그래픽스 강의노트 07 - 3차원 객체

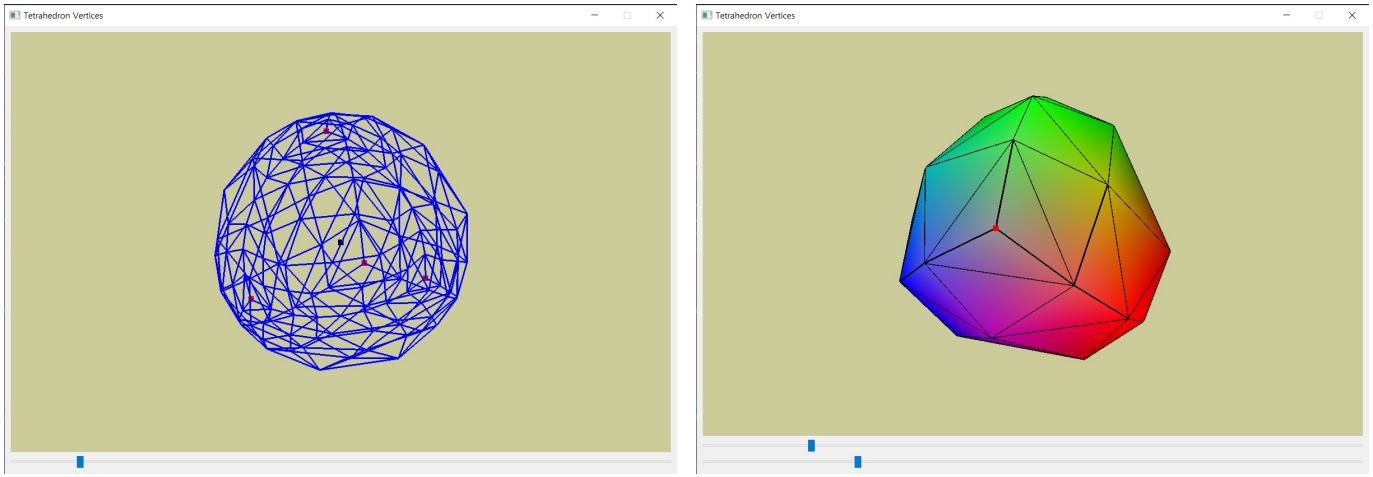
강영민

동명대학교

2021년 2학기

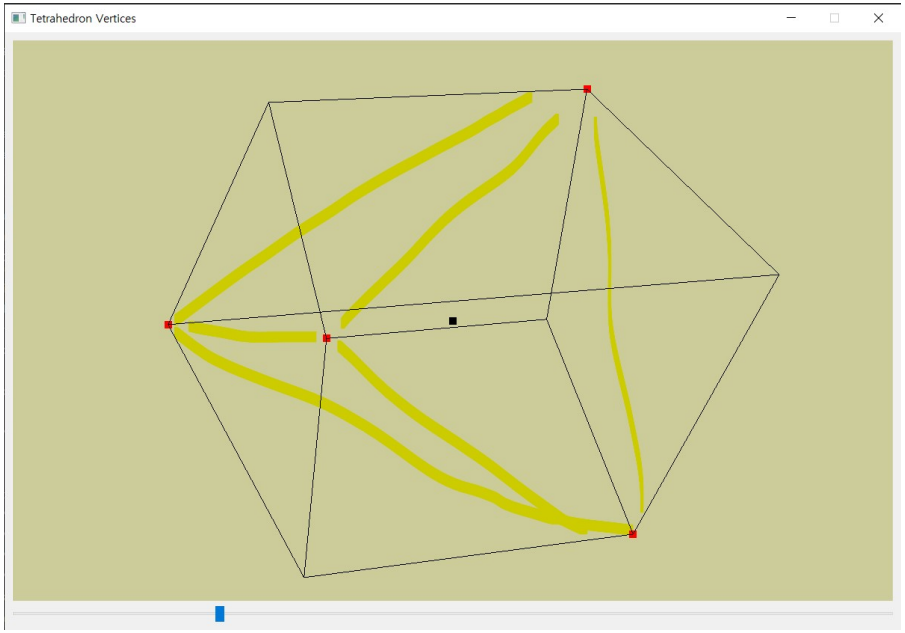
3차원 좌표 - x, y, z

- 2차원이든 3차원이든 동일한 프리미티브
- 와이어프레임(wireframe) - Lines (strip, loop)
- 솔리드(solid) - Triangles (fan, strip), Quads (strip), Polygon



