그래픽스 강의노트 03 - OpenGL 소개

강영민

동명대학교

2021년 2학기

OpenGL

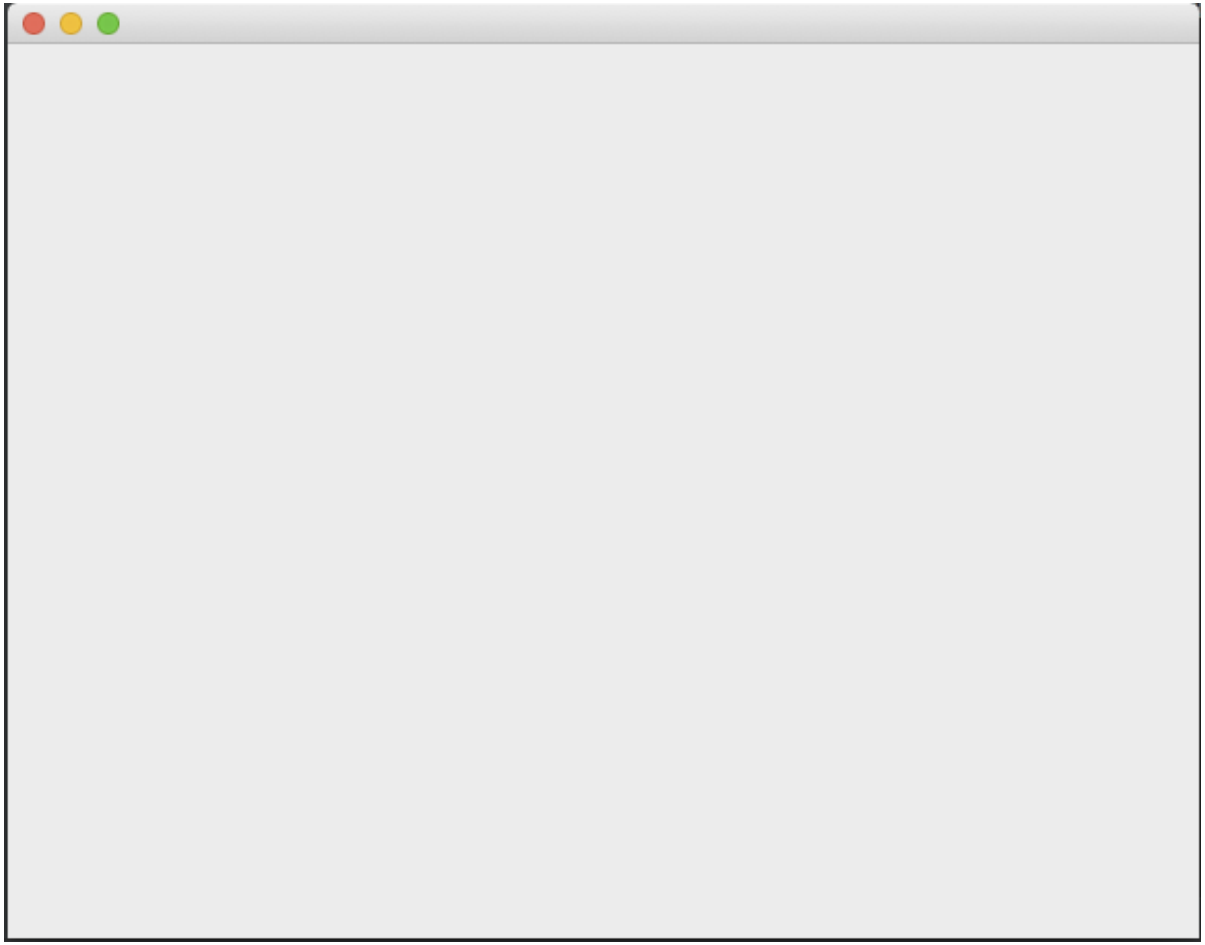
- OpenGL은 특정한 하드웨어나 운영체제에 의존하지 않고 다양한 시스템에 이식(移植)될 수 있는 개방형 라이브러리
- OpenGL을 통한 학습은 실시간 그래픽스에 대한 이해를 돕고, 다양한 시스템에 적용가능한 그래픽스 프로그래밍 기술을 습득하게 함

