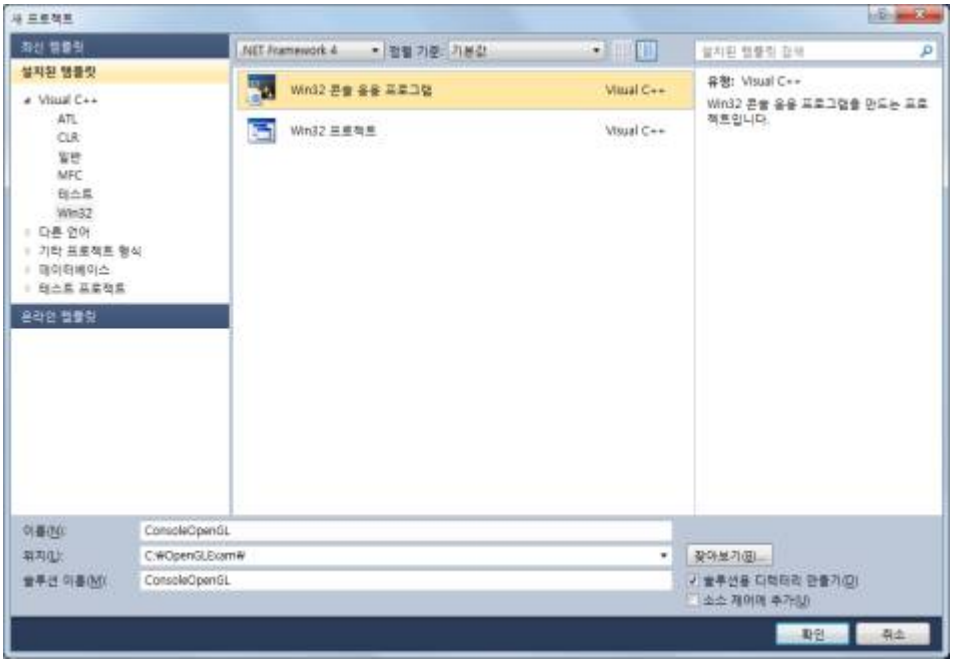
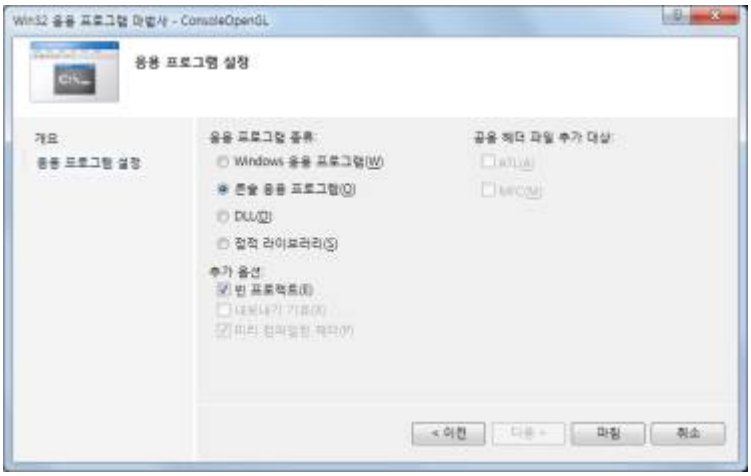
2.첫번째 예제

2-1.콘솔 프로젝트

첫 번째 OpenGL 예제를 만들어 보자. 첫 예제이므로 복잡한 도형 대신 삼각형 하나만 그려 보기로 한다. 개발 절차를 실습해 보는 것이 주된 목적이므로 소스 내용은 아직 몰라도 상관없다. 비주얼 스튜디오 2010을 실행하고 다음 실습을 진행한다. 메뉴에서 파일/새로만들기/프로젝트 항목을 선택한다.



