

그래픽스 프로그래밍 : 2D 프로그래밍 in OpenGL

4주차, 2023

Prof. Kim, Soo Kyun

학습 목표

- OpenGL
- 윈도우 프로그래밍
- 윈도우와 OpenGL 연결하기
- Simple program using OpenGL

OpenGL의 특징

- Silicon Graphics(SGI)사의 워크스테이션용 그래픽스 라이브러리 IrisGL에서 시작
- OpenGL Architecture Review Board (<http://www.opengl.org>)
 - OpenGL 공식기구 : SGI, DEC, IBM, Apple, Microsoft, 등의 컨소시엄
 - 다양한 플랫폼에서 작동되도록 GL을 수정하여 OpenGL 발표
 - 개방형 표준화 산업 컨소시엄인 크로노스 그룹(Khronos Group)이 인수 후 2004년 OpenGL 2.0, 2008년 OpenGL 3.0, 2014년 OpenGL 4.5 표준 발표
- 현재 2D와 3D 그래픽스 API로 가장 널리 사용되는 산업계 표준으로 성장
- 임베디드 시스템을 위한 OpenGL ES, 웹환경을 위한 WebGL, 안전이 중요한 장치를 위한 OpenGL SC 등으로 확대

OpenGL의 특징

- **“그래픽스 하드웨어에 대한 소프트웨어 인터페이스”**
 - 하드웨어에 독립적, 상위 수준(High-Level)의 그래픽스 API이므로 객체 단위 프로그래밍 가능
- **플랫폼에 독립적**
 - PC, 워크스테이션 및 Mac OS, Window, Unix등 다양한 운영 체제 지원
 - WebGL은 대부분의 웹 브라우저 지원
- **다양한 그래픽스 기능을 지원하여 응용 소프트웨어 개발 용이**
 - 모델링, 변환, 색상, 명암, 그림자, 텍스처 매핑, 블렌딩 등의 고급 그래픽스 처리 기능을 제공
 - 가속 하드웨어를 사용해 처리하는 경우 많은 양의 그래픽 데이터 실시간 처리 가능
- **OpenGL 가속 하드웨어는 셰이딩 언어(GLSL)를 하드웨어적으로 가속하는 기능 제공**

OpenGL의 장점

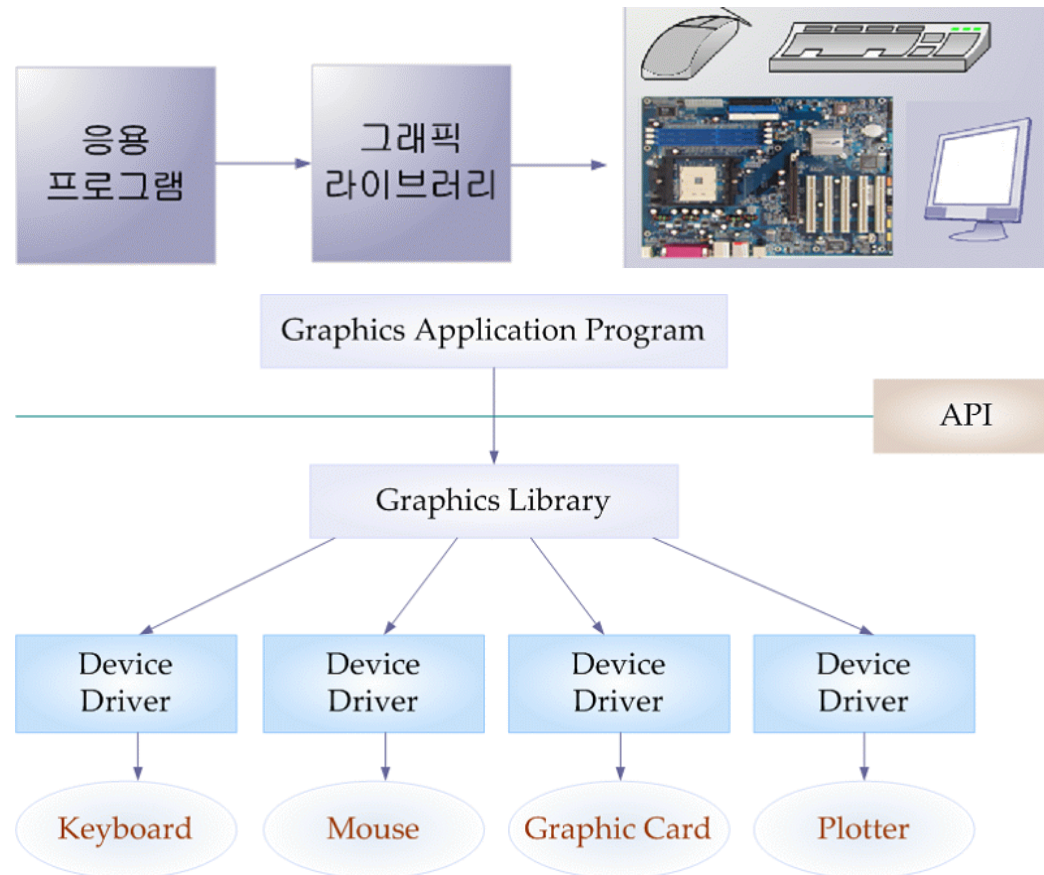
- **안정성(Stability)**
 - 지난 10년 동안 다양한 플랫폼에서 지원되어 그 사양이 충분히 검증되면서 발전
- **신뢰성 및 이식성(Reliability & Portability)**
 - OpenGL 응용프로그램은 운영체제나 윈도우 시스템에 상관없이 동일한 출력결과를 생성
- **확장성(Scalability)**
 - 가전기기로부터 PC, 슈퍼 컴퓨터에 이르기까지 다양한 종류의 시스템에서 동일하게 작동
- **편리성(Ease of Use)**
 - 직관적인 인터페이스와 논리적인 명령어들로 구성
- **문서화(Well-documented)**
 - 문서화 작업이 잘 이루어져 있으며 많은 책들이 출판

다른 버전들 (Other Versions)

- **OpenGL ES**
 - 임베디드 시스템
 - 버전 1.0: 간소화된 OpenGL 2.1
 - 버전 2.0: 간소화된 OpenGL 3.1
 - 셰이더 기반
- **WebGL**
 - ES 2.0의 자바스크립트 구현
 - 최신 브라우저에서 지원
- **OpenGL 4.6 (July 2017)**
 - 지오메트리 셰이더 및 테셀레이터 추가

그래픽 API

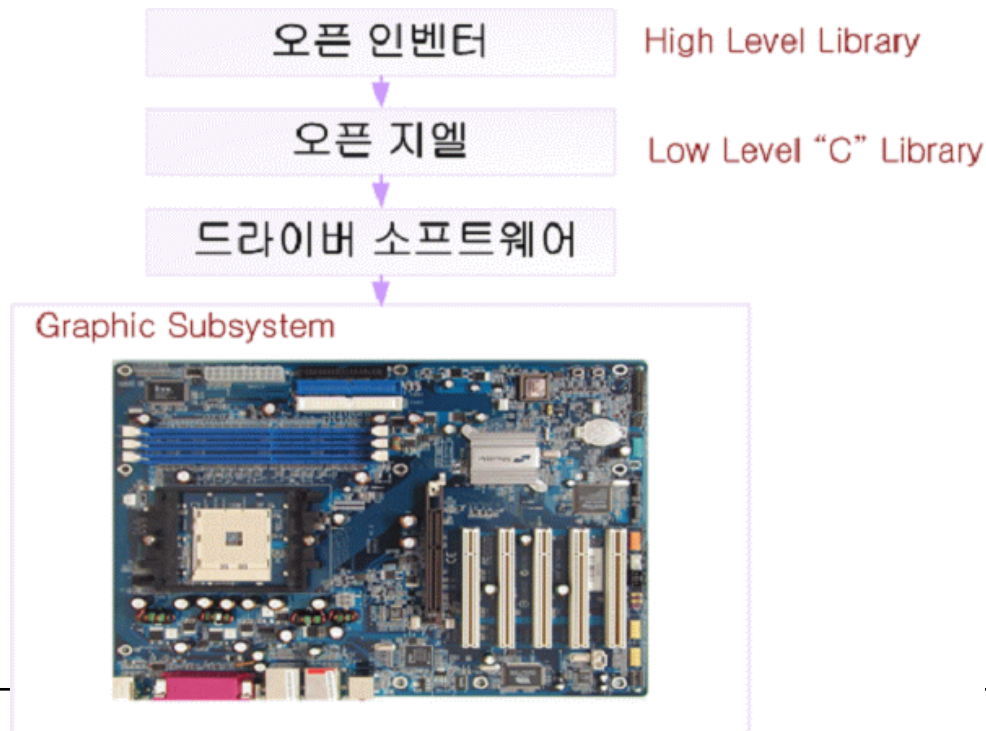
- 응용프로그램 인터페이스
 - 라이브러리



OpenGL

- 저수준 API

- 장면을 묘사하는 것이 아니라 구체적 프러시저를 호출
- cf. DirectX from Microsoft: 호환성 결여
- 하드웨어와 거의 직접 연관 (하드웨어 성능을 최대한 발휘)
- Inventor, VRML, Java3D 등 고수준 API의 기반
- 드라이버 소프트웨어에 비해서는 상대적으로 고수준 함수



OpenGL 설계원리

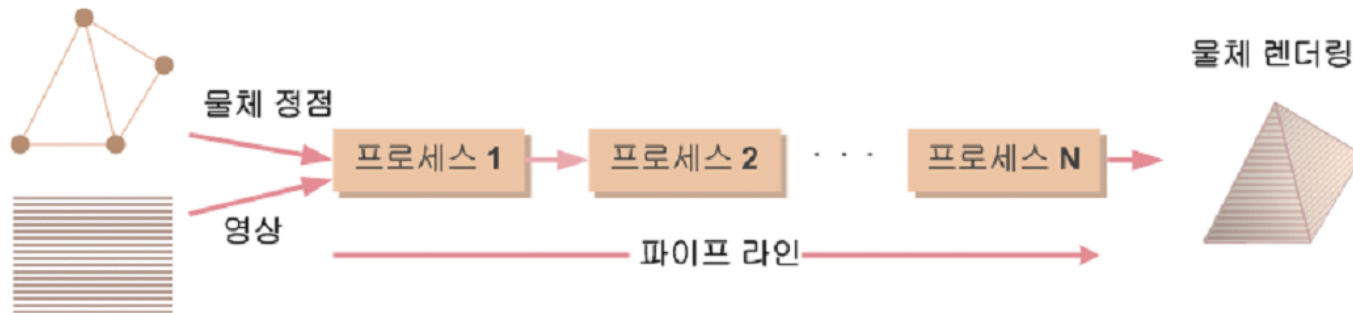
- **범용성(Generality)**
 - 워크스테이션, 슈퍼 컴퓨터, 개인용 컴퓨터 -> 운영체제에 무관
- **효율성(Performance)**
 - 그래픽 하드웨어의 가속 기능을 최대한 발휘.
 - 회사마다 서로 다른 기능-> 공통적인 부분을 찾아내어 그 성능을 극대화
 - 공통부분이 아닌 것에 대해서는 활성화 또는 비활성화 등 기능 모드를 제공
- **독립성(Orthogonality)**
 - 기능 간의 독립성을 최대한 보장
 - 기능끼리 서로 얽혀 발생하는 오류를 방지
- **완전성(Completeness)**
 - 특정 하드웨어 기능에 대해서는 ARB 확장 형태로 명령어를 제공
 - 다수의 하드웨어가 확장 기능을 지원하면 표준기능으로 변경
 - 소프트웨어적으로라도 실행할 수 있도록 배려
- **상호 작업성(Interoperability)**
 - 그래픽 명령은 A 컴퓨터에서 내리되 실행은 B 컴퓨터에서
 - 클라이언트-서버 모델(Client-Server Model)지원
 - 성능이 낮은 클라이언트 컴퓨터가 고성능 서버를 이용



파이프라인

- GPU 설계원리

- CPU 파이프라인과 유사
- 분업에 의한 동시처리로 처리속도를 극대화
 - Ex. 컨베이어 시스템
- 파이프라인 서브 프로세스는 모두 하드웨어화

