시하기

- 목차
  - 디버그 용도의 문자열을 출력창에 표시하기
  - 간단한 텍스트 문자열을 대화 상자에 표시하기
  - 문자 집합과 문자열 조작

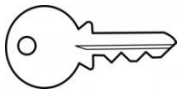
디버그 메시지를 표시하기

03.HelloTrace

- `printf`를 사용한 디버깅?
  - 콘솔 프로그램: no problem
  - MFC 프로그램: `TRACE` (디버그 모드로 실행 시의 출력창에서만)
  - Win32 프로그램: ??
- Win32에서 TRACE의 사용

```
void TRACE(LPCTSTR lpszFormat, ...) {
        TCHAR lpszBuffer[0x160];
        va_list fmtList;
        va_start( fmtList, lpszFormat );
        _vstprintf( lpszBuffer, lpszFormat, fmtList );
        va_end( fmtList );
        ::OutputDebugString( lpszBuffer );
}
```

- 실습
  - 마우스 버튼(왼쪽)을 누르면 버튼 좌표를 디버그 출력창으로 출력하는 코드를 작성하시오.



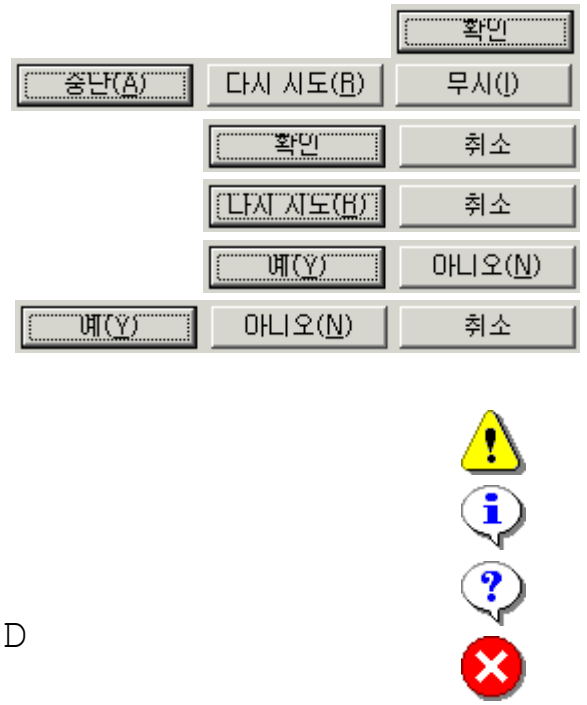
간단한 텍스트 문자열을 화면에 표시하기

- 간단한 메시지 상자: `MessageBox` 함수
  - int MessageBox(HWND hWnd, LPCTSTR lpText, LPCTSTR lpCaption, UINT uType);
  - 인자값
    - 인자1: HWND hWnd: 소유주 윈도우의 handle임. NULL로 지정하면 이 대화상자는 소유주 없음.
    - 인자2: LPCTSTR lpText: 표시할 메시지
    - 인자 3: LPCTSTR lpCaption: 대화상자 타이틀. NULL로 지정하면 디폴트값인 "Error"가 사용됨.
    - 인자 4: UINT uType: 대화상자의 내용과 행위(push buttons, icon, default button) 명시. 디폴트는 "MB_OK"임.
  - 리턴값
    - 선택된 버튼에 따라서 해당 int 값을 리턴함
      - IDABORT, IDCANCEL, IDCONTINUE, IDIGNORE, IDNO, IDOK, IDRETRY, IDTRYAGAIN, IDYES
      - Cancel 버튼을 누르는 대신 ESC 키를 눌러도 됨

간단한 텍스트 문자열을 화면에 표시하기'