OpenGL을 사용하기 위한 준비

- Mac OS, Unix/Linux, Windows - 각자의 윈도우 생성 시스템 가짐
- OS 독립적 GL 프로그래밍 - 플랫폼 독립적 윈도우 생성이 필요
- 수업에 사용할 언어
  - Python
  - 필요한 라이브러리 - pyOpenGL
  - $ pip install pyOpenGL
- 수업에 사용할 윈도우 생성 방법
  - Qt
  - $ pip install pyQt5

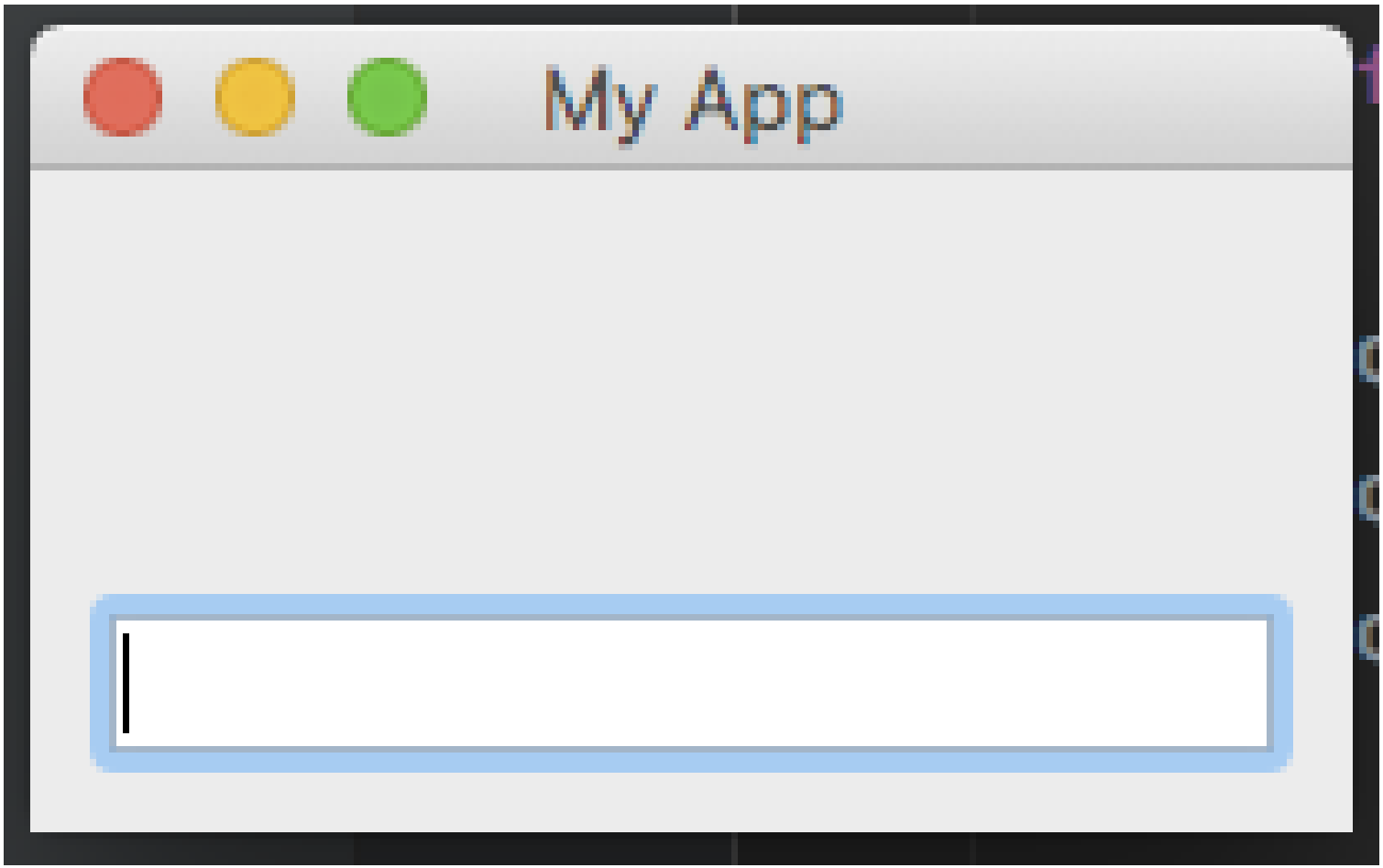
간단한 Qt 윈도우 생성

- pyQt5를 이용한 윈도우 생성 예제
  - 클릭! 소스코드 링크
- 플랫폼 독립적 윈도우 생성이 가능



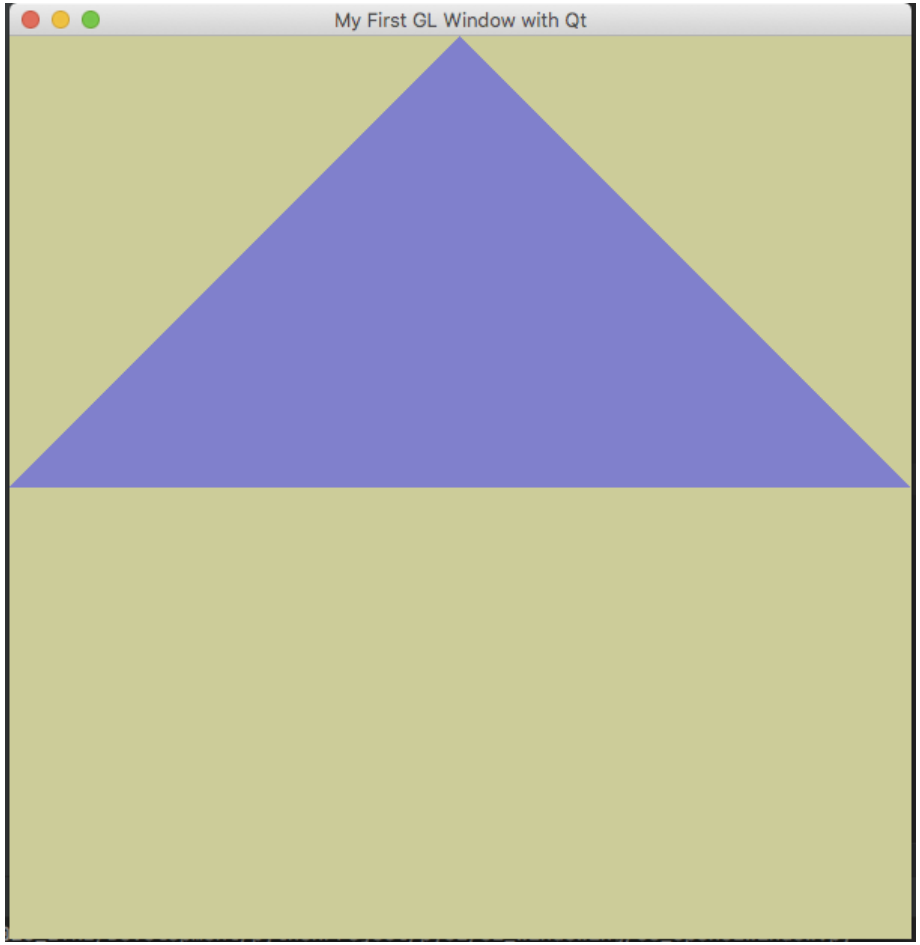
간단한 Qt 윈도우 생성 - Widget 달기

- pyQt5를 이용한 윈도우 생성 + 위젯 예제
  - 클릭! 소스코드 링크
- 다양한 GUI 위젯을 사용할 수 있음 - Qt 학습은 별도로



OpenGL - Qt 윈도우 내의 GLWidget으로 추가

- pyQt5를 이용한 GL 위젯 생성 예제
  - 클릭! 소스코드 링크
- OpenGL 프로그래밍이 가능한 Widget 객체 생성



간단한 OpenGL 프로그램 테스트

```
class MyGLWindow(QOpenGLWidget):

    def __init__(self, parent=None):
        super(MyGLWindow, self).__init__(parent)
    # OpenGL 초기화
    def initializeGL(self):