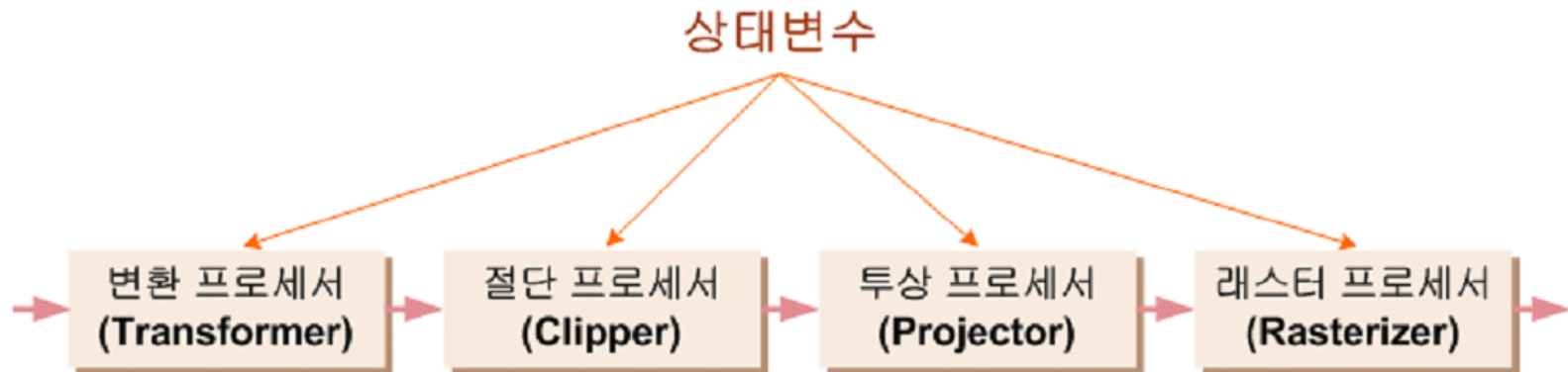


OpenGL 상태(State)

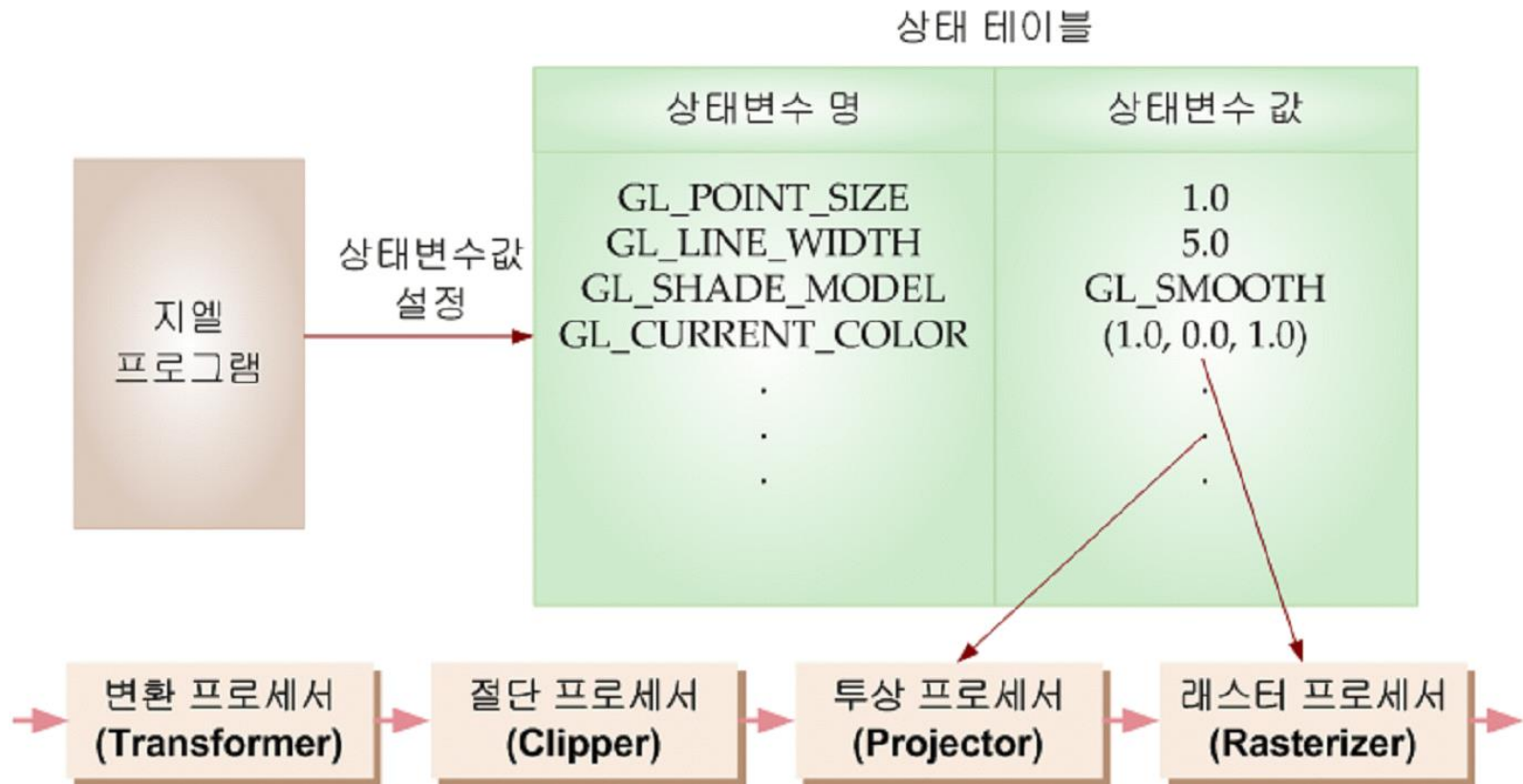
- OpenGL은 상태 머신
- OpenGL 함수는 2가지 유형이 있음
 - Primitive 생성
 - 프리미티브가 표시되면 출력을 발생시킬 수 있음
 - 정점이 처리되는 방법과 프리미티브의 모양은 상태에 의해 제어됨
 - 상태 변경 (State changing)
 - 변환 함수 (Transformation functions)
 - 어트리뷰트 함수 (Attribute functions)
 - 3.1에서 대부분의 상태 변수는 애플리케이션에 의해 정의되어 셰이더로 전송됨

OpenGL 상태(State)

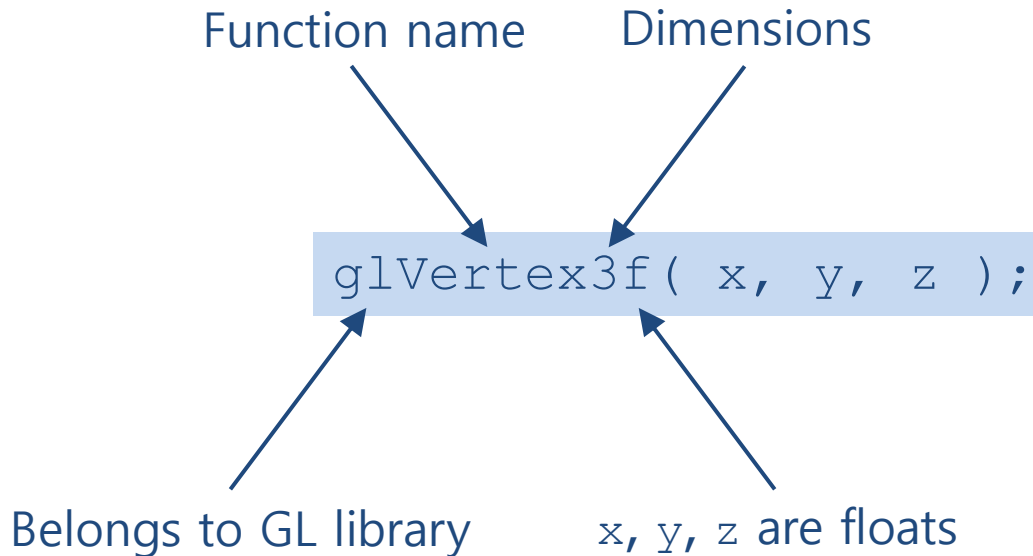
- 지엘의 역할 = 상태변수 설정
- 파이프라인은 상태변수를 참조해서 자동으로 실행됨



지엘 프로그램, 상태변수, 파이프라인



OpenGL 함수 형식



접미사	데이터 타입	C/C++ 타입명	지엘 타입명
f	32-bit floating point	float	GLfloat
d	64-bit floating point	double	GLdouble
b	8-bit integer	signed char	GLbyte
ub	8-bit unsigned integer	unsigned char	GLubyte, GLboolean
i	32-bit integer	int or long	GLint
ui	32-bit unsigned integer	unsigned long	GLuint, GLenum, GLbitfield
s	16-bit integer	short	GLshort

OpenGL 함수 형식

- 벡터 타입



- 지엘은 API

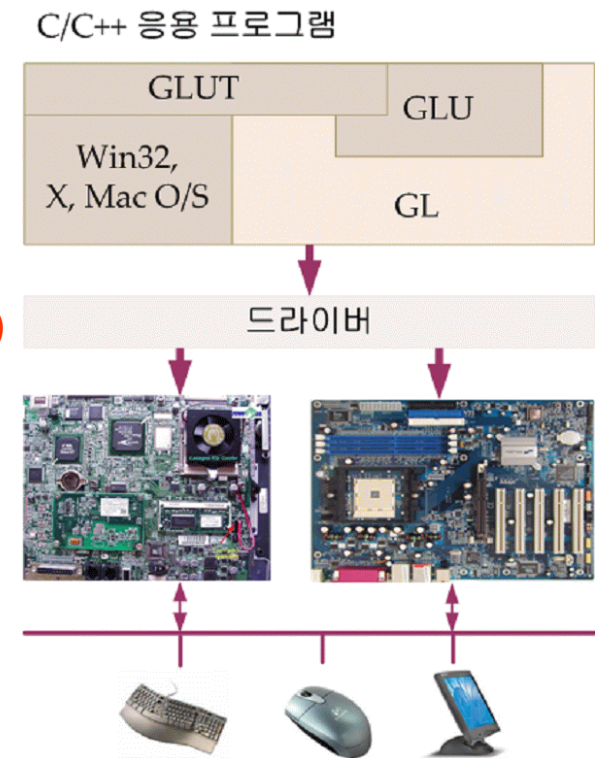
- 명령어가 아니라 함수명이며, 그러나 혼용함

- 지엘은 비 객체지향적

- 처리속도를 향상
- 함수 오버로딩이 불가능
- 3차원 정점이라면 `glVertex3f()`, 2차원 정점이라면 `glVertex2f()`

OpenGL 라이브러리(Libraries)

- **지엘 라이브러리(GL: OpenGL Core Library)**
 - 렌더링 기능을 제공하는 함수 라이브러리
- **지엘 유틸리티 라이브러리(GLU: OpenGL Utility Library)**
 - 50여개의 함수. GL 라이브러리의 도우미
 - 다각형 분할, 투상, 2차원 곡면, 너브스등 고급기능을 제공하는 함수
 - GL 함수로 작성
- **지엘 유틸리티 툴 킷(GLUT: OpenGL Utility Toolkit)**
 - 사용자 입력을 받아들이거나 화면 윈도우를 제어하기 위한 함수
 - 윈도우 운영체제 기능과의 인터페이스



Windows 프로그래밍의 기본

Windows 프로그래밍의 기본

- 윈도우는 무엇인가?
- 윈도우 프로그래밍을 하는 이유는 무엇인가?
- 윈도우가 만들어 지고, 화면에 색이 칠해지는 과정은 생각해 보았는가?
 - **GDI (Graphic Device Interface)**에 대해 들어보았는가?
- OpenGL과 윈도우는 어떻게 연결되는가?