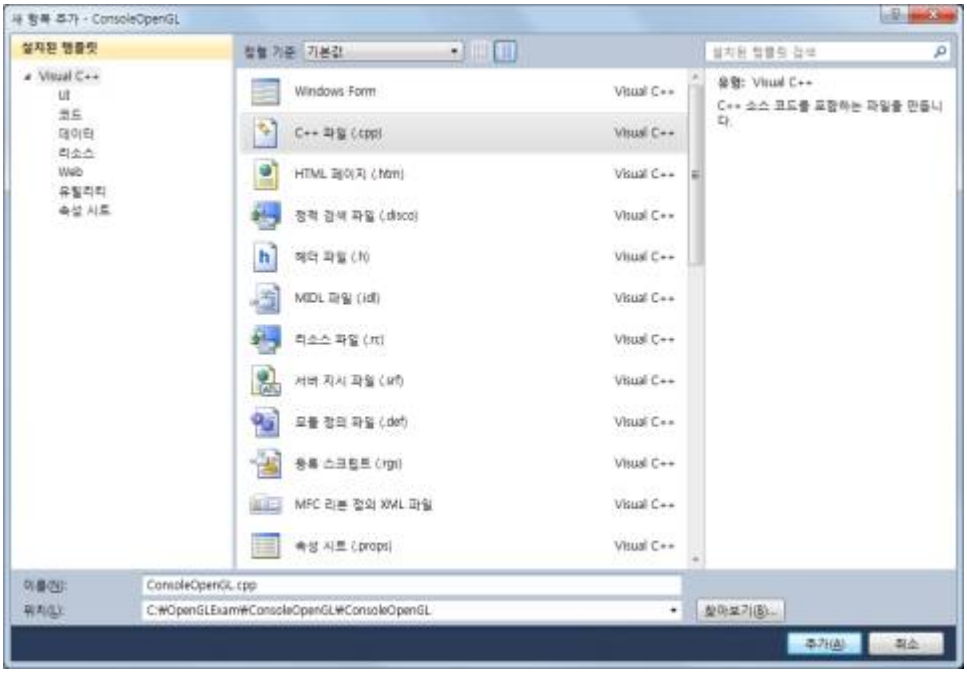
Win32 콘솔 응용 프로그램을 선택하고 프로젝트 이름은 ConsoleOpenGL로 입력한다. 적당한 실습 폴더를 선택하되 이 강좌는 C:₩OpenGLExam 폴더에 저장했다. 확인 버튼을 누르면 마법사가 나타나며 응용 프로그램 종류가 콘솔로 이미 선택되어 있다.



다른 옵션은 건드릴 필요없고 추가옵션에서 빈 프로젝트 체크 박스를 선택한다. 미리 제공되는 템플릿을 사용하는 것도 가능은 하다. 그러나 크로스 플랫폼을 고려해야 하므로 미리 컴파일된 헤더니 리소스니 하는 것들은 배제하는 것이 좋다.

마법사가 만들어 주는 프레임워크는 실무 프로젝트시에 상당히 편리하지만 이렇게 만들면 비주얼 스튜디오 전용의 예제가 되어 버린다. 소스 차원의 이식성 확보를 위해 가급적이면 순수한 C 코드로 작성하는 것이

유리하다. 프로젝트 껍데기만 만든 후 프로젝트/새 항목 추가를 선택하여 소스 파일을 만든다.



아무 내용도 없는 빈 소스 파일이 프로젝트에 추가된다. 이 파일에 다음과 같이 입력한다. OpenGL의 Hello World 버전이라고 할 수 있는데 최대한 간단하게 작성했다.

ConsoleOpenGL

```
#nclude <gl/glut.h>

void DoDisplay()
{
	glClear(GL_COLOR_BUFFER_BIT);

	glBegin(GL_TRIANGLES);
	glVertex2f(0.0, 0.5);
	glVertex2f(-0.5, -0.5);
	glVertex2f(0.5, -0.5);
	glEnd();
	glFlush();
}

void main()
{
	glutCreateWindow("OpenGL");
	glutDisplayFunc(DoDisplay);
	glutMainLoop();
}
```