04.HelloMessageBox

- uType
  - push buttons (*는 디폴트를 의미)
    - MB_OK (OK*)
    - MB_ABORTRETRYIGNORE (Abort*, Retry, Ignore)
    - MB_OKCANCEL (OK*, Cancel)
    - MB_RETRYCANCEL (Retry*, Cancel)
    - MB_YESNO (Yes*, No)
    - MB_YESNOCANCEL (Yes*, No, Cancel)
  - icon
    - MB_ICONEXCLAMATION, MB_ICONWARNING
    - MB_ICONINFORMATION, MB_ICONASTERISK
    - MB_ICONQUESTION
    - MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND
  - default button
    - MB_DEFBUTTON1 (default), ..., MB_DEFBUTTON4
- 예제
  - 마우스 버튼(왼쪽, 오른쪽)을 누르면 메시지상자를 표시



문자 집합

- 문자 집합(character set)
  - 문자 인코딩을 명시: ANSI, Unicode 등
  - 이로 인한 에러 발생을 해결할 수 있어야 함
- 문자 집합 설정
  - 디폴트: 유니코드
  - 프로젝트 속성 대화 상자: '구성 속성' > '일반' > '문자 집합'
    - 설정 안 함 : 상수 정의하지 않음 (SBCS로 동작됨)
    - 유니코드 문자 집합 사용 : '_UNICODE' 상수를 정의함
    - 멀티바이트 문자 집합 사용 : '_MBCS' 상수를 정의함
- 문자 집합의 종류
  - SBCS
  - MBCS
  - Unicode

문자 집합'

- 표현
  - SBCS, MBCS는 char (CHAR)
  - Unicode는 wchar_t (WCHAR)

- 타입들
  - CHAR : char
  - LPSTR : char*
  - LPCSTR : const char*

  - WCHAR : wchar_t
  - LPWSTR : wchar_t*
  - LPCWSTR : const wchar_t*

  - TCHAR : CHAR 또는 WCHAR
  - LPTSTR : TCHAR*
  - LPCTSTR : const TCHAR*

문자열 입력 함수

- 유니코드용 함수 집합
  - printf vs. wprintf
  - str*xxx* vs. wcs*xxx* (eg. strcpy vs. wcscpy)
- 문자열 상수
  - "L" : 유니코드 문자열임을 명시
  - Eg. Strlen("John") vs wcslen(L"홍길동")
- 문자집합에 따른 윈도우 함수의 두 분류
  - 윈도우 함수 : MBCS(SBCS 포함) 또는 유니코드 의 두 종류로 분류됨
    - 기준: 컴파일 시에 '_MBCS' 상수 정의 또는 '_UNICODE' 상수 정의 여부
  - Eg. Win32 함수인 SetWindowsText는 내부적으로
    - MBCS의 경우에는 SetWindowsTextA 함수로 대치되고
    - 유니코드의 경우에는 SetWindowsTextW 함수로 대치됨
    - 두 함수의 인자 (문자열 관련) 타입이 다름

문자열 입력 함수'

- 함수의 작성
  - MBCS의 경우

```
char* message = "Hello";
MessageBox(NULL, message, "Hi", MB_OK);
```

  - 유니코드의 경우

```
WCHAR* message = L"Hello";
MessageBox(NULL, message, L"Hi", MB_OK);
```

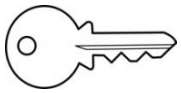
- TCHAR를 사용한 작성
  - 상수 문자열에 대해서 "_T"를 사용
    - eg., _T("John")은 MBCS에서는 "John"으로, 유니코드에서는 L"John"으로 해석함

```
TCHAR* message = _T("Hello");
MessageBox(NULL, message, _T("Hi"), MB_OK);
```