Windows 프로그래밍의 기본

- 그래픽 디바이스 인터페이스[GDI, Graphic Device Interface]
 - 윈도우 3.1/ 95/98/NT에서 도형을 처리하는 프로그램
 - 선 그리기, 컬러 관리와 같은 그래픽 함수들을 구현하는 일을 담당하는 윈도우의 구성 요소
 - 모든 응용 프로그램은, 화면 표시 또는 인쇄 시 그래픽 디바이스 인터페이스(**GDI**)에 처리를 의뢰하면 **GDI**는 그 명령을 디스플레이 드라이버나 인쇄기 드라이버가 처리할 수 있는 형태로 변환하여 각각의 드라이버에 보냄으로써 화면 표시 또는 인쇄가 실행됨

Windows 프로그래밍의 기본

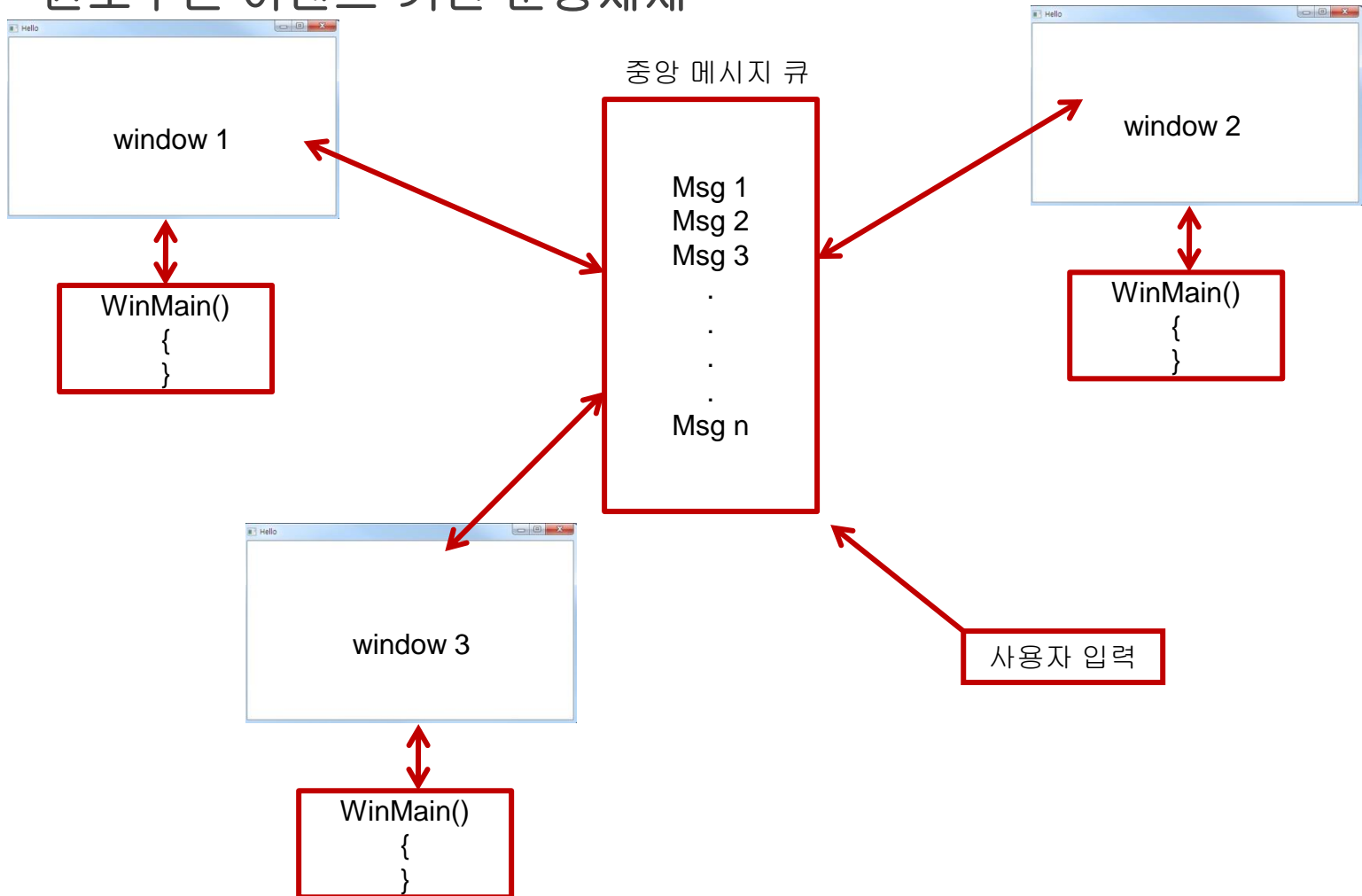
- 윈도우 시스템의 **WIN32**라이브러리
 - 2차원 그래픽스를 위한 함수들은 제공
 - 그러나 3차원 그래픽스를 위한 라이브러리는 없음
- 윈도우 시스템에서의 3차원 그래픽스 라이브러리 개발에 대한 필요성 대두
 - OpenGL
 - Dirext3D
- **OpenGL** 프로그래밍 관점에서 볼 때 한가지 중요한 점
 - 윈도우 시스템도 하나의 래스터 그래픽스 시스템으로서의 성질을 가지고 있음
 - 프레임 버퍼가 정의 되어야 함
 - 내부적으로 윈도우의 기본 요소들을 어떻게 표현할 것인가가 설정되어야 함
 - 선분, 다각형, 폰트 등등

윈도우 시스템과 OpenGL의 연결

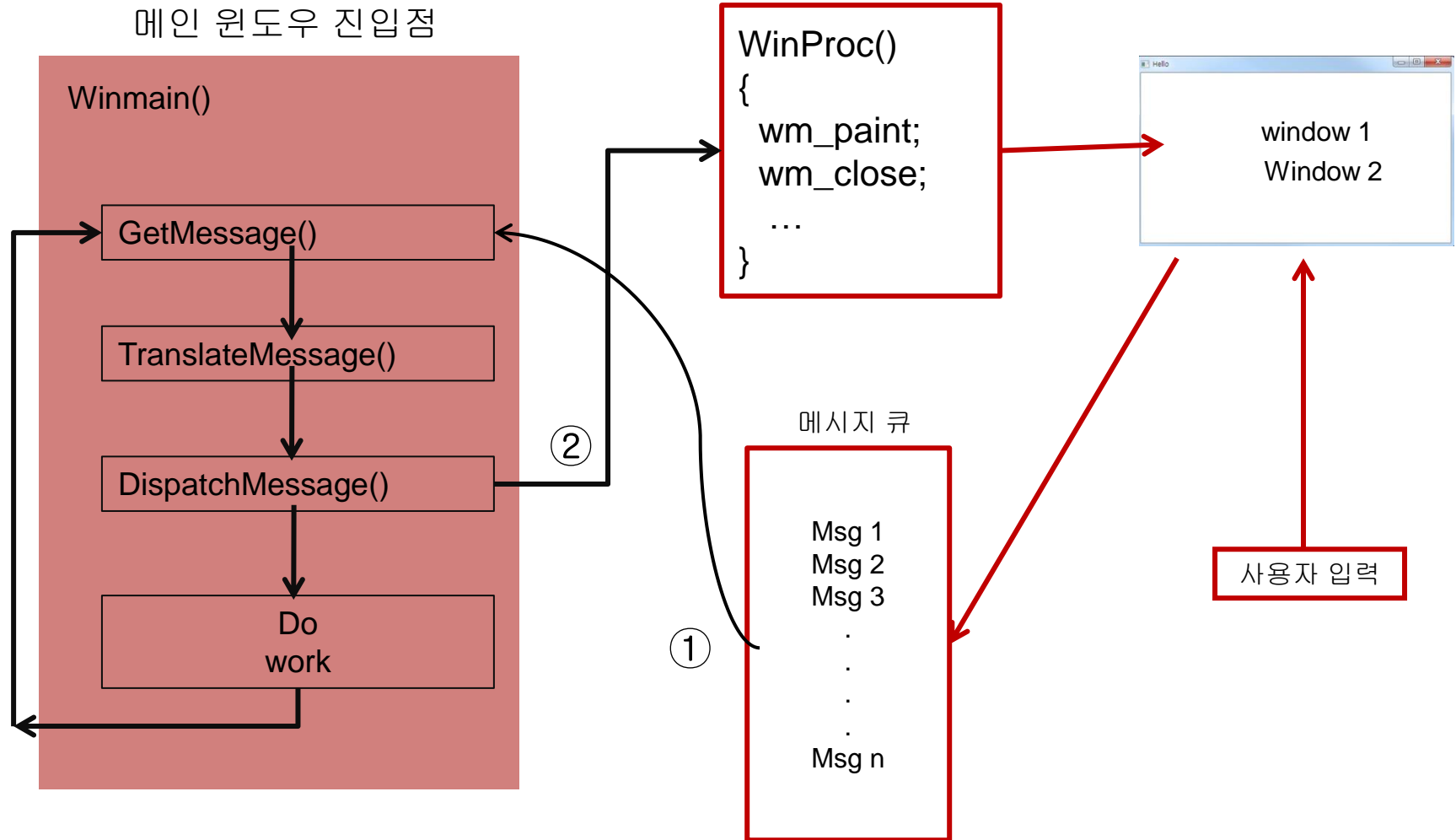
- 윈도우를 이용하여 3차원 그래픽스를 하기 위한 방법
 - 윈도우 시스템에서 제공하는 래스터 그래픽스 시스템을 기반으로 프로그래밍해야 함
 - 윈도우 프로그래밍 문맥(context)에서 하나의 추상적인 래스터 시스템인 OpenGL을 윈도우 시스템의 그래픽스 시스템에 연결
 - OpenGL에서 제공하는 3차원 라이브러리를 통해 3차원 그래픽스를 프로그래밍함
 - 실제적으로 하드웨어를 제어하는 윈도우 시스템에 의해 결과 화면 출력

Windows 프로그래밍의 기본

- 윈도우는 이벤트-기반 운영체제

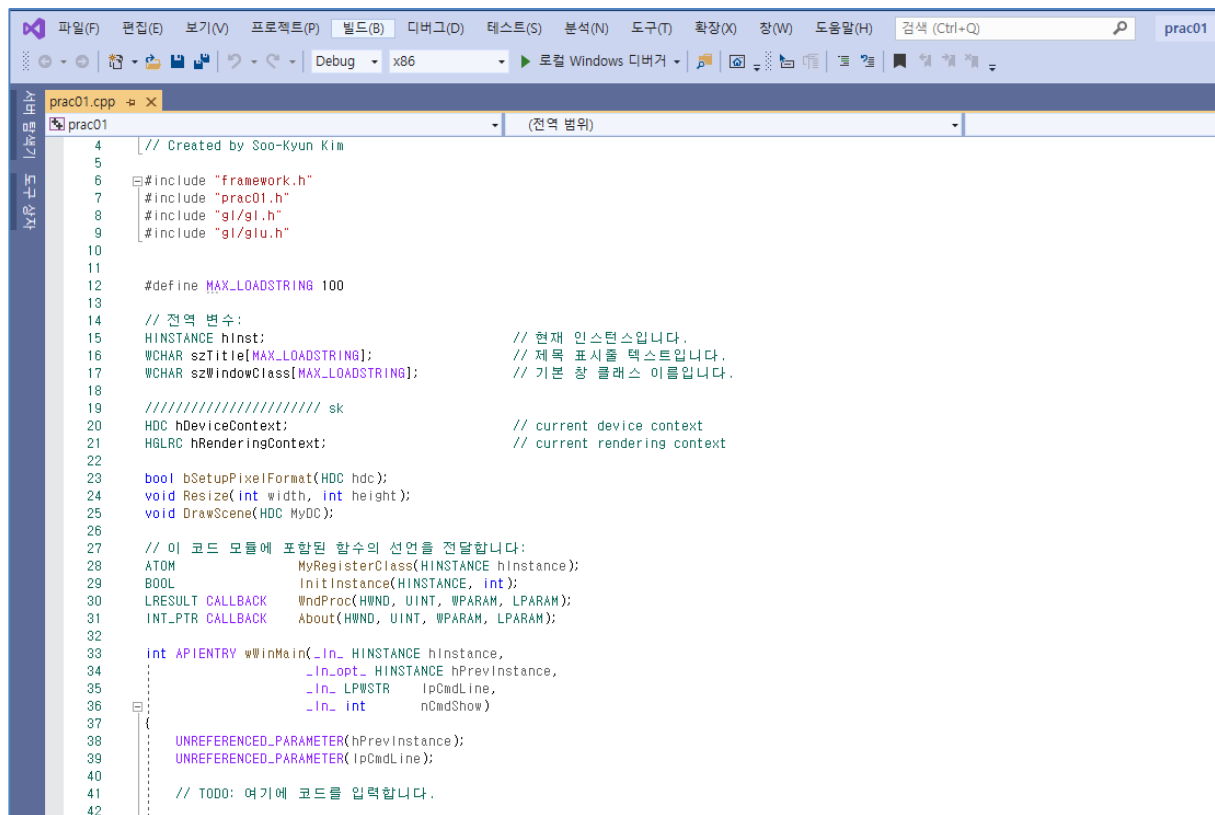


메인 이벤트 루프



윈도우와 OpenGL 연결하기

- **prac01.cpp** 를 이용한 프로젝트 만들기
 - “하영드리미 수업게시판” → “Prac01_4주차Prac_Graphics Proramming2D_학생용” 프로젝트를 다운로드 받을 것