사면체의 정점(vertices)

- 기본적인 입체 도형
- 네 개의 정점
- 네 개의 면



- 프리미티브로 이 점들을 연결하면 사면체 그리기 가능

사면체의 정점 찍기

Lines 1–25 / 125

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

import sys

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

import numpy as np
import math

class MyGLWidget(QOpenGLWidget):

    def __init__(self, parent=None):
        super(MyGLWidget, self).__init__(parent)
        self.center = np.array([0,0,0], dtype=float)
        self.verts = np.array([[-1,1,-1], [1, -1, -1], [1,1,1], [-1,-1,1]],
     dtype=float)
        self.angle = 0.0

    def initializeGL(self):
        # OpenGL 그리기를 수행하기 전에 각종 상태값을 초기화
        glClearColor(0.8, 0.8, 0.6, 1.0)
        glPointSize(10)

    def resizeGL(self, width, height):
```

사면체의 정점 찍기

Lines 26–50 / 125

```
        # 카메라의 투영 특성을 여기서 설정
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        gluPerspective(60, width/height, 0.1, 10)

    def paintGL(self):
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        glMatrixMode(GL_MODELVIEW)
        glLoadIdentity()
        x = 2.2*math.sin(self.angle)
        z = 2.2*math.cos(self.angle)
        gluLookAt(x, 2.0, z, 0,0,0, 0,1,0)

        # 색과 프리미티브를 이용한 객체 그리기
        # 광원의 색
        glBegin(GL_POINTS)
        glColor3f(0,0,0)
        glVertex3fv(self.center)
        for i in range(len(self.verts)):
            glColor3f(1,0,0)
            glVertex3fv(self.verts[i])
        glEnd()

        glBegin(GL_LINE_LOOP)
        glColor3f(0,0,0)
```

사면체의 정점 찍기

Lines 51–75 / 125

```
glVertex3f(-1,1,1)
glVertex3f(-1,-1,1)
glVertex3f(1,-1,1)
glVertex3f(1,1,1)
glEnd()
glBegin(GL_LINE_LOOP)
glVertex3f(-1,1,-1)
glVertex3f(-1,-1,-1)
glVertex3f(1,-1,-1)
glVertex3f(1,1,-1)
glEnd()
glBegin(GL_LINE_LOOP)
glVertex3f(-1,-1,-1)
glVertex3f(-1,-1,1)
glVertex3f(1,-1,1)
glVertex3f(1,-1,-1)
glEnd()
glBegin(GL_LINE_LOOP)
glVertex3f(-1,1,-1)
glVertex3f(-1,1,1)
glVertex3f(1,1,1)
glVertex3f(1,1,-1)
glEnd()
```

사면체의 정점 찍기

Lines 76–100 / 125

```
        # 그려진 프레임버퍼를 화면으로 송출
        glFlush()

    def set_angle(self, val):
        self.angle = 6.28*val/100
        self.update()


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)
```

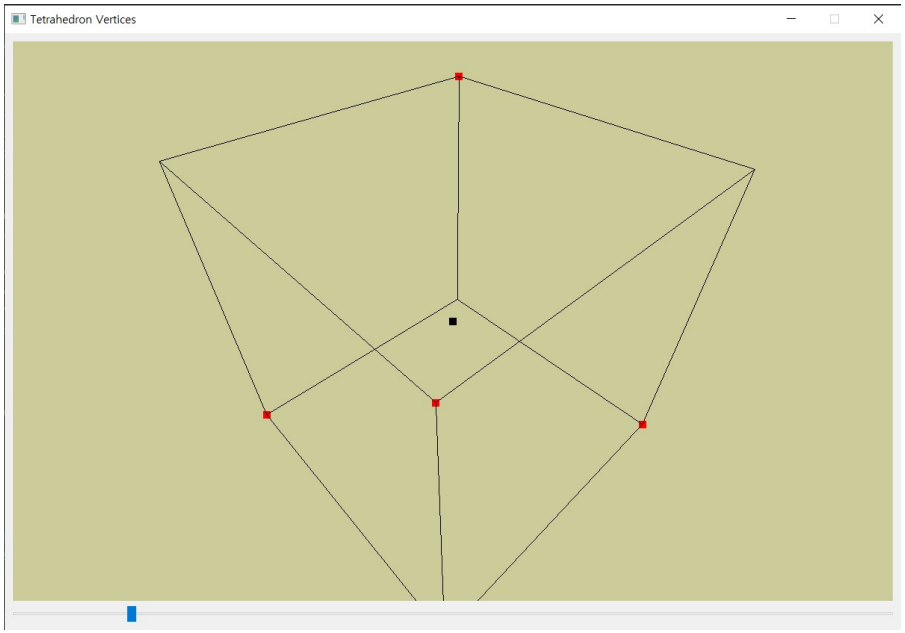
사면체의 정점 찍기

Lines 101–125 / 125