        glClearColor(0.8, 0.8, 0.6, 1.0)

    # 윈도우 크기 변경시
    def resizeGL(self, width, height):
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()

    # OpenGL 그리기 코드
    def paintGL(self):
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        glMatrixMode(GL_MODELVIEW)
        glLoadIdentity()

        # 그리기 코드

        glFlush()     # 그린 결과를 화면에 송출
```

OpenGL의 특징

- 실시간 그래픽 라이브러리로서 사실상의 (*de facto*) 산업 표준
- 플랫폼에 독립적
  - OpenGL을 이용하여 작성한 그래픽 프로그램은 여러 종류의 다른 운영체제를 가진 시스템에 쉽게 이식
- OpenGL은 상태 기계(state machine)
  - OpenGL의 동작은 현재의 상태에 의해 결정
  - 이 상태는 변경하지 않으면 계속해서 유지
  - OpenGL에는 많은 종류의 상태 변수가 있으며, 이 상태는 한 번 설정하면 다시 변경하지 않는 한 계속해서 설정된 상태를 유지
  - 예: 선의 두께를 결정했다면, 이후에 이 두께를 변경하지 않는 이상 모든 선이 설정된 두께로 그려짐

OpenGL의 관례

- OpenGL API의 명령들은 gl-Command-dimension-type의 꼴