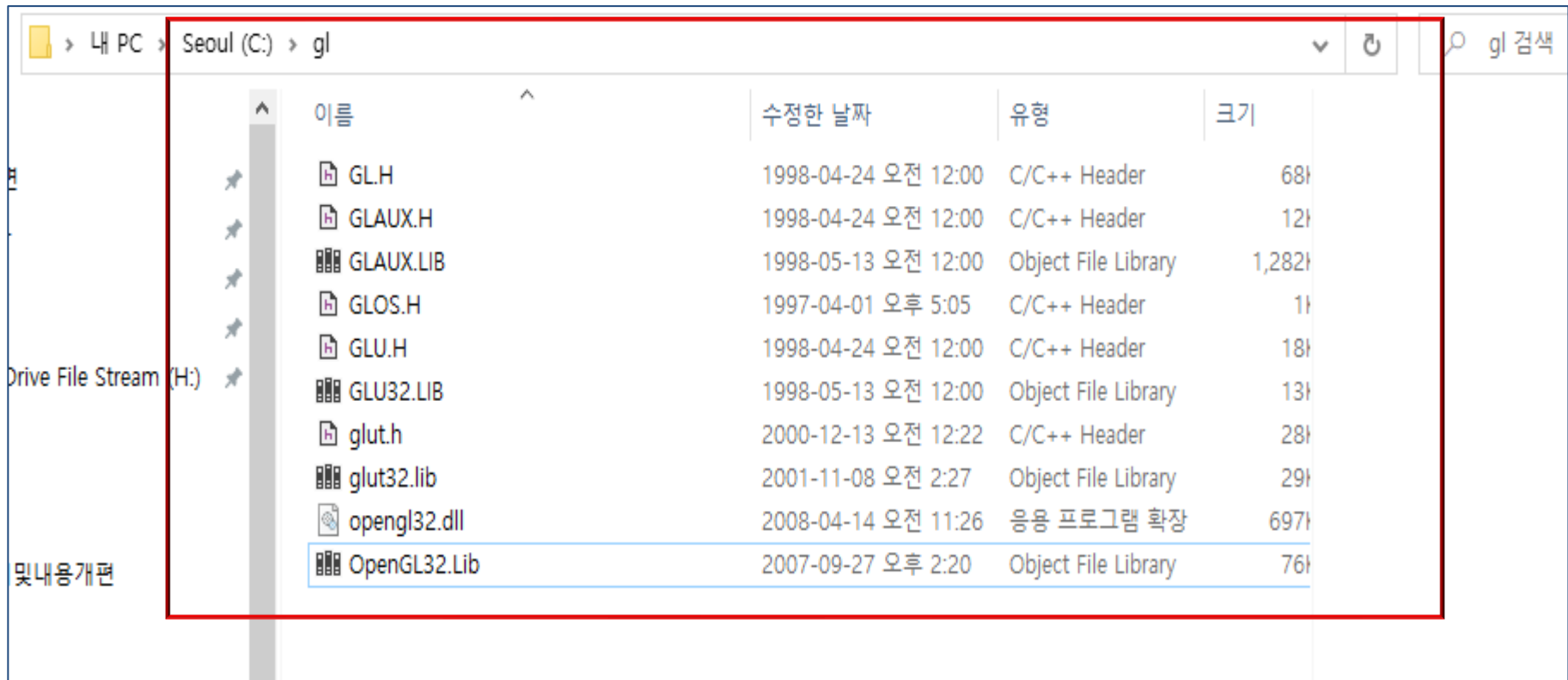
```
4 // Created by Soo-Kyun Kim
5
6 #include "framework.h"
7 #include "prac01.h"
8 #include "gl.h"
9 #include "glu.h"
10
11
12 #define MAX_LOADSTRING 100
13
14 // 전역 변수:
15 HINSTANCE hInst; // 현재 인스턴스입니다.
16 WCHAR szTitle[MAX_LOADSTRING]; // 제목 표시줄 텍스트입니다.
17 WCHAR szWindowClass[MAX_LOADSTRING]; // 기본 창 클래스 이름입니다.
18
19 ////////////////////////////////////////////////// sk
20 HDC hDeviceContext; // current device context
21 HGLRC hRenderingContext; // current rendering context
22
23 bool bSetupPixelFormat(HDC hdc);
24 void Resize(int width, int height);
25 void DrawScene(HDC MyDC);
26
27 // 이 코드 모듈에 포함된 함수의 선언을 전달합니다:
28 ATOM MyRegisterClass(HINSTANCE hInstance);
29 BOOL InitInstance(HINSTANCE, int);
30 LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
31 INT_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);
32
33 int APIENTRY wWinMain(_In_ HINSTANCE hInstance,
34                     _In_opt_ HINSTANCE hPrevInstance,
35                     _In_ LPWSTR lpCmdLine,
36                     _In_ int nCmdShow)
37 {
38     UNREFERENCED_PARAMETER(hPrevInstance);
39     UNREFERENCED_PARAMETER(lpCmdLine);
40
41     // TODO: 여기에 코드를 입력합니다.
42 }
```


OpenGL 설치 [window 11 & VS2019 기준]

파일명	폴더 위치
opengl32.dll	c:\windows\System32 C:\Windows\SysWOW64
glut32.dll	c:\windows\System32 C:\Windows\SysWOW64
glu32.dll	c:\Windows\System32 C:\Windows\SysWOW64
opengl32.lib	C:\gl (gl 폴더를 생성하고 "opengl32.lib" 파일 넣기)
glu32.lib	C:\gl (gl 폴더를 생성하고 "glu32.lib" 파일 넣기)
glut32.lib	C:\gl (gl 폴더를 생성하고 "glut32.lib" 파일 넣기)
gl.h	C:\gl (gl 폴더를 생성하고 "gl.h" 파일 넣기)
glu.h	C:\gl (gl 폴더를 생성하고 "glu.h" 파일 넣기)
glut.h	C:\gl (gl 폴더를 생성하고 "glut.h" 파일 넣기)

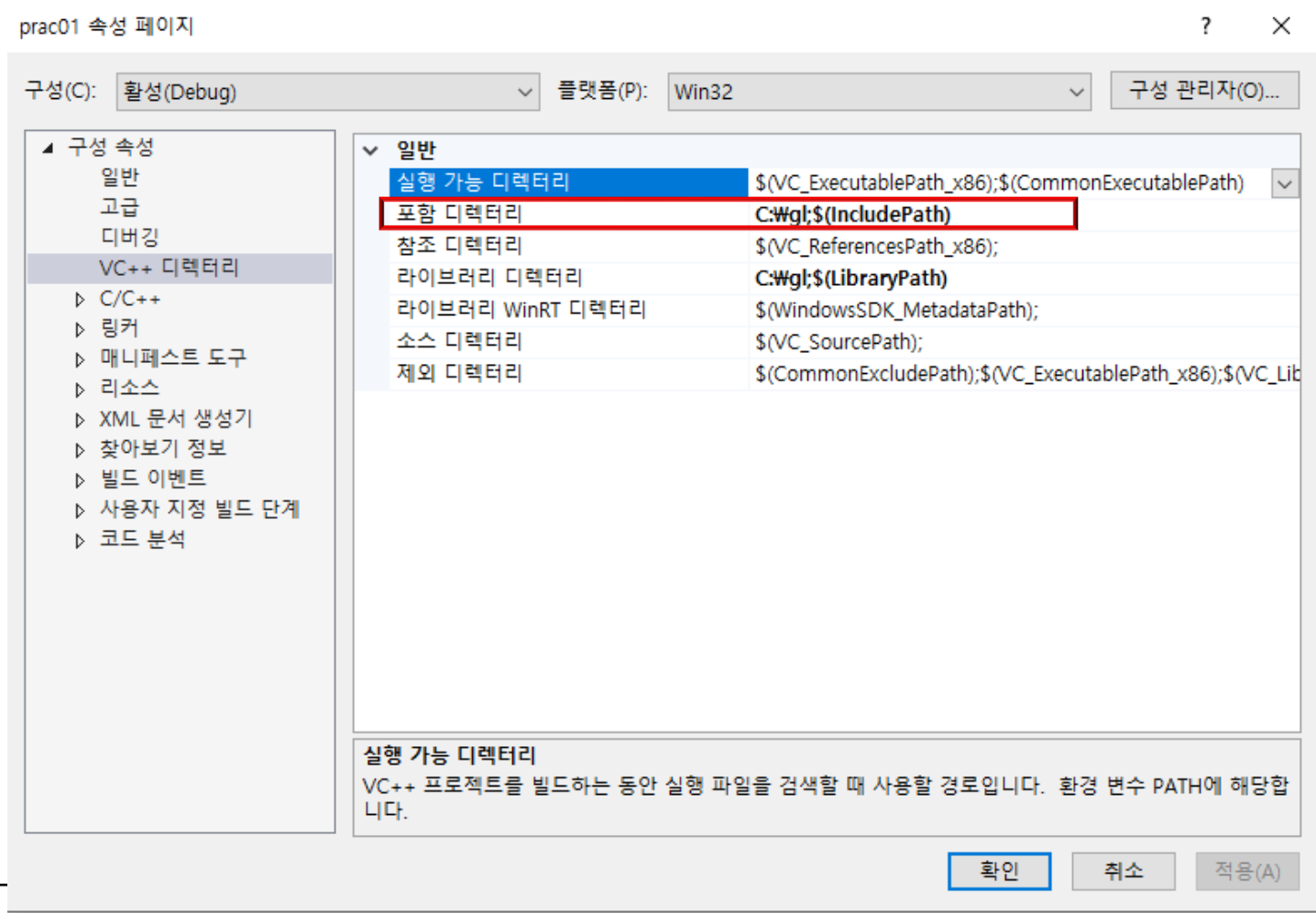
GLUT Library (참고)

- 탐색기 C 드라이브 gl 폴더를 만들고, opengl 관련 h 파일 및 lib 파일을 넣음



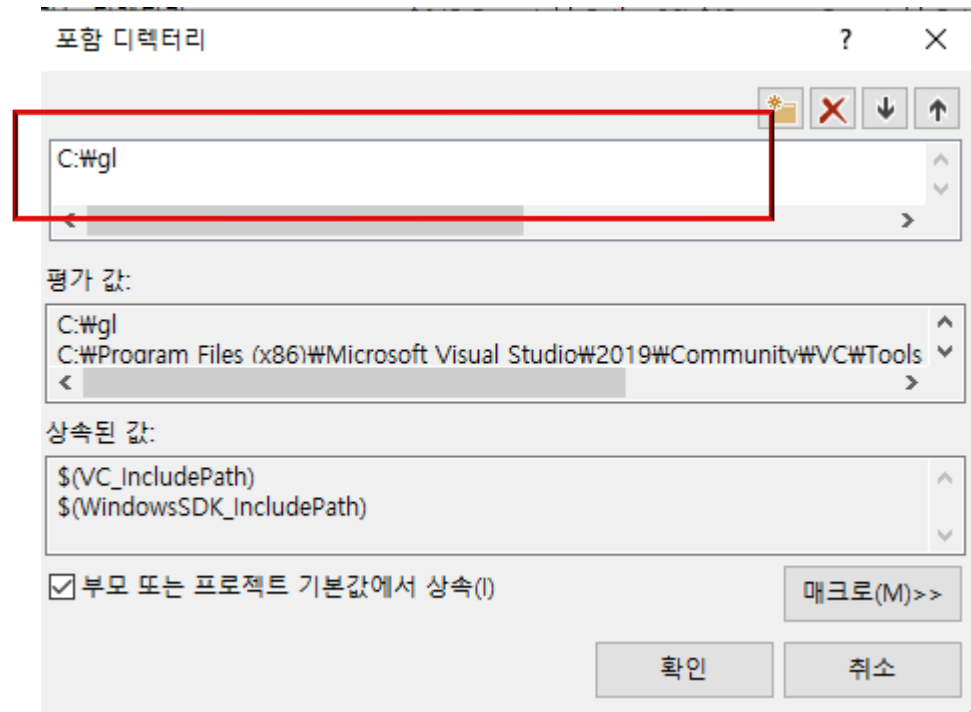
GL Library (참고)