```
        angle_slider = QSlider(Qt.Horizontal)
        gui_layout.addWidget(angle_slider)
        angle_slider.valueChanged.connect(lambda val: self.glWidget.set_angle
    (val))

def main(argv = []):
    app = QApplication(argv)
    window = MyWindow('Tetrahedron Vertices')
    window.setFixedSize(1200, 800)
    window.show()
    sys.exit(app.exec_())

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

정점 찍기 결과

- 슬라이더를 이용하여 카메라를 회전시킴
- 붉은색 점이 사면체의 정점



사면체 정점 연결하기

Lines 1–25 / 125

```
from OpenGL.GL import *
from OpenGL.GLU import *
import sys
from PyQt5.QtWidgets import QOpenGLWidget, QApplication, QMainWindow,
    QVBoxLayout, QWidget, QSlider
from PyQt5.QtCore import *
import numpy as np
import math

def drawTriangle(v0, v1, v2):
    glBegin(GL_LINE_LOOP)
    glVertex3fv(v0)
    glVertex3fv(v1)
    glVertex3fv(v2)
    glEnd()

class MyGLWidget(QOpenGLWidget):

    def __init__(self, parent=None):
        super(MyGLWidget, self).__init__(parent)
        self.center = np.array([0,0,0], dtype=float)
        self.verts = np.array([[-1,1,-1], [1, -1, -1], [1,1,1], [-1,-1,1]],
     dtype=float)
        self.angle = 0.0

    def initializeGL(self):
        # OpenGL 그리기를 수행하기 전에 각종 상태값을 초기화
```

사면체 정점 연결하기

Lines 26–50 / 125

```
        glClearColor(0.8, 0.8, 0.6, 1.0)
        glPointSize(10)

    def resizeGL(self, width, height):
        # 카메라의 투영 특성을 여기서 설정
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        gluPerspective(60, width/height, 0.1, 10)

    def paintGL(self):
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        glMatrixMode(GL_MODELVIEW)
        glLoadIdentity()
        x = 2.2*math.sin(self.angle)
        z = 2.2*math.cos(self.angle)
        gluLookAt(x, 2.0, z, 0,0,0, 0,1,0)

        # 색과 프리미티브를 이용한 객체 그리기
        # 광원의 색
        glBegin(GL_POINTS)
        glColor3f(0,0,0)
        glVertex3fv(self.center)
        for i in range(len(self.verts)):
            glColor3f(1,0,0)
            glVertex3fv(self.verts[i])
```

사면체 정점 연결하기

Lines 51–75 / 125

```
glEnd()

glLineWidth(3)
glColor3f(0,0,1)
drawTriangle(self.verts[0], self.verts[2], self.verts[1])
drawTriangle(self.verts[1], self.verts[2], self.verts[3])
drawTriangle(self.verts[3], self.verts[2], self.verts[0])
drawTriangle(self.verts[0], self.verts[1], self.verts[3])

glLineWidth(1)
glColor3f(0,0,0)
glBegin(GL_LINE_LOOP)
glColor3f(0,0,0)
glVertex3f(-1,1,1)
glVertex3f(-1,-1,1)
glVertex3f(1,-1,1)
glVertex3f(1,1,1)
glEnd()
glBegin(GL_LINE_LOOP)
glVertex3f(-1,1,-1)
glVertex3f(-1,-1,-1)
glVertex3f(1,-1,-1)
glVertex3f(1,1,-1)
glEnd()
glBegin(GL_LINE_LOOP)
```