  - glVertex3f는 정점(Vertex)의 위치를 설정하는 OpenGL 명령으로 3차원 데이터이며, 각 차원의 값을 부동소수점(floating point)로 표현한다는 의미
- OpenGL 명령어의 뒤에 붙어 입력 파라미터의 자료형을 표시하는 접미사는 표와 같음

f	32 비트 부동소수점	float	GLfloat
d	64 비트 부동소수점	double	GLdouble
b	8 비트 정수	char	GLbyte
ub	8 비트 부호 없는 정수	unsigned char	GLubyte
i	32 비트 정수	int or long	GLint
ui	32 비트 부호 없는 정수	unsigned int|long	GLuint, GLenum
s	16 비트 정수	short	GLshort

OpenGL 프로그램의 기본 형태

```
# import everything you want
callback_for_display() {
    for (그려질 모든 객체에 대해) {
        변환 설정;
        glBegin(그리기 프리미티브 지정);
            [[정점(vertex) 정보 제공;]]
        glEnd();
    }
    glFlush() 또는 glutSwapBuffers();
}
main(sys.argv) {
    [[윈도우 초기화;]]
    glMatrixMode(GL_PROJECTION);
    [[투영 행렬 설정;]]
    glMatrixMode(GL_MODELVIEW);
    [[카메라의 위치와 방향 잡기;]]
    [[콜백 함수의 등록;]]
    [[메인 루프로 들어가기;]]
}
```