- 포함 디렉터리 설정



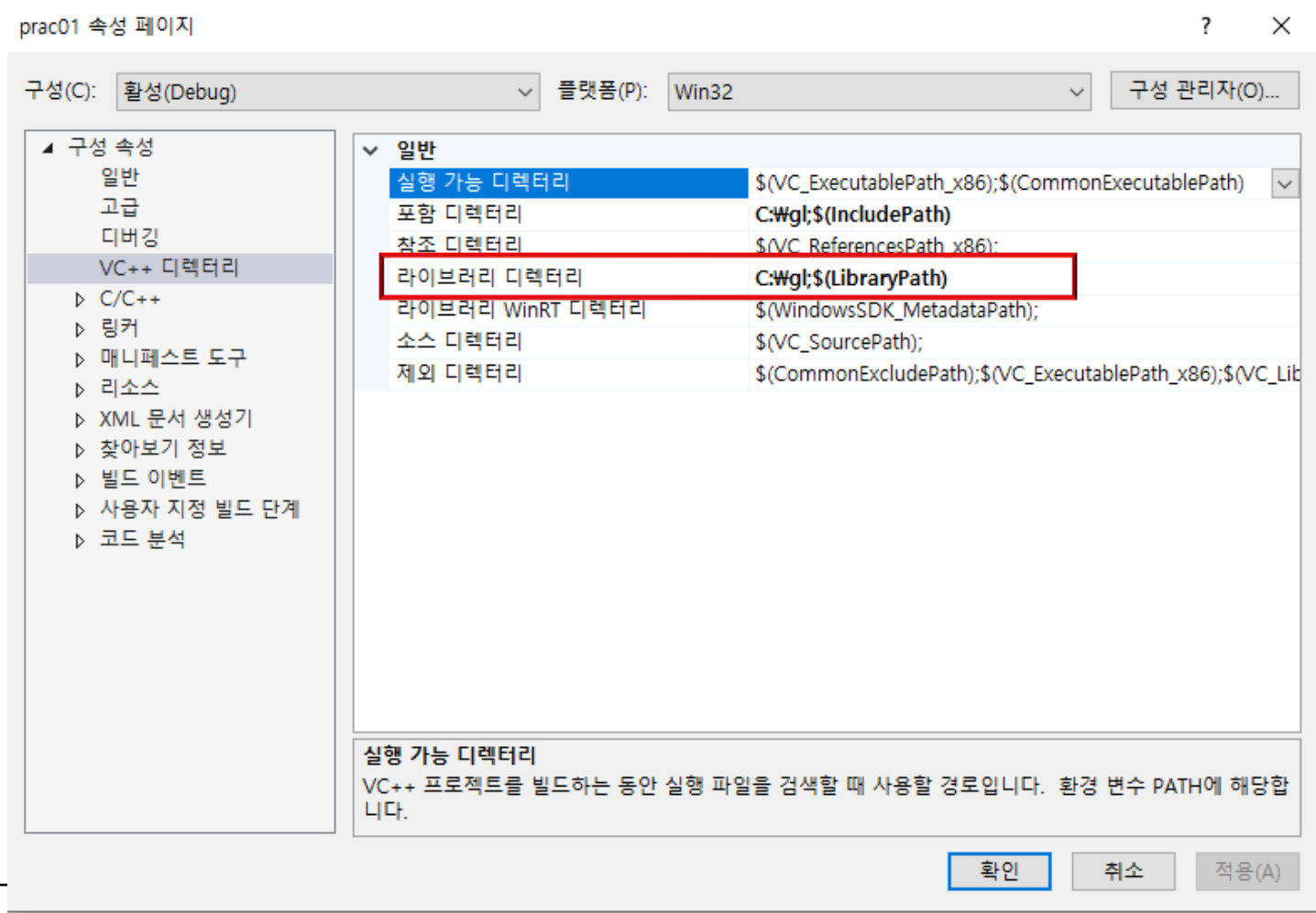
GL Library (참고)

- 포함 디렉터리에서 우리가 만든 c:\gl 폴더를 포함 시킨다.



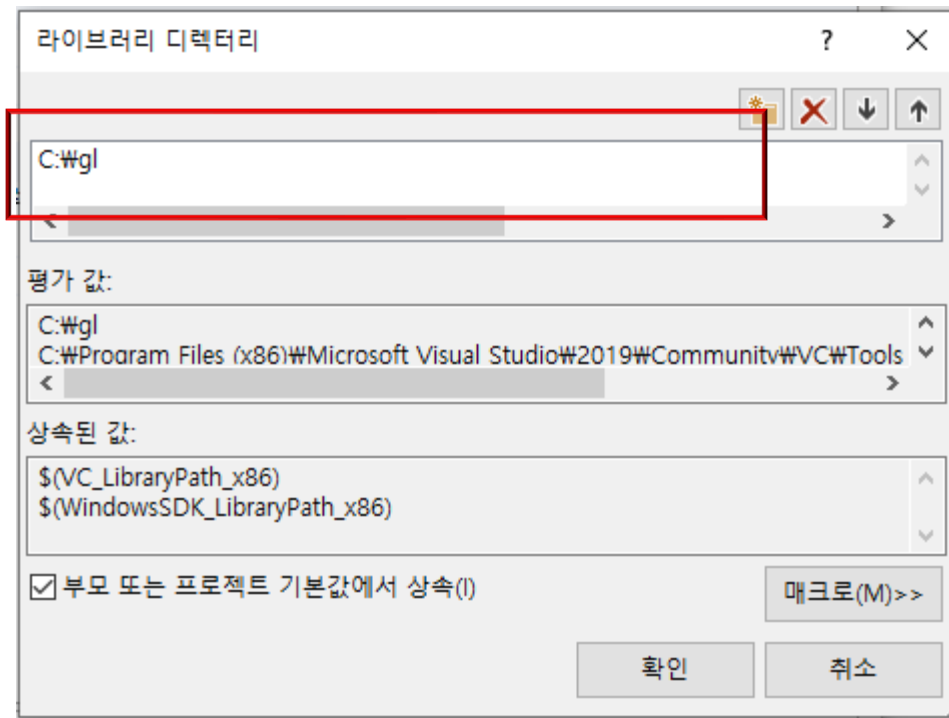
GL Library (참고)

- 라이브러리 디렉터리 설정



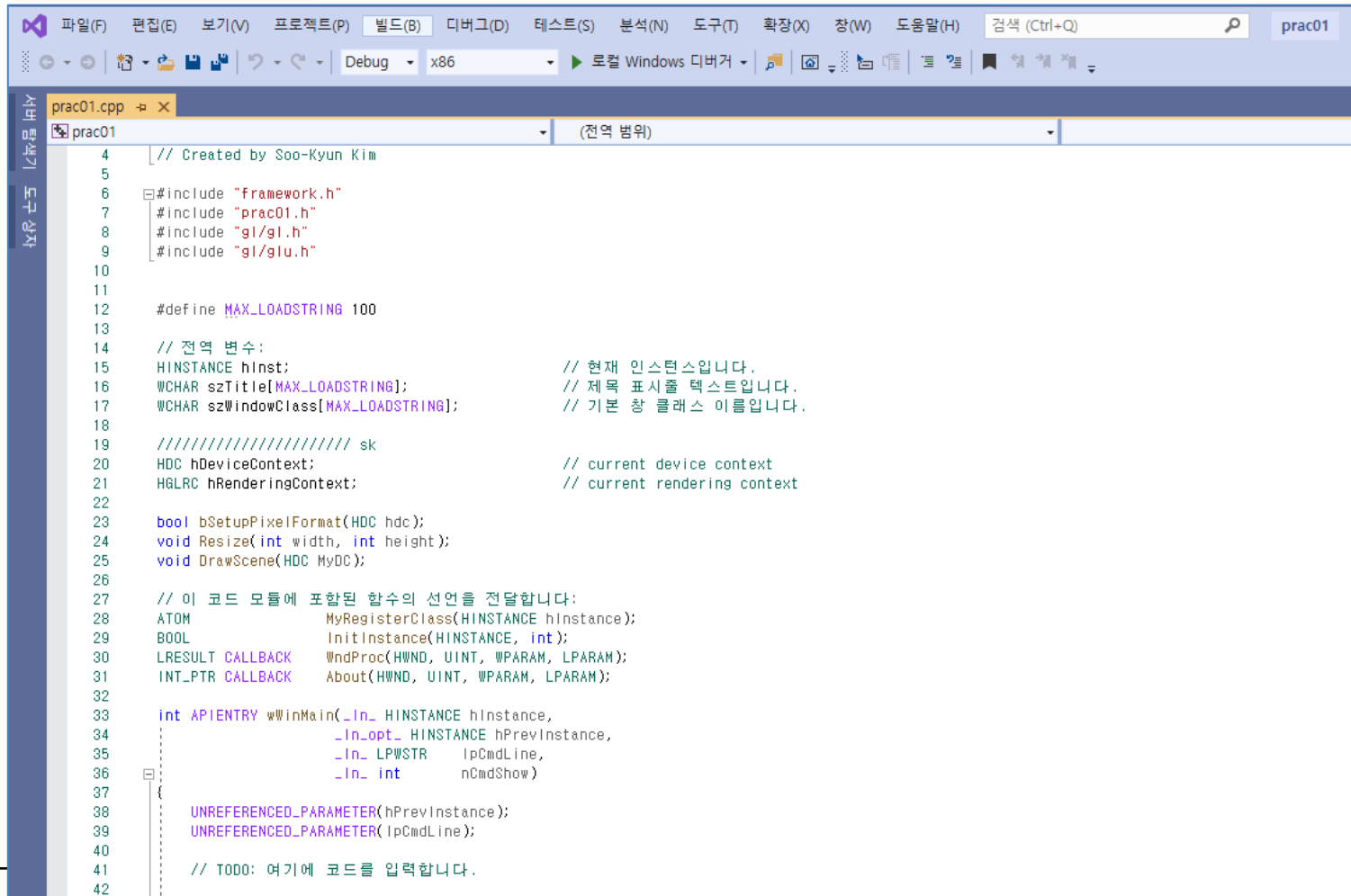
GL Library (참고)

- 라이브러리 디렉터리에서 우리가 만든 c:\gl 폴더를 포함 시킨다.



OpenGL Programming

- MS Visual Studio 2019



```
4  // Created by Soo-Kyun Kim
5
6  #include "framework.h"
7  #include "prac01.h"
8  #include "gl/gl.h"
9  #include "gl/glu.h"
10
11
12  #define MAX_LOADSTRING 100
13
14  // 전역 변수:
15  HINSTANCE hInst;                // 현재 인스턴스입니다.
16  WCHAR szTitle[MAX_LOADSTRING]; // 제목 표시줄 텍스트입니다.
17  WCHAR szWindowClass[MAX_LOADSTRING]; // 기본 창 클래스 이름입니다.
18
19  ////////////////////////////////////////////////// sk
20  HDC hDeviceContext;             // current device context
21  HGLRC hRenderingContext;       // current rendering context
22
23  bool bSetupPixelFormat(HDC hdc);
24  void Resize(int width, int height);
25  void DrawScene(HDC MyDC);
26
27  // 이 코드 모듈에 포함된 함수의 선언을 전달합니다:
28  ATOM MyRegisterClass(HINSTANCE hInstance);
29  BOOL InitInstance(HINSTANCE, int);
30  LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
31  INT_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);
32
33  int APIENTRY wWinMain(_In_ HINSTANCE hInstance,
34                      _In_opt_ HINSTANCE hPrevInstance,
35                      _In_ LPWSTR lpCmdLine,
36                      _In_ int nCmdShow)
37  {
38      UNREFERENCED_PARAMETER(hPrevInstance);
39      UNREFERENCED_PARAMETER(lpCmdLine);
40
41      // TODO: 여기에 코드를 입력합니다.
42
```

ClassView

The screenshot displays the Visual Studio IDE with the ClassView window open on the right. The main editor shows the source code of `prac01.cpp`. The ClassView window is divided into two panes: 'Solution Explorer' and 'Class View'. The 'Solution Explorer' pane shows the project structure, including the `prac01` project, which contains a `src` folder with `prac01.cpp` and `prac01.h`, and a `framework` folder with `framework.h`, `prac01.h`, `Resource.h`, and `targetver.h`. The 'Class View' pane shows the class hierarchy, with `prac01` as the root class. Below the class hierarchy, the 'Properties' window is open, showing the properties of the selected `prac01.cpp` file. The properties include: (이름) `prac01.cpp`, 내용 `False`, 상대 경로 `prac01.cpp`, 전체 경로 `C:\Users\USER\Desktop\Prac01\src\prac01.cpp`, 파일 형식 `C/C++ 코드`, and 프로젝트에 포함됨 `True`.

```
1 // prac01.cpp : 애플리케이션에 대한 진입점을 정의합니다.
2 //
3 //
4 // Created by Soo-Kyun Kim
5
6 #include "framework.h"
7 #include "prac01.h"
8 #include "gl/gl.h"
9 #include "gl/glu.h"
10
11 #define MAX_LOADSTRING 100
12
13 // 전역 변수:
14 HINSTANCE hInst; // 현재 인스턴스입니다.
15 WCHAR szTitle[MAX_LOADSTRING]; // 제목 표시줄 텍스트입니다.
16 WCHAR szWindowClass[MAX_LOADSTRING]; // 기본 창 클래스 이름입니다.
17
18 ////////////////////////////////////////////////// sk
19 HDC MyDC;
20 HGLRC MyRC;
21 bool bSetupPixelFormat(void);
22 void Resize(const int cx, const int cy);
23 void DrawScene(void);
24
25 // 이 코드 모듈에 포함된 함수의 선언을 전달합니다:
26 ATOM MyRegisterClass(HINSTANCE hinstance);
27 BOOL InitInstance(HINSTANCE, int);
28 LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
29 INT_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);
30
31 int APIENTRY wWinMain(_In_ HINSTANCE hinstance,
32                     _In_opt_ HINSTANCE hPrevinstance,
33                     _In_ LPWSTR lpCmdLine,
34                     _In_ int nCmdShow)
35 {
36 }
```