사면체 정점 연결하기

Lines 76–100 / 125

```
            glVertex3f(-1,-1,-1)
            glVertex3f(-1,-1,1)
            glVertex3f(1,-1,1)
            glVertex3f(1,-1,-1)
            glEnd()
            glBegin(GL_LINE_LOOP)
            glVertex3f(-1,1,-1)
            glVertex3f(-1,1,1)
            glVertex3f(1,1,1)
            glVertex3f(1,1,-1)
            glEnd()

            # 그려진 프레임버퍼를 화면으로 송출
            glFlush()

        def set_angle(self, val):
            self.angle = 6.28*val/100
            self.update()

class MyWindow(QMainWindow):

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

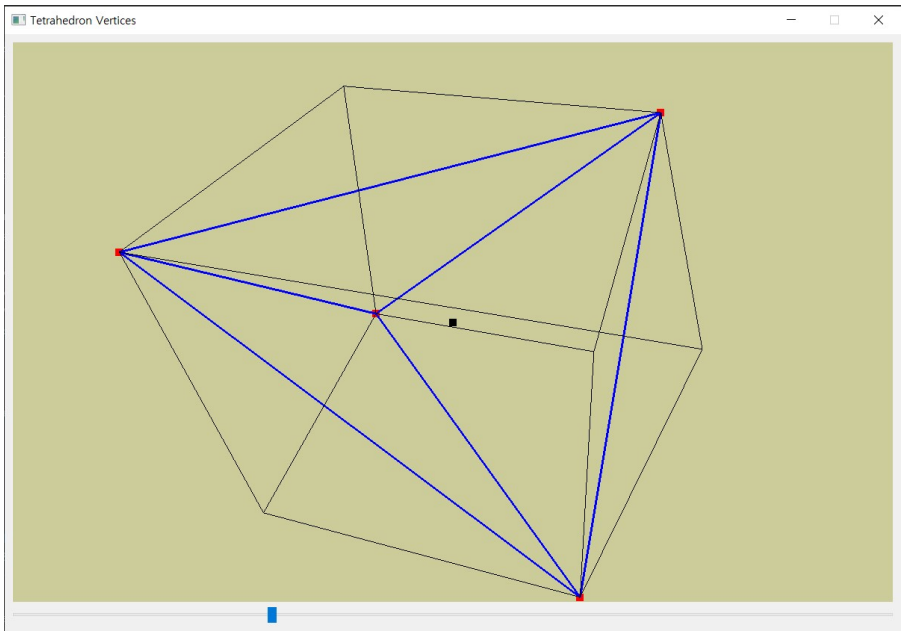
사면체 정점 연결하기

Lines 101–125 / 125

```
        # 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)

        angle_slider = QSlider(Qt.Horizontal)
        gui_layout.addWidget(angle_slider)
        angle_slider.valueChanged.connect(lambda val: self.glWidget.set_angle
    (val))

def main(argv = []):
    app = QApplication(argv)
    window = MyWindow('Tetrahedron Vertices')
    window.setFixedSize(1200, 800)
    window.show()
    sys.exit(app.exec_())

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

정점 연결 결과

- 슬라이더를 이용하여 카메라를 회전시킴
- 붉은색 점이 사면체의 정점
- 각 면이 wireframe으로 그려짐



사면체 면을 분할하기

Lines 1–25 / 75

```
# drawTriangle function

def drawTriangle(v0, v1, v2, subdivision=0):
    if subdivision==0:
        glBegin(GL_LINE_LOOP)
        glVertex3fv(v0)
        glVertex3fv(v1)
        glVertex3fv(v2)
        glEnd()

    elif subdivision > 0:
        v01, v12, v20 = (v0 + v1), (v1+v2), (v2+v0)
        l01 = np.linalg.norm(v01)
        l12 = np.linalg.norm(v12)
        l20 = np.linalg.norm(v20)
        v01 /= l01
        v12 /= l12
        v20 /= l20

        drawTriangle(v0, v01, v20, subdivision = subdivision-1)
        drawTriangle(v01, v1, v12, subdivision = subdivision-1)
        drawTriangle(v20, v12, v2, subdivision = subdivision-1)
        drawTriangle(v01, v12, v20, subdivision = subdivision-1)
```