간단한 OpenGL 프로그램 다시 보기

Lines 1–25 / 50

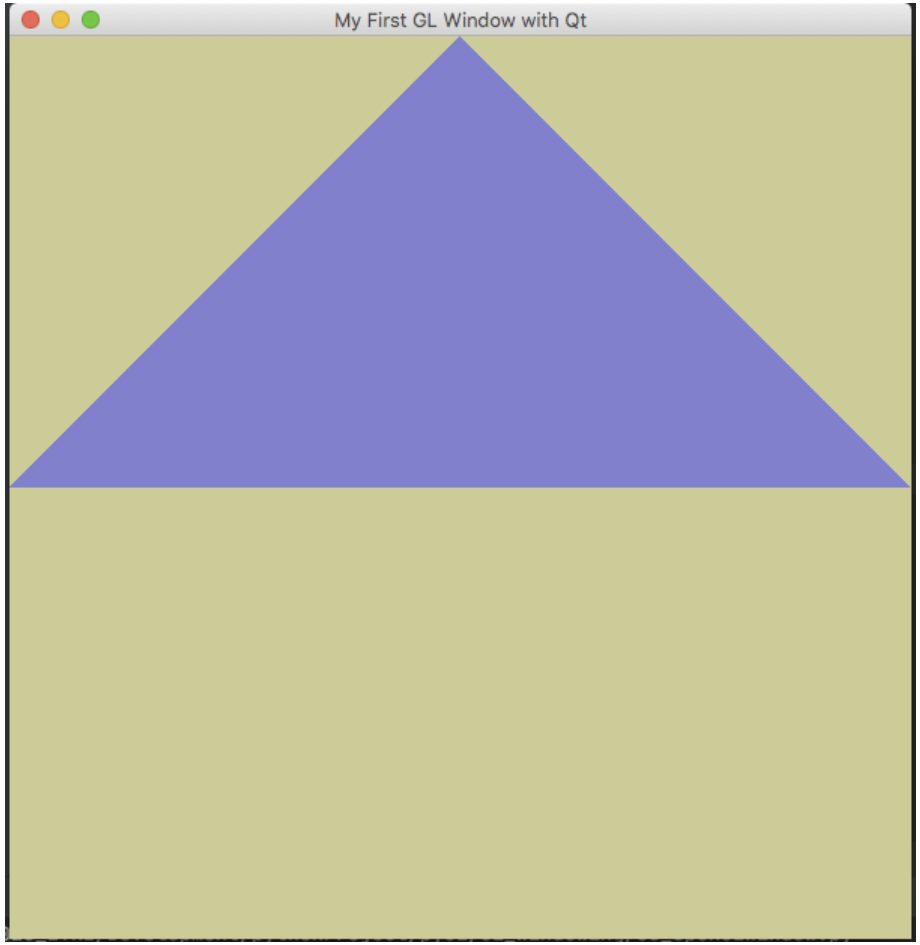
```
from OpenGL.GL import *
from OpenGL.GLU import *

import sys

from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget, QOpenGLWidget


class MyGLWindow(QOpenGLWidget):

    def __init__(self, parent=None):
        super(MyGLWindow, self).__init__(parent)

    def initializeGL(self):
        # OpenGL 그리기를 수행하기 전에 각종 상태값을 초기화
        glClearColor(0.8, 0.8, 0.6, 1.0)

    def resizeGL(self, width, height):
        # 카메라의 투영 특성을 여기서 설정
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()

    def paintGL(self):
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        glMatrixMode(GL_MODELVIEW)
```

간단한 OpenGL 프로그램 다시 보기

Lines 26–50 / 50

```
        glLoadIdentity()

        # 색과 프리미티브를 이용한 객체 그리기
        glColor3f(0.5, 0.5, 0.8)
        glBegin(GL_TRIANGLES)
        glVertex3fv([-1.0, 0.0, 0.0])
        glVertex3fv([ 1.0, 0.0, 0.0])
        glVertex3fv([ 0.0, 1.0, 0.0])
        glEnd()

        # 그려진 프레임버퍼를 화면으로 송출
        glFlush()

def main(argv = []):
    app = QApplication(argv)
    window = MyGLWindow()
    window.setWindowTitle('Example1')
    window.setFixedSize(600, 600)
    window.show()
    sys.exit(app.exec_())

if __name__ == '__main__':
    main(sys.argv)
```