출력 보기 선택(S):

오류 목록 출력

준비 32

소스 제어에 추가

32

전역 변수 (Global Variables)

prac01.cpp

prac01

(전역 범위)

InitInstance(HINSTANCE)

```
1 // prac01.cpp : 애플리케이션에 대한 진입점을 정의합니다.
2 //
3 //
4 // Created by Soo-Kyun Kim
5
6 #include "framework.h"
7 #include "prac01.h"
8 #include "gl/gl.h"
9 #include "gl/glu.h"
10
11
12 #define MAX_LOADSTRING 100
13
14 // 전역 변수:
15 HINSTANCE hInst; // 현재 인스턴스입니다.
16 WCHAR szTitle[MAX_LOADSTRING]; // 제목 표시줄 텍스트입니다.
17 WCHAR szWindowClass[MAX_LOADSTRING]; // 기본 창 클래스 이름입니다.
18
19 ////////////////////////////////////////////////// sk
20 HDC hDeviceContext; // current device context
21 HGLRC hRenderingContext; // current rendering context
22
23 bool bSetupPixelFormat(HDC hdc);
24 void Resize(int width, int height);
25 void DrawScene(HDC MyDC);
26
27 // 이 보드 모듈에 포함된 함수의 선언을 전달합니다:
28 ATOM MyRegisterClass(HINSTANCE hInstance);
29 BOOL InitInstance(HINSTANCE, int);
30 LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
31 INT_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);
32
33 int APIENTRY wWinMain(_In_ HINSTANCE hInstance,
```

Window Procedure

```
prac01.cpp  X
prac01      (전역 범위)  InitInstance(HINSTANCE, int)

136  //
137  LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
138  {
139      RECT clientRect;
140      switch (message)
141      {
142      case WM_CREATE:
143          // Initialize for the OpenGL rendering
144          hDeviceContext = GetDC(hWnd);
145          if (!bSetupPixelFormat(hDeviceContext)) {
146              MessageBox(hWnd, "Error in setting up pixel format for OpenGL", "Error", MB_OK | MB_ICONERROR);
147              DestroyWindow(hWnd);
148          }
149          hRenderingContext = wglCreateContext(hDeviceContext);
150          wglMakeCurrent(hDeviceContext, hRenderingContext);
151          break;
152
153      case WM_SIZE:
154          GetClientRect(hWnd, &clientRect);
155          Resize(clientRect.right, clientRect.bottom);
156          InvalidateRect(hWnd, NULL, false);
157
158          break;
159      /*
```

bSetupPixelFormat () 1

```
prac01.cpp  x
prac01  (전역 범위)  InitInstance(HINSTANCE, int)

223     return (INT_PTR)FALSE;
224 }
225
226 bool bSetupPixelFormat(HDC hdc)
227 {
228     PIXELFORMATDESCRIPTOR pfd;
229     int pixelformat;
230
231     pfd.nSize = sizeof(PIXELFORMATDESCRIPTOR);
232     pfd.nVersion = 1;
233     pfd.dwFlags = PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL | PFD_DOUBLEBUFFER;
234     pfd.dwLayerMask = PFD_MAIN_PLANE;
235     pfd.iPixelFormat = PFD_TYPE_RGBA;
236     pfd.cColorBits = 24;
237     pfd.cDepthBits = 16;
238     pfd.cAccumBits = 0;
239     pfd.cStencilBits = 0;
240
241     if ((pixelformat = ChoosePixelFormat(hdc, &pfd)) == 0) {
242         MessageBox(NULL, "ChoosePixelFormat() failed!!!", "Error", MB_OK | MB_ICONERROR);
243         return false;
244     }
245
246     if (SetPixelFormat(hdc, pixelformat, &pfd) == false) {
247         MessageBox(NULL, "SetPixelFormat() failed!!!", "Error", MB_OK | MB_ICONERROR);
248         return false;
249     }
250
251     return true;
252 }
253
```

bSetupPixelFormat () 2

- 두 그래픽스 시스템을 연결하기 위한 중요한 사항
 - OpenGL에서 사용하기 위하여 GDI에서 제공하는 화소 형식(pixel format)을 적절히 지정해야 함
- OpenGL에서 사용해야할 프레임 버퍼에 대한 정보
 - 색깔 버퍼를 위해 화소 당 몇 비트를 사용할 것인가?
 - 더블 버퍼를 쓸 것인가?
 - 등등
- 프레임 버퍼는 하드웨어 성능에 의해 제한됨

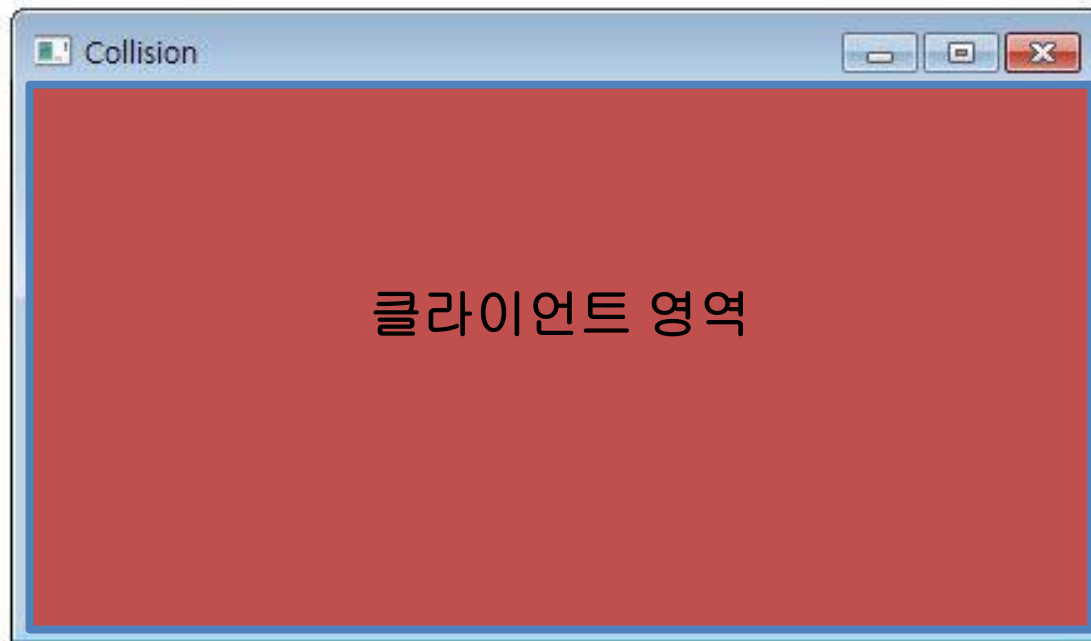
bSetupPixelFormat () 3

- 연결 방법

- OpenGL에서 희망하는 프레임 버퍼의 성능을 화소 형식을 통해 기술함
 - **bSetupPixelFormat()**
- GDI에 화소 정보를 넘겨주지만 실질적으로 하드웨어 성능이 따르지 못하면, 윈도우는 최대한 OpenGL이 요구하는 성능에 만족 시켜주는 화소형식을 결정함
- 윈도우 시스템의 그래픽스에 관련된 모든 정보를 저장하는 디바이스 문맥에 저장 후 위의 정보를 사용함

그리느냐!!! 마느냐!!! 1

- 윈도우 GDI(Graphics Device Interface)
 - 그래픽의 출력 담당
 - 프린터에 프린트하는 기능도 담당
- 우리가 그려야 하는 범위
 - 윈도우의 클라이언트 영역



그리느냐!!! 마느냐!!! 2

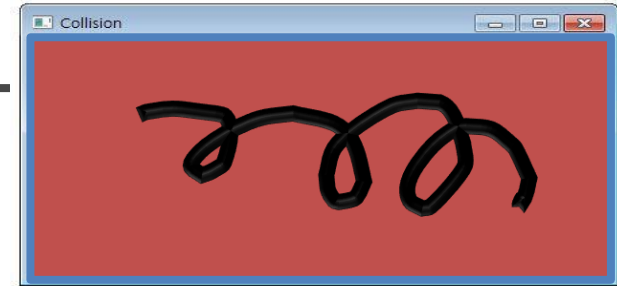
- 그리기 위해 중요한 개념

- 유효화(Validating)

- 윈도우의 클라이언트 영역을 훼손하면, 윈도우는 WM_PAINT 메시지를 응용 프로그램으로 보냄
 - 프로그램은 윈도우에게 그 메시지를 받아서 처리했다는 것을 알려야 함
 - 윈도우에게 알리는 과정을 사각형을 “유효화 한다” 라고 함

- 그려야 할 사각형의 정의

- 그릴 필요가 있는 영역인 무효 사각형 (Invalid rectangle)만 신경을 쓰면 됨
 - WM_PAINT 자료구조에 포함되어 있음



그리느냐!!! 마느냐!!! 3

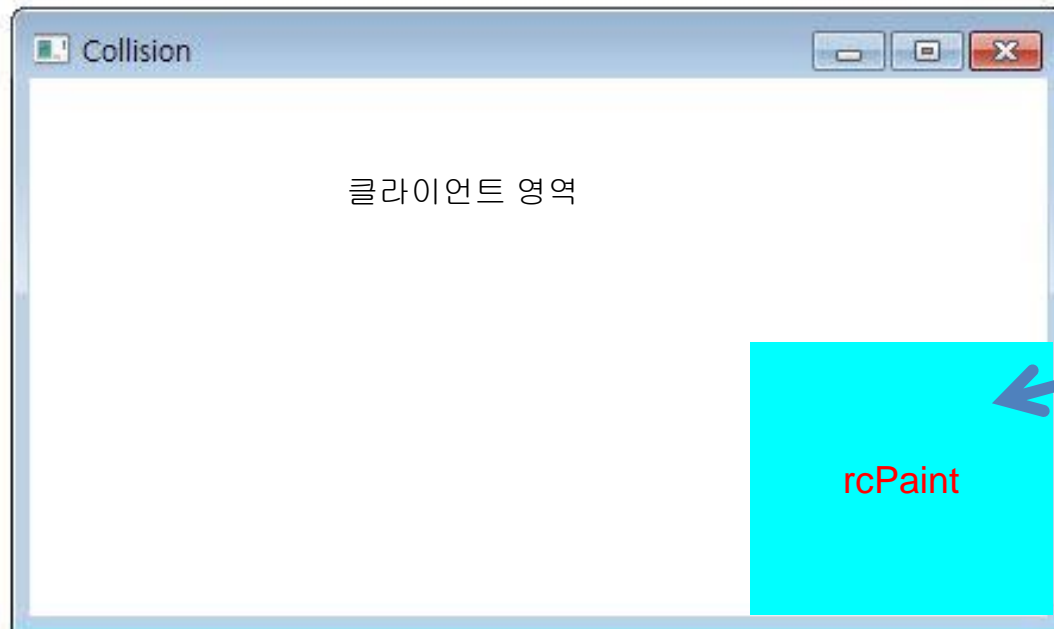
- **WM_PAINT**
 - BeginPaint() ... EndPaint()
 - GetDC() ... ReleaseDC()

그리느냐!!! 마느냐!!! 4

- WM_PAINT

- BeginPaint() ... EndPaint()

- 장점: 자동으로 유효화 (Validating) 수행
 - 단점: 무효사각형을 나타내는 rcPaint 속성을 가지고 있으며, 윈도우의 클리핑 영역 외부에는 어떤 것도 그릴 수 없음
 - 그리고 싶다면 rcPaint 를 다시 정의 해야함