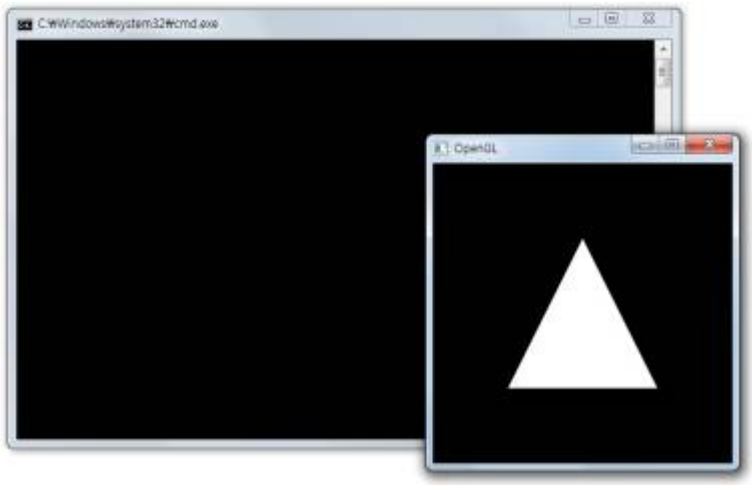
OpenGL을 사용하려면 주요 헤더 파일인 gl.h와 glu.h도 포함시키는 것이 원칙적이나 이 예제는 glut.h 하나만 포함시켰다. glut 헤더 파일을 열어 보면 다음과 같은 구문이 있기 때문이다. 알아서 필요한 파일을 포함시켜 주므로 이 헤더 파일 하나만 포함하면 된다.

```
#nclude <GL/gl.h>
#nclude <GL/glu.h>
```

외부 라이브러리를 사용하려면 임포트 라이브러리도 연결해야 한다. 프로젝트 옵션의 설정 페이지에서 opengl32.lib, glu32.lib, glut32.lib를 링크해야 하는데 glut.h에 다음 구문이 작성되어 있으므로 그러지 않아도 상관없다.

```
#pragma comment (lib, "opengl32.lib")   /* link with Microsoft OpenGL lib */
#pragma comment (lib, "glu32.lib")      /* link with OpenGL Utility lib */
#pragma comment (lib, "glut32.lib")     /* link with Win32 GLUT lib */
```

이 구문은 비주얼 C++의 확장 문법이며 헤더 파일에서 필요한 설정을 직접 변경하므로 아주 편리하다. 결국 OpenGL을 사용하기 위한 준비는 glut.h 헤더 파일 하나만 포함하면 되는 셈이다. 컴파일하여 실행해 보자. 콘솔 프로젝트이므로 콘솔창이 먼저 뜨고 별도의 그래픽창이 실행되어 흰색 삼각형 하나를 그릴 것이다.

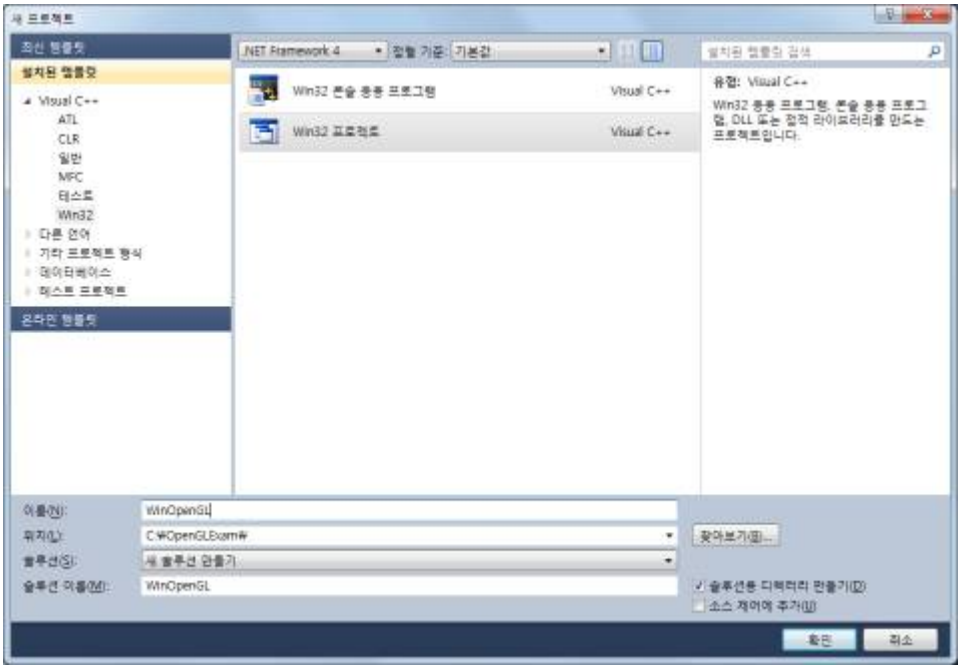


뒤쪽의 콘솔창은 별 하는 일 없이 그냥 실행된 것이고 삼각형이 그려진 윈도우가 OpenGL창이다. 삼각형을 출력하기만 할 뿐 아직 별다른 코드가 없으므로 입력에는 전혀 반응하지 않는다. 창의 크기를 조정하면 삼각형 크기도 비례적으로 바뀐다.