    - _T 매크로와 동일한 것들 : TEXT, _TEXT, __TEXT, __T

Win32 상수와 타입들

- Win32 편의 타입들
  - 타입과 그 타입의 포인터 타입을 함께 정의
    - 타입명 앞에 'P' 또는 'LP'를 붙임
- 자주 쓰이는 타입들
  - 부울: BOOL
    - 값=TRUE 또는 FALSE (내부구현: BOOL==int, TRUE==1, FALSE==0)
  - 바이트 데이터: BYTE/WORD/DWORD
    - unsigned char/short/long (8/16/32-bit)
  - 정수 표현: CHAR/SHORT/INT/LONG
    - char/short/int/long (8/16/32/32-bit)
  - Unsigned 정수 표현: UCHAR/USHORT/UINT/ULONG
  - 실수 표현: FLOAT/DOUBLE
    - float/double (32/64-bit)
- 참고: MSDN “Windows Data Types”



COM 프로그래밍

- COM(Component Object Model)이란?
  - 컴포넌트 기반 응용을 만들 수 있도록 MS사에서 개발한 소프트웨어구조
- COM 오브젝트
  - 인터페이스(interface)라고 부름
    - C++에서의 클래스와 비슷하게 이용됨
    - 모든 COM 인터페이스는 IUnknown COM 인터페이스를 상속받아 구현됨
  - 언어에 완전히 독립적이고 프로세스 간 통신 기능을 기본으로 제공
- DirectX
  - 다양한 COM 인터페이스를 제공
  - 이름은 모두 접두어 'I'를 붙여서 명명되었음

COM 객체의 생성과 소멸

- COM 객체의 생성
  - COM 객체는 자신의 메모리 관리를 스스로 수행함
  - 따라서 C++의 new 키워드로 생성 불가함
  - 객체의 참조(포인터)를 요청하면 됨
    - 다른 COM 인터페이스의 함수나 특별한 함수를 호출
- COM 객체의 소멸
  - C++의 delete가 아님
  - 인터페이스의 이용이 종료되면 이를 알리면 됨
    - 인터페이스의 Release 함수를 호출
      - 모든 인터페이스에는 Release 함수가 구현되어 있음

COM 사용에서 리턴값의 확인

- COM 객체
  - 응용프로그램을 지원하는 블랙박스 형태의 컴포넌트
  - 따라서, 올바로 사용하려면: COM의 정확한 명세를 알아야 함
    - API 설명서 참조
  - COM 함수의 실행 결과
    - 기대한 실행을 하지 않을 수도 있음
    - 반드시 실행 결과를 살펴서 실행 도중에 에러가 있었는지를 확인해야 함
- HRESULT를 사용한 실행 결과의 확인
  - HRESULT는 LONG과 동일한 32비트 정수임
  - 많은 Win32 API 함수나 COM의 함수들은 HRESULT 타입의 값을 리턴함
    - 표준 리턴 코드로 정의된 실행 결과를 리턴함
  - 예외: 표준 리턴 코드를 사용하지 않는 경우도 흔함
    - 어떤 함수: 다양한 에러 코드를 가지는 함수들은 자신의 에러 코드를 직접 정의하고 이를 리턴함
    - 어떤 함수: 에러가 없지만 리턴되는 32비트의 일부 비트들을 다른 용도로 활용
    - 따라서, HRESULT를 리턴하는 함수에 대한 에러 확인은 그 함수의 API를 보고 확인해야함

COM 사용에서 리턴값의 확인'

- HRESULT 타입
  - 에러가 없으면 'S_'로 시작하는 상수값을 리턴
    - 일반적으로, 성공이면 S_OK를 리턴
  - 에러가 있으면 'E_'로 시작하는 상수값을 리턴
    - 일반적으로, 실패이면 E_FAIL을 리턴

```
HRESULT hr = SomeFunction();
if (hr == E_FAIL) { /* Error */ }
```

- 매크로 사용
  - 에러인 경우이지만 E_FAIL 외에 다른 값을 리턴할 수도 있음
    - 위의 코드는 문제가 생김
  - Win32에서 FAILED와 SUCCEDDED라는 매크로를 지원함
    - HRESULT 값을 입력으로 하고 BOOL 값을 출력으로 함
    - FAILED/SUCCEEDED는 에러/성공에 대한 코드들에 대해서 TRUE를 리턴함

```
if (FAILED(SomeFunction())) { /* Error */ }
```

