FreeGLUT 기반 OpenGL 개발환경 구축하기

김준호

Abstract

Ubuntu 16.04 LTS에서 FreeGLUT를 이용하여 OpenGL 기반 그래픽스 개발 환경을 구축하는 방법을 살펴본다.

1 OpenGL

OpenGL은 산업계 표준 삼차원 그래픽스 API 중 하나이다. 특히, OpenGL API는 플랫폼 독립적으로 설계되어 있어, OpenGL로 구성한 그래픽스 어플리케이션은 다양한 운영체제에서 특별한 수정없이 돌아갈 수 있다.

여기서는 국민대학교 소프트웨어학부의 공식 실습환경인 Ubuntu 16.04 LTS에서 FreeGLUT를 이용하여 OpenGL 기반 그래픽스 개발 환경을 구축하고 활용하는 방법을 학습한다.

2 FreeGLUT

OpenGL 프로그램은 일반적으로 시스템에 탑재된 그래픽카드를 구동하여 OpenGL이 관리하는 프레임버퍼에 원하는 그림을 그리는 방식으로 돌아간다. 그런데, OpenGL로 그린 프레임버퍼의 내용이 사용자가 쓰고 있는 운영체제의 화면에 반영되기 위해서는 굉장히 복잡한 부가적인 과정이 필요하다. GLUT는 이러한 복잡한 과정을 단순화시켜 놓은 OpenGL 개발용 유틸리티 툴킷 중 하나이다.

GLUT는 Mark J. Kilgard에 의해서 1994년에 만들어진 라이브러리이다. 굉장히 오래전에 만들어진 라이브러리이기 때문에 더 이상 업데이트가 이루어지지 않고 있다. 대신 동일한 기능을 가진 오픈소스 라이브러리인 FreeGLUT가 등장하여 계속적인 버전 업데이트가 이루어지며 널리 쓰이고 있다. 따라서 GLUT 대신 FreeGLUT를 쓰도록 하자.

2.1 FreeGLUT 개발환경 설치

Ubuntu 16.04 LTS에서 다음과 같이 apt-get을 이용하여 FreeGLUT 기반 개발환경을 손쉽게 설치할 수 있다.

```
sudo apt-get install freeglut3-dev
```

FreeGLUT 개발환경 설치가 끝나면 Ubuntu 16.04 LTS는 내부적으로 크게 다음의 두가지 내용이 업데이트 된다.

- /usr/include/GL/ 아래에 있는 헤더 파일들
- /usr/lib/x86_64-linux-gnu/ 아래에 있는 라이브러리 파일들 (* 64-bit Ubuntu 16.04 LTS의 경우)

FreeGLUT를 깔아서 생기는 헤더 파일과 라이브러리 파일은 이후, g++ 컴파일러를 이용해서 FreeGLUT 기반 프로그램을 생성하고자 할 때 활용될 것이다.

3 Hello World 작성

OpenGL 개발환경과 FreeGLUT 개발환경의 설치가 끝났으면, 간단한 FreeGLUT 기반 프로그램을 작성해 보자.

3.1 소스코드 작성

에디터를 통해 C++ 파일(예: hello_world.cpp)을 Fig. 1과 같이 작성한다.

3.2 소스컴파일 및 실행

터미널에서 g++ 컴파일러를 이용해서 다음과 같이 hello_world.cpp 파일을 컴파일 해 보자.

```
g++ hello_world.cpp -o hello -lglut -lGL
```

컴파일이 성공적으로 끝나면 현재 디렉토리에 실행가능한 파일인 hello가 생성된다. 만일 컴파일 에러가 생겼다면 C++ 소스파일에 오타가 있거나 g++ 컴파일 옵션에 오타가 있는 경우이다. 특히 대소문자를 구분하니 오타가 있는지 여부를 잘 살펴보도록 하자.

컴파일 후 만들어진 실행파일을 다음과 같이 터미널에서 실행시키면 Fig. 2와 같은 화면이 뜬다.

```
./hello &
```

여기까지 성공적으로 실행되었다면, Ubuntu 16.04 LTS에서 정상적으로 OpenGL 기반 개발환경이 구축된 것이다.