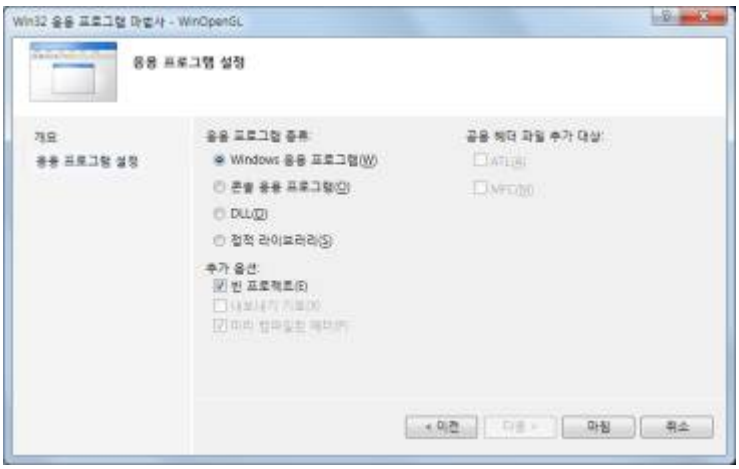
2-2.윈도우 프로젝트

콘솔 프로젝트도 Win32 응용 프로그램이고 glut가 그래픽 창을 생성해 주므로 실습에 별 문제는 없다. 형태가 단순해서 소스양이 적고 C 표준을 준수하므로 다른 운영체제나 컴파일러에서도 수정없이 컴파일된다. 한마디로 이식성이 뛰어나다. 그러나 콘솔창이 뜬 후 실행 윈도우가 뜬다는 면에서 보기 좋지 않고 테스트 후 콘솔창도 직접 닫아야 한다는 면에서 불편하다.

다음은 윈도우 버전의 예제를 작성해 보자. 새 프로젝트를 만들되 Win32 프로젝트 항목을 선택하고 프로젝트 이름은 WinOpenGL로 입력한다.



마법사는 응용 프로그램 종류 옵션을 Windows 응용 프로그램으로 선택해 준다. 콘솔 프로젝트와 마찬가지로 아래쪽의 빈 프로젝트 옵션을 선택한다.



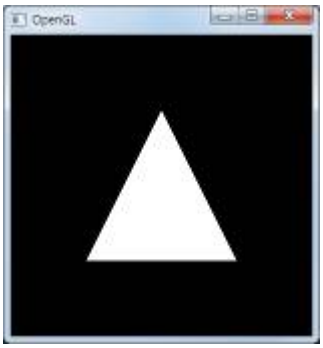
WinOpenGL.cpp 파일을 프로젝트에 추가하고 다음 소스를 작성한다.

WinOpenGL

```
#nclude <windows.h>
#nclude <gl/glut.h>

void DoDisplay()
{
	glClear(GL_COLOR_BUFFER_BIT);

	glBegin(GL_TRIANGLES);
	glVertex2f(0.0, 0.5);
	glVertex2f(-0.5, -0.5);
	glVertex2f(0.5, -0.5);
	glEnd();
	glFlush();
}
```

```
int APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance
        ,LPSTR lpszCmdParam,int nCmdShow)
{
    glutCreateWindow("OpenGL");
    glutDisplayFunc(DoDisplay);
    glutMainLoop();
    return 0;
}
```

콘솔 프로젝트와 소스 내용은 거의 동일하다. 진입점이 main에서 WinMain으로 바뀌었고 WinMain의 인수에 사용된 타입들을 위해 windows.h를 포함한 정도만 다르다. 실행해 보자.