에러 코드의 확인

- DX 함수의 에러 발생 여부
  - HRESULT 리턴값
  - 에러가 발생되는 경우에 리턴된 에러 코드가 어떤 의미?
- 디버그 모드에서의 에러 코드 확인 방법
  - 헤더파일 DxErr.h
  - 라이브러리 링크 DxErr.lib
  - 디버그 모드로 실행해야 함
    - Visual Studio의 출력창에 메시지가 출력됨
  - 에러 코드의 확인을 위한 소스 코드 :

```
if (FAILED(hr=DXSomeFunction(invalidParam))) {
        char buf[2048];
        sprintf(buf, "My Error: %s: %s₩n",
        DXGetErrorString(hr), DXGetErrorDescription(hr));
        OutputDebugString(buf); //출력을 Visual Studio의 디버그창으로 보냄
}
```

  - 예시: 인자값의 오류로 인해서 에러가 발생한 경우

```
My Error: E_INVALIDARG: An invalid parameter was passed to the returning function
```

에러 코드의 확인’

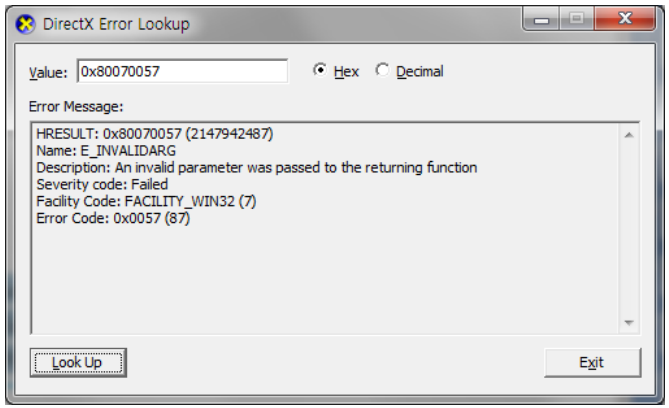
- 디버그 모드에서의 에러 코드 확인 방법’
  - 매크로를 사용한 더욱 간결한 표현

```
if (FAILED(hr=DXSomeFunction(invalidParam))) {
        DXTRACE_ERR( TEXT("My Error: "), hr );
}
```

  - 예시: 동일한 에러에 대해서 아래의 메시지가 출력창에 출력됨

```
mysource.cpp(97): My Error: hr=E_INVALIDARG (0x80070057)
```

- 참고: 프로그램 메뉴에
  - Microsoft DirectX SDK » DirectX Utilities » DirectX Error Lookup
    - Value에 16진수나 10진수의 에러 코드를 입력하면 이 에러 코드에 대한 자세한 정보들을 보여줌



에러 처리 편리 함수

- 선택사항: 편리 함수 만들기

```
#define CheckHr(hr) CheckForDxError(__FILE__,__LINE__,hr)
```

  - Visual Studio가 파일명과 라인번호를 제공함
  - 출력된 에러메시지를 더블클릭하면, 해당 위치로 바로 이동함

```
void CheckForDxError(const char *file, int line, HRESULT hr)
{
  if (!FAILED(hr)) return;

  // DX 에러명과 에러 명세를 얻음
  char desc[1024];
  sprintf(desc,"(DX) %s - %s", DXGetErrorString(hr), DXGetErrorDescription(hr));

  // 위의 메시지와 더불어 파일명과 라인번호를 출력함
  char buf[2048];
  sprintf(buf, "%s(%d) : Error: %s₩n", file, line, desc);
  OutputDebugString(buf);

  // 디버거가 에러 발생 위치로 break되도록 함. 개발자가 편리하게 문제를 고칠 수 있음.
  DebugBreak();
}
```