이 영역은 다른 객체에
의해 무효화됨

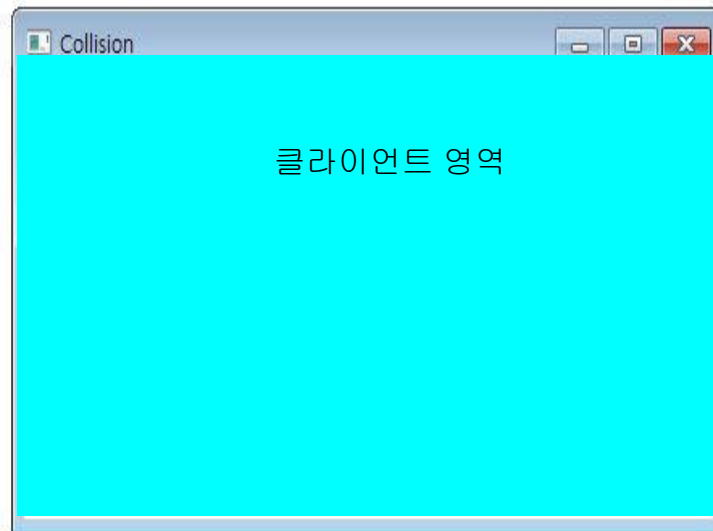
→ 이 영역만 다시
그리면 됨

그리느냐!!! 마느냐!!! 5

- WM_PAINT

- GetDC() ... ReleaseDC()

- 장점: 클리핑 사각형은 전체 클라이언트 영역이 됨. 따라서 클라이언트 영역을 그리기 위해 매번 무효 사각형을 다시 정의할 필요가 없음
 - 단점: 무효 사각형을 유효화 하기 위해 직접 **ValidateRect()**를 호출. 그렇지 않으면 윈도우는 언제까지나 응용 프로그램에 WM_PAINT 메시지를 보낼 것임 (**윈도우는 유효화가 이루어지지 않으면 계속 WM_PAINT 메시지를 보냄**)



InvalidateRect()

- BOOL InvalidateRect(

HWND hWnd, //무효화할 윈도우 핸들

CONST RECT *lpRect, // 무효화할 사각형 영역 포인터

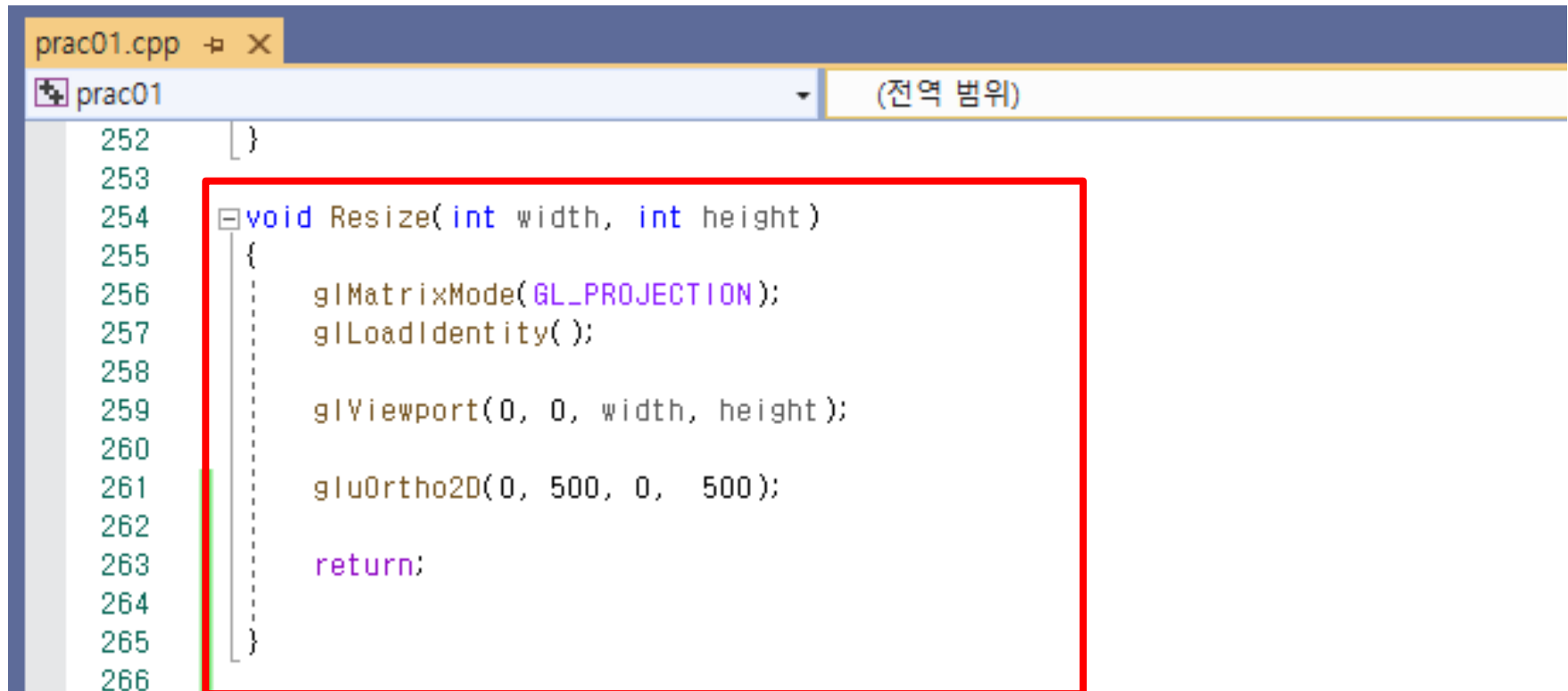
BOOL bErase); // BeginPaint()를 위해 플래그 지움

- 무효 사각형 (Invalid Rectangle) : 다시 그릴 필요가 있는 유일한 영역
- InvalidateRect()함수는 클라이언트 영역의 임의의 부분만 그림

Window Procedure

```
prac01.cpp  x
prac01      (전역 범위)  Ini
175      }
176      break;
177      /*
178      case WM_PAINT:
179      {
180          DrawScene(hDeviceContext);
181          ValidateRect(hWnd, NULL);
182
183          /*
184             PAINTSTRUCT ps;
185             HDC hdc = BeginPaint(hWnd, &ps);
186             // TODO: 여기에 hdc를 사용하는 그리기 코드를 추가합니다...
187             EndPaint(hWnd, &ps);
188             */
189      }
190      break;
191      case WM_DESTROY:
192          // Destroy all about OpenGL
193          if (hRenderingContext)
194              wglDeleteContext(hRenderingContext);
195          if (hDeviceContext)
196              ReleaseDC(hWnd, hDeviceContext);
197
198          PostQuitMessage(0);
199          break;
200      default:
201          return DefWindowProc(hWnd, message, wParam, lParam);
202      }
203      return 0;
204  }
205  }
```

ReSize () & DrawScene ()



The image shows a screenshot of a C++ code editor with a file named 'prac01.cpp'. The editor window has a tab labeled 'prac01' and a dropdown menu showing '(전역 범위)'. The code is displayed with line numbers from 252 to 266. A red rectangular box highlights the 'void ReSize(int width, int height)' function, which is defined between lines 254 and 265. The function body includes calls to 'glMatrixMode(GL_PROJECTION)', 'glLoadIdentity()', 'glViewport(0, 0, width, height)', and 'gluOrtho2D(0, 500, 0, 500)', followed by a 'return;' statement.

```
252     }  
253  
254     void ReSize(int width, int height)  
255     {  
256         glMatrixMode(GL_PROJECTION);  
257         glLoadIdentity();  
258  
259         glViewport(0, 0, width, height);  
260  
261         gluOrtho2D(0, 500, 0, 500);  
262  
263         return;  
264     }  
265  
266
```

ReSize () & DrawScene ()

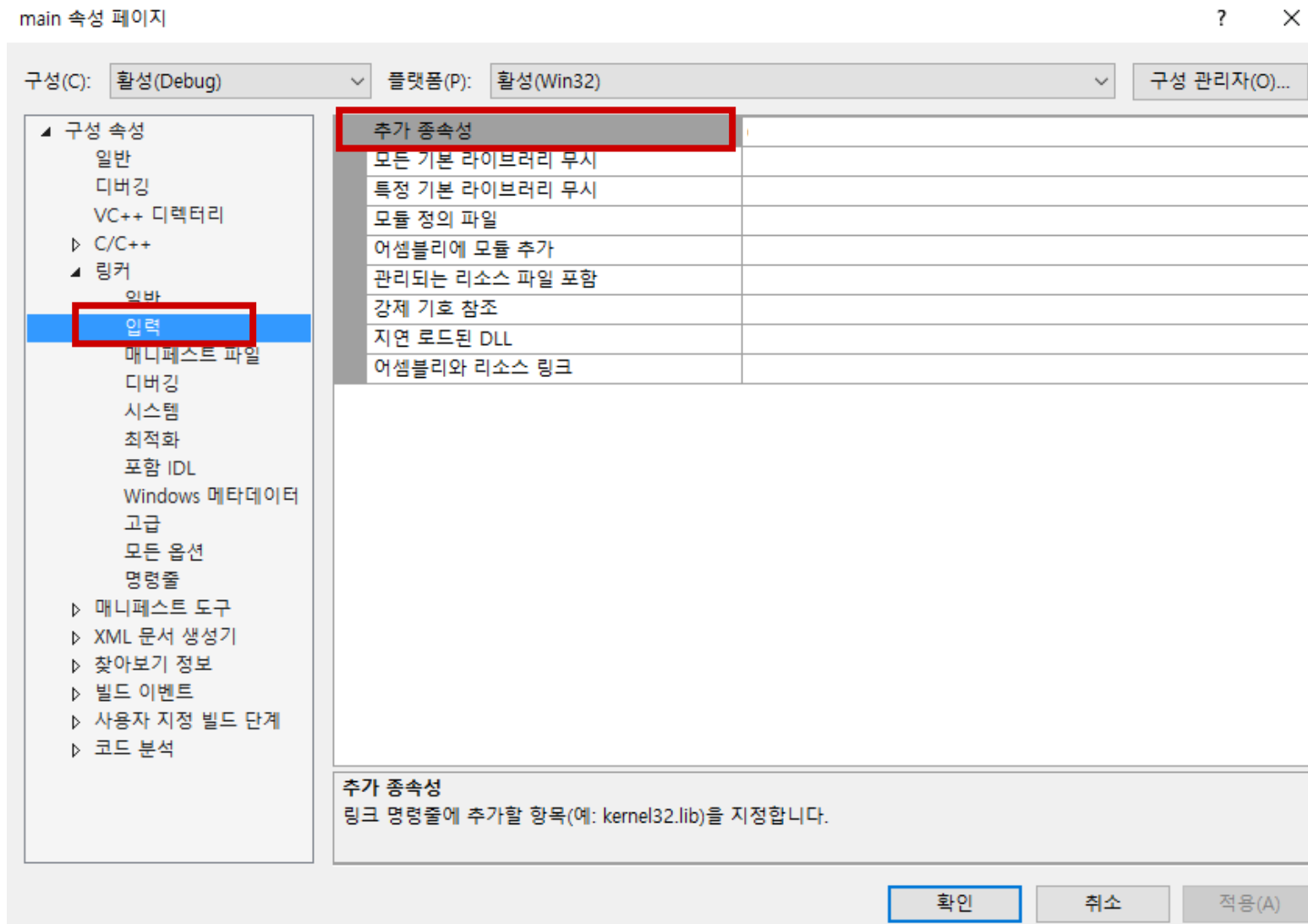
prac01.cpp

prac01

(전역 범위)

```
279
280  /*
281   * DrawScene : to draw a scene
282   */
283  void DrawScene(HDC MyDC)
284  {
285      glEnable(GL_DEPTH_TEST);
286
287      glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
288      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
289
290      glMatrixMode(GL_MODELVIEW);
291      glLoadIdentity();
292      SwapBuffers(MyDC);
293
294      return;
295  }
```

Project Settings (1/5)



Project Settings (2/5)

main 속성 페이지

? X

구성(C): **활성(Debug)**

플랫폼(P): **활성(Win32)**

구성 관리자(O)...

구성 속성

일반

디버깅

VC++ 디렉터리

▶ C/C++

링커

일반

입력

매니페스트 파일

디버깅

시스템

최적화

포함 IDL

Windows 메타데이터

고급

모든 옵션

명령줄

▶ 매니페스트 도구

▶ XML 문서 생성기

▶ 찾아보기 정보

▶ 빌드 이벤트

▶ 사용자 지정 빌드 단계

▶ 코드 분석

추가 종속성

모든 기본 라이브러리 무시

특정 기본 라이브러리 무시

모듈 정의 파일

어셈블리에 모듈 추가

관리되는 리소스 파일

강제 기호 참조

지연 로드된 DLL

어셈블리와 리소스

kernel32.lib;user32.lib;gdi32.lib;winspool.lib;comdlg32.lib;advapi32.lib;shell

추가 종속성

opengl32.lib
glu32.lib

평가 값:

opengl32.lib
glu32.lib

상속된 값:

kernel32.lib
user32.lib
gdi32.lib

☒ 부모 또는 프로젝트 기본값에서 상속()

매크로(M) >>

확인

취소

추가 종속성

링크 명령줄에 추가할 항목(예: kernel32.lib)을 지정합니다.