실행 결과는 동일하되 콘솔창이 열리지 않고 바로 OpenGL 윈도우가 열리므로 훨씬 더 깔끔하고 편리하다. 그러나 WinMain은 윈도우즈 운영체제에서만 존재하는 진입점이므로 리눅스나 매킨토시에서는 이 예제를 실행해 볼 수 없다는 맹점이 있다. 타 환경에서는 WinMain을 main으로 바꿔 줘야만 컴파일된다. 콘솔 프로젝트가 좀 구닥다리이고 창이 두개나 열려 불편하지만 이식성은 훨씬 더 좋다.

사실 두 프로젝트 형식은 거의 차이가 없다. 이 책에서는 윈도우 버전으로 예제를 만들기로 한다. 왜냐하면 대부분의 사람들이 윈도우즈 환경에서 실습을 할 것이고 매 실습마다 창을 두 개씩이나 열었다 닫았다 하는 것이 너무 불편하기 때문이다. 2~3개의 예제를 같이 실행해 놓고 비교할 때는 창이 4개, 6개씩이나 열리므로 정신이 엄청 사납다. 실무 프로젝트가 아니라 학습용이므로 학습에 효율적인 쪽으로 선택했다.

2-3. Dev-C++

다음은 공개 컴파일러 Dev-C++로 프로젝트를 만들어 보자. 사실 비주얼 스튜디오는 OpenGL 실습용으로 사용하기에는 너무 덩치가 크고 유료 프로그램이라 껄쩍지근한 면이 있다. Dev-C++ 4992를 설치한 후 다음 순서대로 프로젝트를 작성한다. 콘솔 버전도 작성할 수 있지만 윈도우 버전을 바로 만들어 보자. 파일/새로 만들기/프로젝트 항목을 선택한다.

위쪽의 템플릿에서 Windows Application 항목을 선택하고 프로젝트 이름은 DevOpenGL로 준다. 확인 버튼을 누르면 프로젝트를 저장하는데 프로젝트 폴더를 별도로 생성해 주지 않으므로 WinOpenGL 폴더를 생성한 후 저장하는 것이 좋다. 탐색기를 굳이 열 필요는 없고 저장 대화상자 위쪽의 새 폴더 버튼을 누르면 된

다. C:₩OpenGLExam 폴더로 이동한 후 DevOpenGL 이라는 폴더를 만들고 이 폴더 안에 DevOpenGL.dev 파일을 저장한다.

main.cpp가 기본적으로 제공되며 이 소스에는 일반적인 Win32 프로젝트가 이미 입력되어 있다. OpenGL과는 맞지 않으므로 전부 지운 후 다시 입력한다.

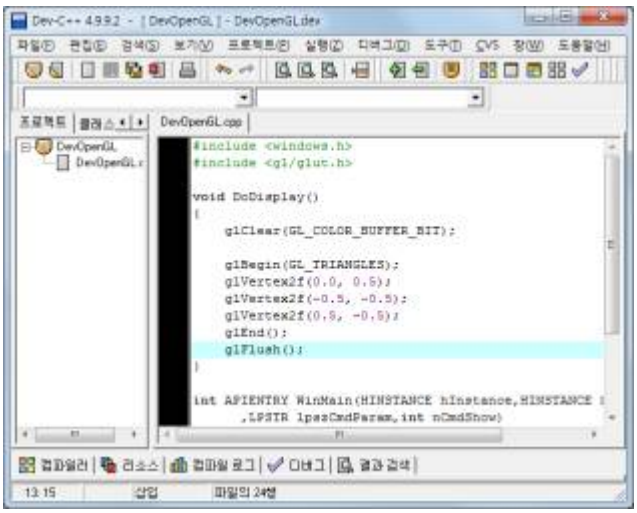
DevOpenGL

```
#nclude <windows.h>
#nclude <gl/glut.h>

void DoDisplay()
{
	glClear(GL_COLOR_BUFFER_BIT);

	glBegin(GL_TRIANGLES);
	glVertex2f(0.0, 0.5);
	glVertex2f(-0.5, -0.5);
	glVertex2f(0.5, -0.5);
	glEnd();
	glFlush();
}

int APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance
	,LPSTR lpszCmdParam,int nCmdShow)
{
	glutCreateWindow("OpenGL");
	glutDisplayFunc(DoDisplay);
	glutMainLoop();
	return 0;
}
```

비주얼 C++ 프로젝트의 소스와 단 한글자도 틀리지 않고 완전히 동일하다. 디폴트 파일명이 main.cpp로 되어 있는데 이 파일명을 그대로 사용해도 상관은 없지만 차후 프로젝트 관리시에 불편하다. 저장 버튼을 눌러 DevOpenGL.cpp 파일로 이름을 바꾸어 저장하는 것이 좋다.