용어의 사용

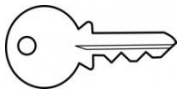
- 정확한 용어의 이해와 사용
  - 애매한 경우가 많음
  - 용어를 정확히 구분해야 내용을 정확히 이해할 수 있음
  - 특히 한글 용어
    - 약속하자
- 인자와 인자값
  - 인자(parameter)
    - 함수의 정의에서 함수 헤더에 명시된 변수를 의미
    - 인자 = 매개변수, 형식인자, 형식인수
  - 인자값(argument)
    - 함수의 호출 시에 인자로 할당될 값, 변수, 수식 등을 의미
    - 인자값 =인수, 실행인자, 실인자, 실인수

```
void firstfunc(int a, int b) {}
void secondfunc(int c) { firstfunc(c, c+2); }
```

  - 참고: 종종 parameter/argument 를 매개변수/인자로 번역

용어의 사용'

- 함수 관련
  - 함수(function)
    - C/C++ 언어로 정의한 함수(function)
    - C++ 클래스, 인터페이스, 구조체 내에서 정의된 멤버함수
      - 객체지향적 개념에서는 멤버함수를 메서드(method)로도 언급함
- 구조체 관련
  - 요소(element)
    - 벡터/행렬 구조체의 내부 변수들
  - 필드(field)
    - 벡터/행렬 행렬 구조체 외의 모든 구조체(struct) 내부에서 정의된 변수들
    - 다른 용어: 레코드, 요소, 멤버, 변수 등
- 호출 및 자원 관련
  - 리턴(return)
    - 호출된 함수가 호출자에게 계산된 값을 전달하는 것
      - return 키워드로 값을 전달하는 경우와 포인터 인자로 값을 전달하는 경우를 모두 포함
  - 반납(release)
    - 다 사용한 인터페이스나 시스템 자원을 반납하는 행위
    - 소멸(destroy)을 포함하는 보다 포괄적인 의미
  - 소멸(destroy)
    - 생성(create) 절차가 수행되기 전의 상태로 되돌리는 행위
    - 클래스의 소멸자(destructor), delete 키워드, free 함수 등의 호출행위에 해당



Windows GDI 그리기

- 그리기 함수 호출
  - WM_PAINT 메시지 발생 시마다 호출

```
switch (message) {
        case WM_PAINT:
                OnPaint(hWnd);
                break;
        // 다른 메시지들을 다루는 코드…
}
```

- Windows GDI 함수
  - 사각형을 그리는 함수의 작성 예

```
LRESULT OnPaint(HWND hWnd)
{
        PAINTSTRUCT ps;

        BeginPaint(hWnd, &ps);

        RECT rc; GetClientRect(hWnd, &rc);

        FillRect(ps.hdc, &rc, GetSysColorBrush(COLOR_HIGHLIGHT));

        HPEN hMyRedPen = CreatePen(PS_SOLID, 3, RGB(255,0,0));
        HGDIOBJ hOldPen = SelectObject(ps.hdc, hMyRedPen);

        Rectangle(ps.hdc, rc.left+100, rc.top+100, rc.right-100, rc.bottom-100);

        SelectObject(ps.hdc, hOldPen);
        DeleteObject(hMyRedPen);
        EndPaint(hWnd, &ps);
        return 0;
}
```

Windows GDI 그리기'

05.GdiSimpleRectangle

- GDI 프로그래밍 리소스
  - The GDI in Windows API : http://zetcode.com/gui/winapi/gdi/
  - The Graphical Device Interface (Lesson10~17) : http://www.functionx.com/win32/Lesson10.htm
  - 윈도우프로그래밍실습(이종옥) - 6장 그래픽 : http://contents.kocw.or.kr/KOCW/document/2015/korea_sejong/leejonguk/08.pdf
  - 등등

- 실습: Windows GDI 함수들을 사용한 Win32 응용프로그램 작성
  - 윈도우 클라이언트 영역을 하이라이트 컬러로 채우고 그 내부에 붉은색 사각형을 그리는 GDI 프로그램을 작성하자