국민대학교 소프트웨어학부, junho@kookmin.ac.kr

```cpp
// hello_world.cpp

#include <GL/freeglut.h>
#include <iostream>

void init();
void mydisplay();

int main(int argc, char* argv[])
{
  glutInit(&argc, argv);
  glutInitWindowSize(500, 500);
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
  glutCreateWindow("Hello FreeGLUT");

  init();
  glutDisplayFunc(mydisplay);
  glutMainLoop();

  return 0;
}

void init()
{
  glClearColor(0.5f, 0.5f, 0.5f, 1.0f);
}

void mydisplay()
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  glutSwapBuffers();
}
```

Fig. 1. hello_world.cpp 소스파일

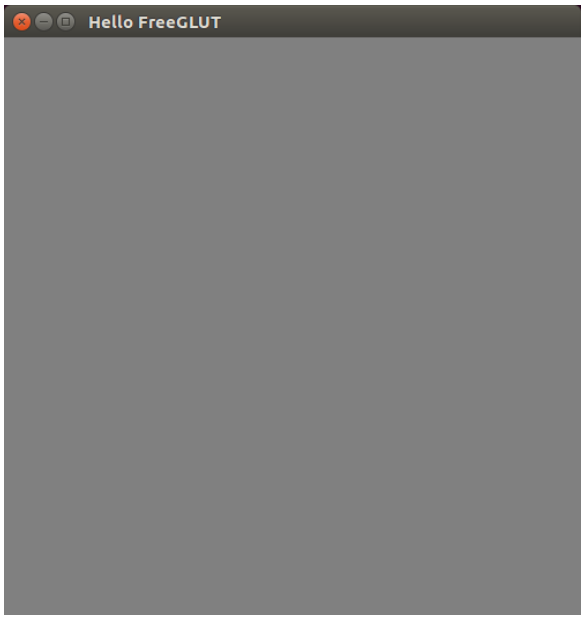

Fig. 2. FreeGLUT를 이용해 작성한 hello_world 프로그램

3.2.1 컴파일 과정 자세히 들어다보기

g++ 컴파일 옵션을 하나하나 살펴보면 다음과 같다.

- g++: C++ 컴파일러로 g++를 이용함
- hello_world.cpp: 컴파일할 소스파일 이름 지정
- -o hello: 컴파일 후 만들어질 실행가능한 파일 이름을 hello로 설정
- -lglut: FreeGLUT 라이브러리 파일을 찾아 링크하도록 함
- -lGL: OpenGL 라이브러리 파일을 찾아 링크하도록 함

만일 소스코드가 수정되어 실행가능한 파일을 새로 만드려면 동일한 g++ 컴파일 과정을 거쳐야 한다. 일반적으로 g++ 컴파일 옵션이 매우 길기 때문에 이를 매번 타이핑하는 작업이 여간 귀찮은게 아니다. 동일한 디렉토리에 다음과 같이 Makefile을 작성해서 컴파일을 간단히 해보자.

```
all:
        g++ hello_world.cpp -o hello -lglut -lGL
```

Makefile을 위와 같이 작성한 후, 터미널에서 make라고 간단히 명령을 내리면 컴파일이 진행된다.

```
make
```

3.3 소스파일 자세히 들어다보기

소스파일 hello_world.cpp 중 아래 항목들을 주목하자.

3.3.1 주목해야할 FreeGLUT 함수들

- glutInitWindowSize(...): FreeGLUT로 띄울 윈도우의 크기 설정. 여기서는 500x500 크기로 설정함.
- glutInitDisplayMode(...): FreeGLUT로 띄울 윈도우의 디스플레이 모드 설정.
  - GLUT_DOUBLE: 컬러버퍼를 2개 두는 더블버퍼 기법을 씀.
  - GLUT_RGBA: 각 픽셀은 R,G,B,A 값을 가지도록 함.
  - GLUT_DEPTH: 프레임버퍼 크기와 동일한 깊이버퍼를 만듬.
- glutCreateWindow(...): 이전에 설정된 값을 이용, 실제로 OpenGL이 돌아갈 수 있는 윈도우를 띄움.
- glutDisplayFunc(...): 화면을 다시 그릴 필요가 있을 때마다 호출할 함수 등록.
- glutMainLoop(...): 무한루프를 돌면서 지속적으로 화면을 그리거나 키보드/마우스로부터 사용자의 입력을 받음.
- glutSwapBuffers(...): 더블버퍼의 위치를 서로 바꿈.

3.3.2 OpenGL 함수들

- glClearColor(...): 컬러버퍼를 새로 그릴때마다 사용할 색 설정
- glClear(...): 설정된 색으로 버퍼를 지운다.
  - GL_COLOR_BUFFER_BIT: 컬러버퍼를 지우도록 설정
  - GL_DEPTH_BUFFER_BIT: 깊이버퍼를 지우도록 설정

4 연습문제

소스파일 hello_world.cpp를 수정해서 프로그램을 다음과 같이 변형해 보자.

1) 윈도우 크기를 800x800으로 바꿔보자.
2) 윈도우 타이틀 바에 'Hello FreeGLUT'가 아닌 자신의 영문이름이 찍히도록 바꿔보자.
3) 윈도우 내부색을 회색에서 흰색으로 바꿔보자.