임포트 라이브러리를 연결하는 #pragma comment 문은 C 표준 문법이 아니며 비주얼 C++의 임의 확장이라 Dev-C++에서는 동작하지 않는다. 그래서 OpenGL 라이브러리 연결은 수작업으로 직접 해야 한다. 프로젝트 옵션창의 매개변수 탭의 링커란에 다음 옵션을 추가한다.

ㅢ(대문자 아이가 아니라 소문자 엘) 다음에 연결할 라이브러리 이름을 적어 주면 된다. 여기까지 작업하면 개발 환경은 다음과 같을 것이다.



이 상태에서 컴파일 및 실행하면 비주얼 C++과 동일한 실행 파일이 만들어진다. 비주얼 C++에 비해 잔손질이 더 필요하고 다소 불편한 면이 있지만 일반적인 콘솔, 윈도우즈 예제 작성에는 별 부족함이 없다.

이 외에 즐겨 사용하는 컴파일러가 있다면 비슷한 방법으로 실습할 수 있다. 헤더 파일 구조나 임포트 라이브러리 연결 방법 정도만 다를 뿐이지 소스 내용은 동일하다. 손에 익은 컴파일러가 있다면 그 컴파일러를 사용하기 바란다.

2-4.첫번째 예제 분석

첫 번째 예제를 대충 분석해 보자. 상세한 함수 구문은 차후 천천히 알아 보기로 하고 전체적인 구조만 분석해 본다. 메인 함수의 코드는 다음 세 줄로 되어 있는데 함수의 접두가 모두 glut로 되어 있다. 즉 이 함수들은 그래픽 출력 환경을 만들어 주는 GLUT 명령어들이다.

```
glutCreateWindow("OpenGL");
glutDisplayFunc(DoDisplay);
glutMainLoop();
```

glutCreateWindow 함수는 OpenGL 출력을 위한 윈도우를 생성한다. 인수로는 타이틀 바에 출력될 제목 문자열을 전달한다. 이 함수에 의해 300 * 300 크기의 그리기 표면을 가지는 윈도우가 생성되며 이후 이 윈도우에 3차원 그래픽이 출력된다.

glutDisplayFunc 함수는 윈도우에 그리기를 전담하는 함수(WM_PAINT 메시지 처리 함수)를 지정한다. 윈도우가 처음 생성될 때, 크기가 변할 때, 출력 내용을 바꾸었을 때 등에 그리기 함수가 호출된다. 이 예에서는

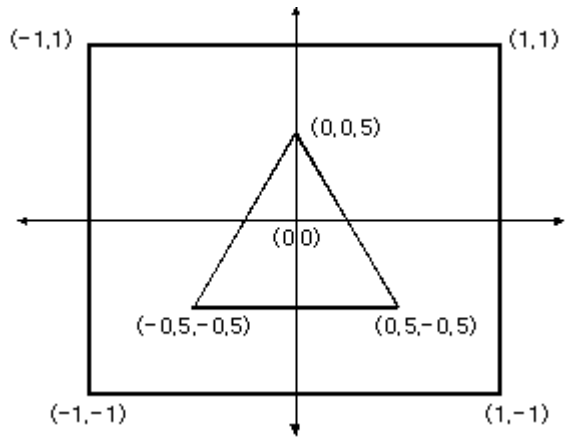
DoDisplay 함수를 지정했다. DoDisplay 함수는 메인 함수 바로 위에 정의되어 있으며 여기서 삼각형을 그린다.

glutMainLoop 함수는 메시지 루프를 돌린다. 그래픽 화면은 그리기, 창 크기 변화, 사용자 입력 등의 사건에 대해 반응하는 이벤트 드리븐 방식으로 동작하며 메시지를 받았을 때 적당한 함수를 호출해야 한다. 이 함수에 의해 윈도우 생성 직후에 DoDisplay가 호출되고 그림이 그려지며 실행을 시작한다. 메시지 루프는 프로그램이 종료될 때까지 계속 실행된다.