확인

취소

적용(A)

Project Settings (3/5)

main 속성 페이지

? X

구성(C): **활성(Debug)**

플랫폼(P): **활성(Win32)**

구성 관리자(O)...

▲ 구성 속성

일반

디버깅

VC++ 디렉터리

▶ C/C++

▲ 링커

일반

입력

매니페스트 파일

디버깅

시스템

최적화

포함 IDL

Windows 메타데이터

고급

모든 옵션

명령줄

▶ 매니페스트 도구

▶ XML 문서 생성기

▶ 찾아보기 정보

▶ 빌드 이벤트

▶ 사용자 지정 빌드 단계

▶ 코드 분석

추가 종속성

opengl32.lib;glu32.lib;%(AdditionalDependencies)

모든 기본 라이브러리 무시

특정 기본 라이브러리 무시

모듈 정의 파일

어셈블리에 모듈 추가

관리되는 리소스 파일 포함

강제 기호 참조

지연 로드된 DLL

어셈블리와 리소스 링크

추가 종속성

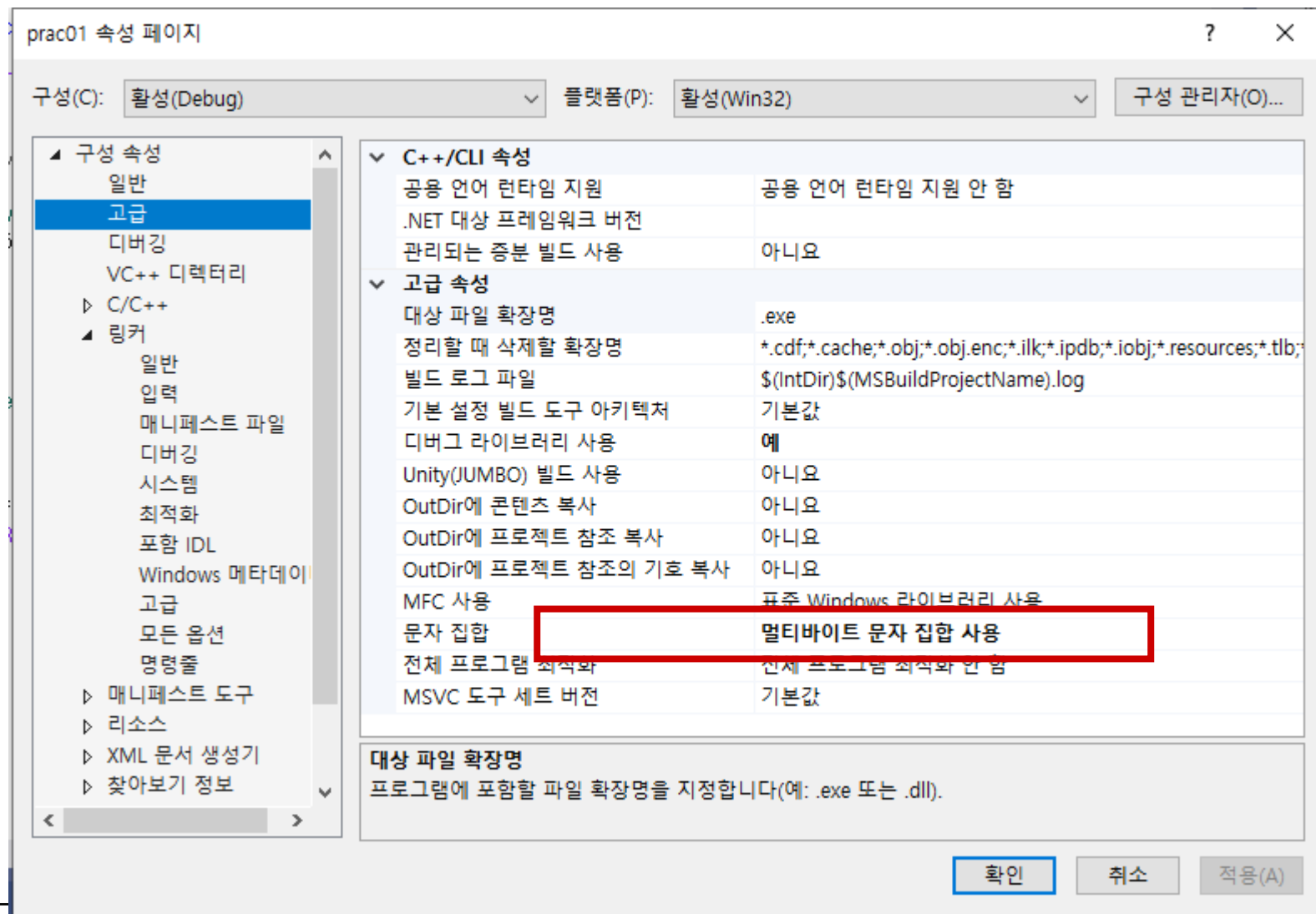
링크 명령줄에 추가할 항목(예: kernel32.lib)을 지정합니다.

확인

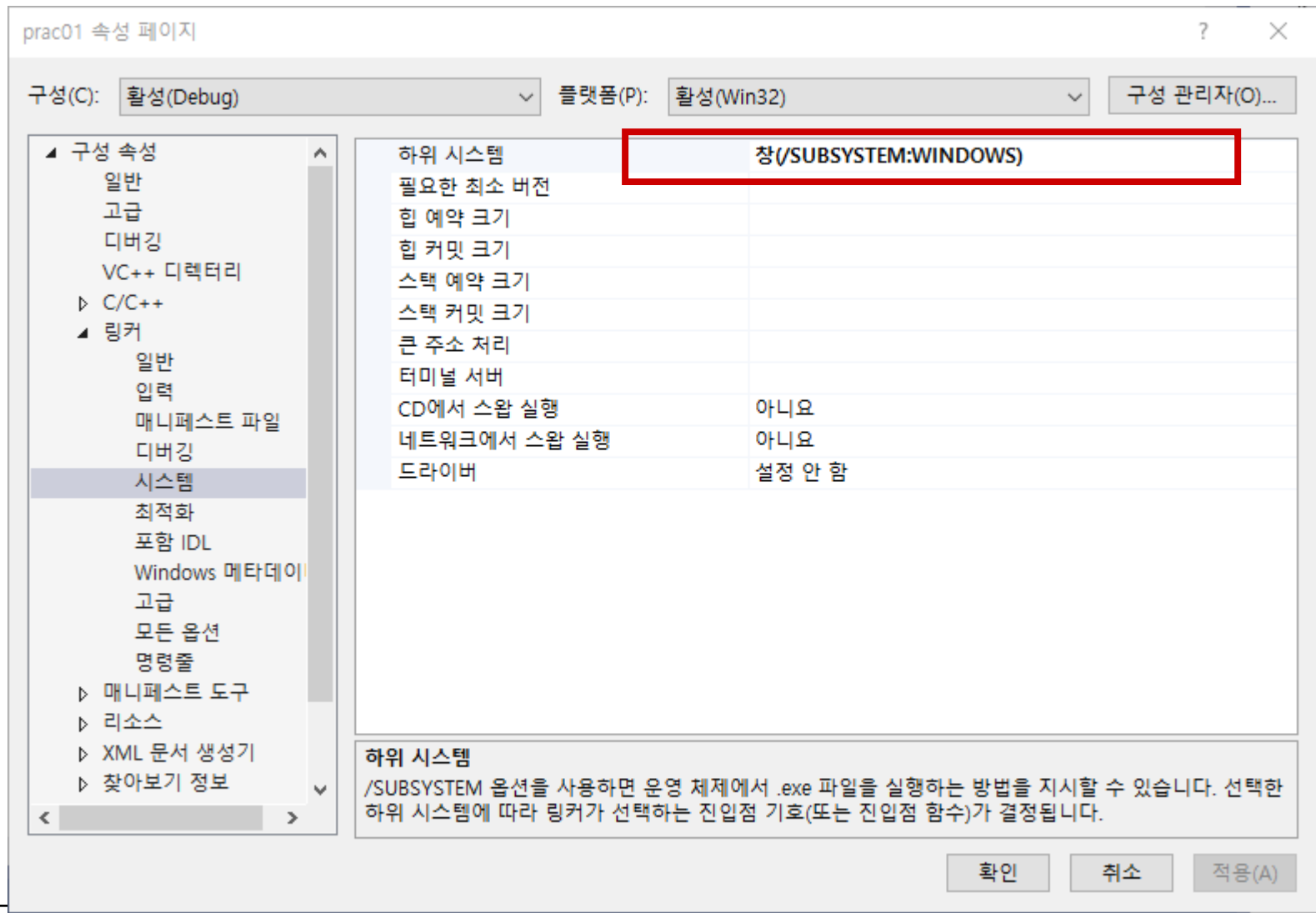
취소

적용(A)

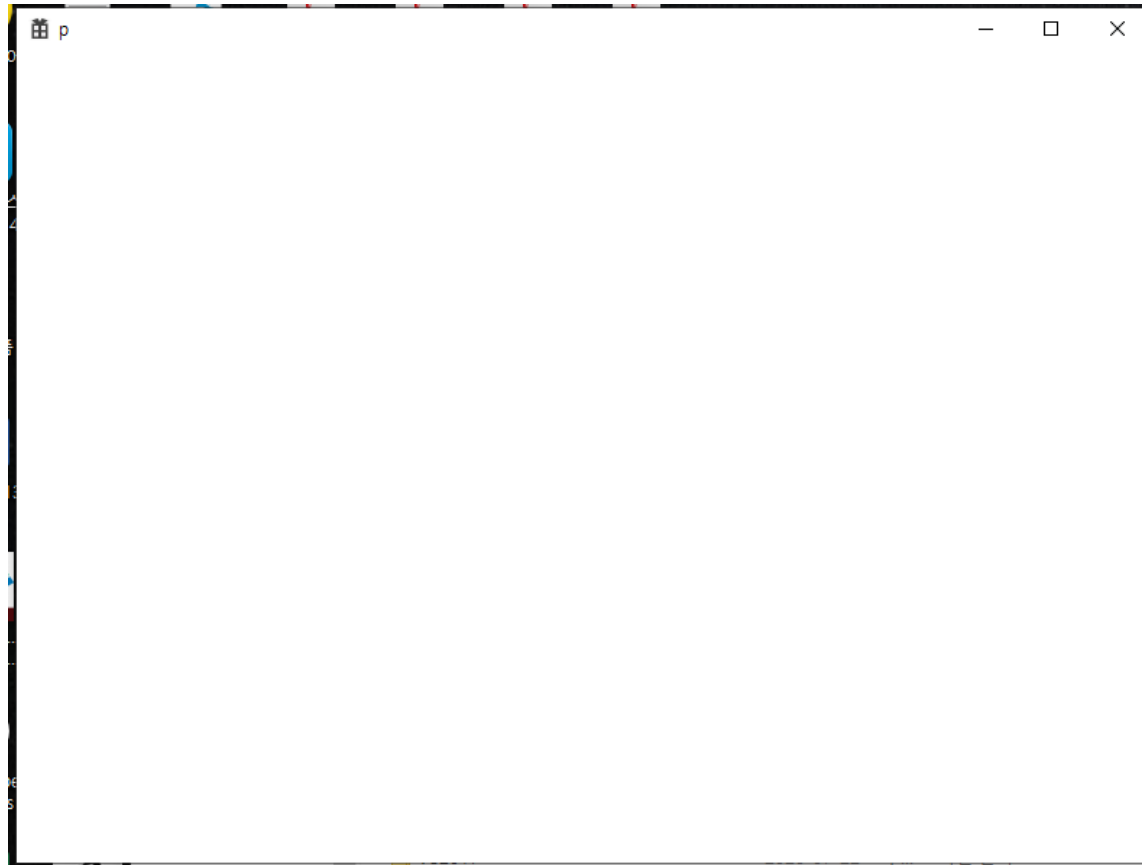
Project Settings (4/5)



Project Settings (5/5)



실행 결과



윈도우 크기: 800x600
메뉴 없앨 것

OpenGL function format

function name dimensions

belongs to GL library x,y,z are floats

`glVertex3f(x,y,z)`

The diagram shows the function `glVertex3f(x,y,z)` with four arrows pointing to its components: a blue arrow from 'belongs to GL library' to 'gl', a black arrow from 'function name' to 'Vertex', a red arrow from 'dimensions' to '3', and a green arrow from 'x,y,z are floats' to 'f'.

`glVertex3fv(p)`

p is a pointer to an array

The diagram shows the function `glVertex3fv(p)` with a green arrow pointing from 'p is a pointer to an array' to the 'v' in the function name.

The End