사면체 면을 분할하기

Lines 26–50 / 75

```
        ...

# OPENGL WIDGET

        def __init__(self, parent=None):
            super(MyGLWidget, self).__init__(parent)
            self.center = np.array([0,0,0], dtype=float)
            c = 1/math.sqrt(3)
            self.verts = np.array([[-c,c,-c], [c, -c, -c], [c,c,c], [-c,-c,c
    ]], dtype=float)
            self.angle = 0.0

        def paintGL(self):
        ...

            # 정점 그리기
            glBegin(GL_POINTS)
            glColor3f(0,0,0)
            glVertex3fv(self.center)
            for i in range(len(self.verts)):
                glColor3f(1,0,0)
                glVertex3fv(self.verts[i])
            glEnd()

            # 면 그리기
            glLineWidth(3)
```

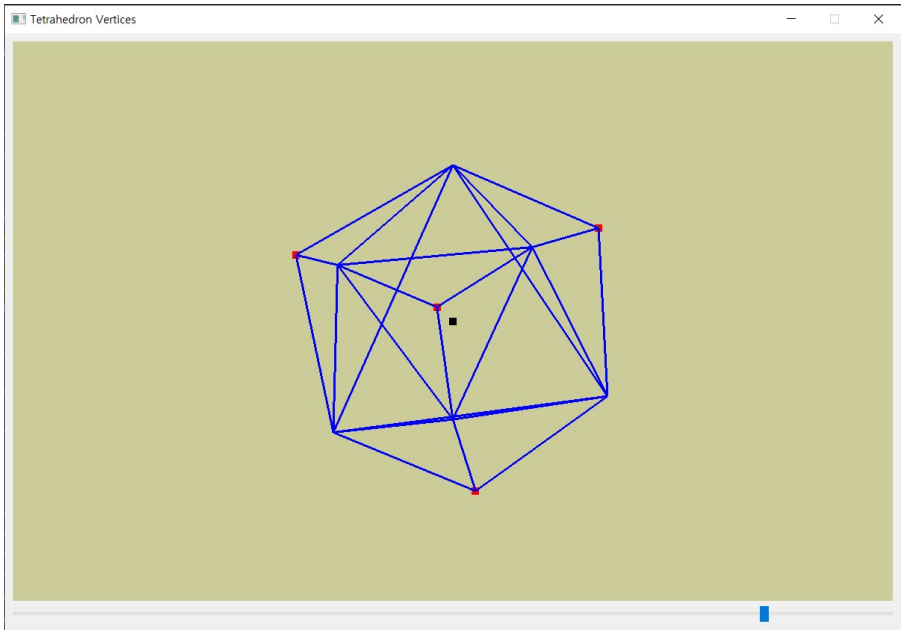
사면체 면을 분할하기

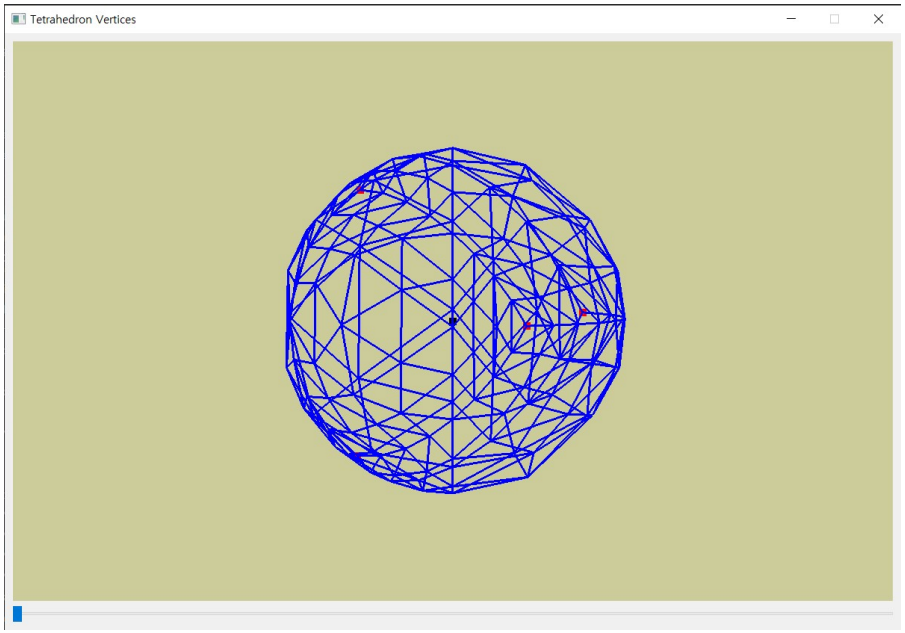
Lines 51–75 / 75

```
glColor3f(0,0,1)

nSub =3
drawTriangle(self.verts[0], self.verts[2], self.verts[1],
             subdivision = nSub)
drawTriangle(self.verts[1], self.verts[2], self.verts[3],