진입 함수는 윈도우 생성, 그리기 함수 지정, 메시지 루프 실행 3가지 명령을 실행함으로써 프로그램을 초기화했다. OpenGL 예제를 위한 최소한의 명령인 셈이다. 이 외에 특수한 초기화 명령이나 콜백 함수 지정 등이 더 들어갈 수 있다. 키보드 입력을 받으려면 glutKeyboardFunc으로 키보드를 처리할 함수를 지정하는 식이다. 이 예제는 단순한 그리기만 하므로 그리기 콜백 함수만 지정했다.

다음은 실제 그리기를 수행하는 DoDisplay 함수를 분석해 보자. 사용자 정의 함수이므로 이름은 물론 마음대로 정할 수 있다. 메인 함수 바로 위쪽에 정의되어 있으며 별도의 인수는 받아들이지 않는다. 출력 내용이 바뀔 때마다 화면을 깔끔하게 지우기 위해 glClear 함수를 호출한다. glBegin은 도형을 그리기 시작한다는 뜻이고 glEnd는 도형을 다 정의했다는 뜻이다. 이 두 함수 호출문 사이에 도형을 구성하는 정점들이 배치된다.

정점들을 어떻게 연결할 것인가는 glBegin의 인수로 지정하는데 GL_TRIANGLES로 주었으므로 세 정점들이 연결되어 삼각형이 그려진다. OpenGL의 좌표계는 다양하게 정의할 수 있지만 별다른 지정이 없을 때의 디폴트 좌표계는 다음과 같다. z 축은 일단 무시하고 x, y 평면축만 보자. 물론 어디까지나 디폴트일 뿐이며 좌표계는 다양한 방식으로 변형 가능하다.



화면의 중앙이 (0,0)이고 X 축은 오른쪽으로, Y 축은 위쪽으로 증가한다. 오른쪽 위 모서리 좌표는 (1,1)이며 왼쪽 아래는 (-1, -1)이다. 임의의 한 점은 -1 ~ 1 사이의 실수값으로 지정한다. glBegin과 glEnd 사이의 glVertex2f 함수로 지정한 세 점이 삼각형의 꼭지점을 가리키고 있으며 이 세 점을 연결하여 삼각형을 그린다. x, y 두 개의 값만 지정했으므로 z 좌표는 자동으로 0이 된다.

다 그린 후에 glFlush로 그리기 명령을 그래픽 카드로 보내 그린다. OpenGL은 속도를 높이기 위해 매 그리기 명령들을 즉시 수행하지 않고 버퍼에 모아 두었다가 한꺼번에 수행한다. glFlush는 버퍼를 비워 명령을 즉시 실행시킨다. 이 호출을 생략하면 정점만 정의될 뿐 출력은 나가지 않는다.

2-5.배포 예제

뭔가를 처음 배울 때는 모든 예제를 직접 만들어 보는 것이 좋다. 눈으로 훑어 보는 것과 손가락으로 직접 입력해 가면 코드 한줄 한줄을 읽는 것은 실습의 깊이가 확연하게 다르다. 문제는 시간이다. 우리 모두 그렇게 한가한 사람이 아니어서 프로젝트 만들고 소스 입력해 넣고 요모 조모 뜯어볼 시간이 없다. 당장 다음 주까지 결과가 나오지 않으면 엄청남 갈굼을 각오해야 하는 사람들 아닌가?

그래서 이 강좌는 모든 예제를 컴파일 가능한 프로젝트로 제공한다. 컴파일러 설치하고 프로젝트 열어서 직접 컴파일해 볼 수도 있고 의심가는 부분은 마음 내키는데로 뜯어 볼 수도 있다. 그러나 이 또한 시간이 많

이 걸리는 일이다. 그래서 재컴파일하지 않고도 모든 예제를 가급적이면 신속하게 살펴볼 수 있도록 예제를 작성했다.