PyQt와 OpenGL 연동



간단한 OpenGL 프로그램 다시 보기

Lines 1–25 / 50

```
from OpenGL.GL import *
from OpenGL.GLU import *

import sys

#from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget,
     QOpenGLWidget
from PyQt5.QtWidgets import QOpenGLWidget, QApplication, QMainWindow, QLabel,
      QLineEdit, QVBoxLayout, QWidget
from PyQt5.QtWidgets import QSlider
from PyQt5.QtCore import *

class MyGLWidget(QOpenGLWidget):

    def __init__(self, parent=None):
        super(MyGLWidget, self).__init__(parent)
        self.r = self.g = self.b = 0.0

    def initializeGL(self):
        # OpenGL 그리기를 수행하기 전에 각종 상태값을 초기화
        glClearColor(0.8, 0.8, 0.6, 1.0)

    def resizeGL(self, width, height):
        # 카메라의 투영 특성을 여기서 설정
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
```

간단한 OpenGL 프로그램 다시 보기

Lines 26–50 / 50

```
def paintGL(self):
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    # 색과 프리미티브를 이용한 객체 그리기
    glColor3f(self.r, self.g, self.b)
    glBegin(GL_TRIANGLES)
    glVertex3fv([-1.0, 0.0, 0.0])
    glVertex3fv([ 1.0, 0.0, 0.0])
    glVertex3fv([ 0.0, 1.0, 0.0])
    glEnd()

    # 그려진 프레임버퍼를 화면으로 송출
    glFlush()

def setR(self, val):
    self.r = val/99
    self.update()
def setG(self, val):
    self.g = val/99
    self.update()
def setB(self, val):
    self.b = val/99
    self.update()
```

간단한 OpenGL 프로그램 다시 보기

Lines 51–75 / 50

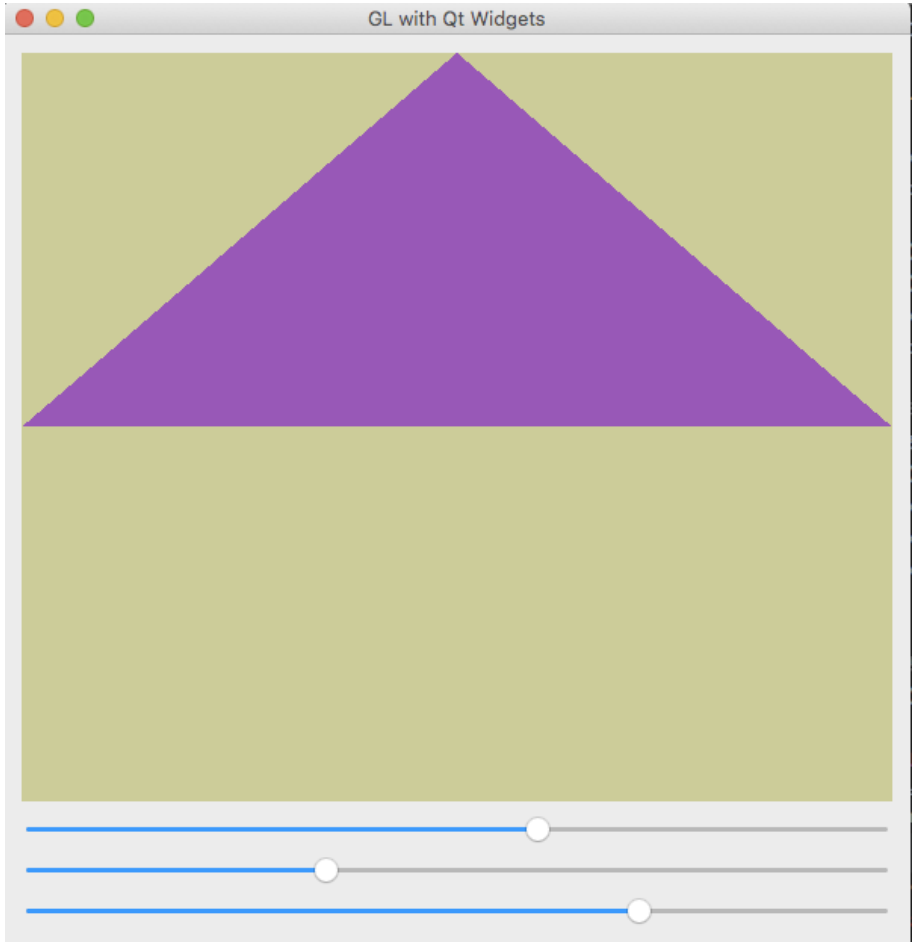
```
class MyWindow(QMainWindow):

    def __init__(self, title = ''):
        QMainWindow.__init__(self)      #call the init for the parent class
        self.setWindowTitle(title)

        ### GUI 설정

        central_widget = QWidget()
        self.setCentralWidget(central_widget)

        gui_layout = QVBoxLayout()     # CentralWidget에 사용될 수직 나열 레이아웃
                                       # 배치될 것들 - GL Window + Control
        central_widget.setLayout(gui_layout)

        self.glWidget = MyGLWidget()  # OpenGL Widget
        gui_layout.addWidget(self.glWidget)

        sliderX = QSlider(Qt.Horizontal)
        sliderX.valueChanged.connect(lambda val: self.glWidget.setR(val))

        sliderY = QSlider(Qt.Horizontal)
        sliderY.valueChanged.connect(lambda val: self.glWidget.setG(val))
```