OpenGL의 예제들은 대부분 그리기 코드가 주이므로 달라지는 부분은 DoDisplay 출력 함수의 코드뿐이다. 프로젝트의 골격은 거의 비슷하며 무엇을 어떻게 그리는가만 달라질 뿐이다. 그래서 예제를 너무 많이 만들지 않고 한 예제에 비슷 비슷한 코드를 통합적으로 작성해 두었다. 각각의 코드를 주석으로 묶어두는 방법도 흔히 사용되지만 그러자면 주석 편집 후 재컴파일이라는 번거로운 과정을 거쳐야 한다.

이 강좌는 팝업 메뉴로 강좌를 선택하는 방식을 사용한다. 실행중에도 언제든지 팝업 메뉴로 실행 코드를 바꿔 볼 수 있다. 팝업 메뉴에서 Action이라는 변수를 변경하고 DoDisplay 함수는 Action 변수에 따라 다른 코드를 실행하는 식이다. 다음 절에서 만들어 볼 ChangeState 예제의 전체 소스는 다음과 같이 작성되어 배포된다. 세 개의 코드가 한 예제에 작성되어 있는 셈이다.

```
#nclude <windows.h>
#nclude <gl/glut.h>

void DoDisplay();
void DoMenu(int value);
int Action;

int APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance
        ,LPSTR lpszCmdParam,int nCmdShow)
{
      glutCreateWindow("OpenGL");
      glutDisplayFunc(DoDisplay);
      glutCreateMenu(DoMenu);
      glutAddMenuEntry("빨간색 배경",0);
      glutAddMenuEntry("빨간색 삼각형",1);
      glutAddMenuEntry("오각형 그리기",2);
      glutAttachMenu(GLUT_RIGHT_BUTTON);
      glutMainLoop();
      return 0;
}

void DoMenu(int value)
{
      if (value < 100) {
            Action = value;
            glClearColor(0.0, 0.0, 0.0, 1.0);
            glColor3f(1.0, 1.0, 1.0);
            glutPostRedisplay();
            return;
      }
}

void DoDisplay()
{
switch(Action) {
case 0:
      ....
      break;
case 1:
      ....
      break;
case 2:
      ....
      break;
}
```

```
}
```

WinMain은 잘 변하지 않으므로 위쪽에 정의하고 DoDisplay는 아래쪽에 두되 미리 원형을 선언해 두었다. DoDisplay 함수는 switch 문으로 Action에 따라 다른 동작을 한다. case 문의 코드가 하나의 예제에 대응된다. 실행중에 언제든지 팝업 메뉴를 열면 Action을 바꿀 수 있는 명령들이 나타나며 친절하게 한글로 설명도 붙여 놓았다.

본문에서 코드를 보일 때는 case 문 안쪽의 코드만 보일 것이다. 여러 개의 코드를 하나의 예제에 묶다 보니 다소 부자연스러운 면도 있다. Action을 변경할 때마다 다른 예제가 변경한 설정을 원래대로 돌리는 코드가 필요하다. 또 case문안에 변수 초기화를 할 수 없으므로 추가적은 { } 괄호가 사용되기도 한다. switch문은 어디까지나 예제 선택을 위한 구문이므로 들여 쓰기도 하지 않았다.

실행중에 회전이나 인수값의 증감을 위해 알파벳키도 종종 사용한다. 키는 예제마다 조금씩 다른데 개별 예제에서 설명한다. 배포 예제의 형식이 다소 비상식적이지만 이렇게 하지 않으면 예제의 수가 너무 많아진다. 프로그래밍에 대한 기본적인 개념이 있다면 배포 예제의 형식은 어렵지 않고 수긍될 것이다. 모든 예제가 실행 파일안에 다 들어 있으므로 컴파일러를 열지 않고 강좌와 실행 파일만 보아도 내용을 이해할 수 있다.

앞부분 처음의 실습 예제 몇 개를 제외하고 나머지 예제는 모두 비주얼 스튜디오의 솔루션으로 묶어 두었다. 컴파일해 볼 때도 프로젝트를 교체할 필요없이 팝업 메뉴에서 시작 프로젝트만 변경함으로써 살펴볼 수 있다. 안드로이드 예제도 하나의 통합 예제에 목록 형식으로 작성되어 있으며 액티비티 하나가 예제 하나에 대응된다.