간단한 OpenGL 프로그램 다시 보기

Lines 76–100 / 50

```
        sliderZ = QSlider(Qt.Horizontal)
        sliderZ.valueChanged.connect(lambda val: self.glWidget.setB(val))

        gui_layout.addWidget(sliderX)
        gui_layout.addWidget(sliderY)
        gui_layout.addWidget(sliderZ)



def main(argv = []):
    app = QApplication(argv)
    window = MyWindow('GL with Qt Widgets')
    window.setFixedSize(600, 600)
    window.show()
    sys.exit(app.exec_())

if __name__ == '__main__':
    main(sys.argv)
```

간단한 OpenGL 프로그램의 실행 결과



프리미티브(primitives) - 1/3

- 그래픽 하드웨어는 프로그래머(programmer)가 지정한 프리미티브(primitive) 설정에 따라 정점의 리스트를 처리
- 프리미티브는 OpenGL이 제공하는 그리기 기본요소
- 입력 정점들을 어떻게 조합할 것인가를 결정
- 프리미티브를 사용하는 방법은 다음과 같다.

```
glBegin(drawing primitive);
        // vertex position, color, normal, etc
        setVertexInfo();
glEnd();
}
```

프리미티브(primitives) - 2/3

- GL_POINTS: 입력된 정점을 하나씩 점으로 가시화
- GL_LINES: 입력된 정점을 두 개씩 묶어 선분으로 표현
- GL_LINE_STRIP: 입력된 정점을 차례대로 연결하여 하나의 폴리라인(polyline)을 구성
- GL_LINE_LOOP: 입력된 정점을 차례로 연결한 뒤에 마지막 점을 시작점으로 연결
- GL_TRIANGLES: 입력된 정점을 세 개씩 묶어 삼각형을 그림
- GL_TRIANGLE_STRIP: 처음 세 개 정점으로 삼각형을 그린 뒤, 정점이 추가될 때마다 삼각형을 직전 두 개 정점과 연결하여 삼각형 추가
- GL_TRIANGLE_FAN: 부채 모양으로 삼각형을 추가해 나감
- GL_QUADS: 정점 네 개씩을 묶어 사각형 그리기
- GL_QUAD_STRIP: 처음 네 개 정점으로 사각형 그리고, 이후 두 개씩 묵어 직전 두 개 정점과 함께 사각형 그리기
- GL_POLYGON: 입력된 모든 정점으로 다각형을 그림