             subdivision = nSub)
drawTriangle(self.verts[3], self.verts[2], self.verts[0],
             subdivision = nSub)
drawTriangle(self.verts[0], self.verts[1], self.verts[3],
             subdivision = nSub)


# 그려진 프레임버퍼를 화면으로 송출
glFlush()
```

사면체 면 분할 결과

- 1회 분할
- 3회 분할





사면체 면을 분할해 좌표를 색상으로 지정하기

Lines 1–25 / 50

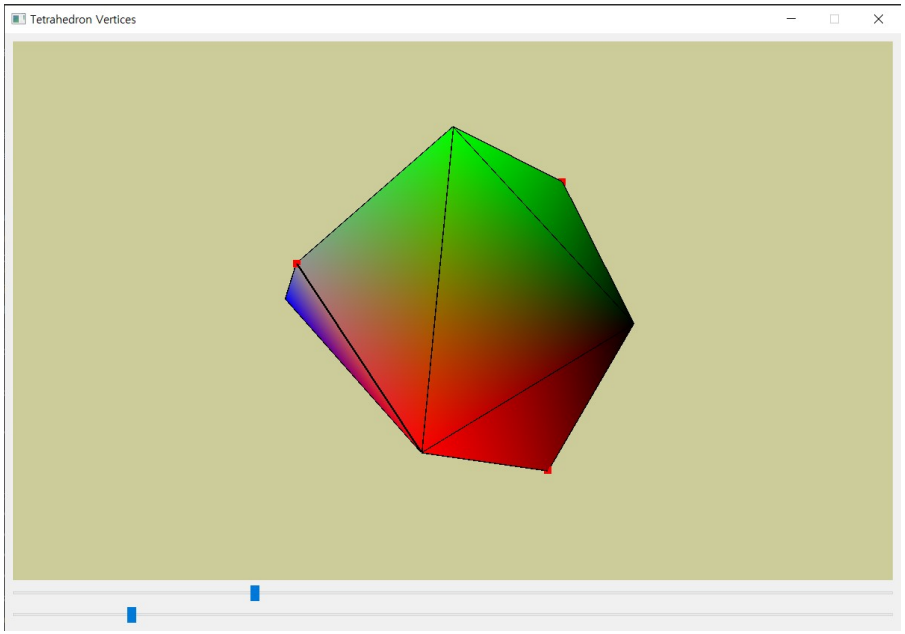
```
def drawTriangle(v0, v1, v2, subdivision=0):

    glLineWidth(3)
    if subdivision==0:
        glBegin(GL_TRIANGLES)
        glColor3fv(v0)
        glVertex3fv(v0)
        glColor3fv(v1)
        glVertex3fv(v1)
        glColor3fv(v2)
        glVertex3fv(v2)
        glEnd()
        glBegin(GL_LINE_LOOP)
        glColor3f(0,0,0)
        glVertex3fv(v0)
        glVertex3fv(v1)
        glVertex3fv(v2)
        glEnd()
    elif subdivision > 0:
        v01, v12, v20 = (v0 + v1), (v1+v2), (v2+v0)
        l01 = np.linalg.norm(v01)
        l12 = np.linalg.norm(v12)
        l20 = np.linalg.norm(v20)
        v01 /= l01
```

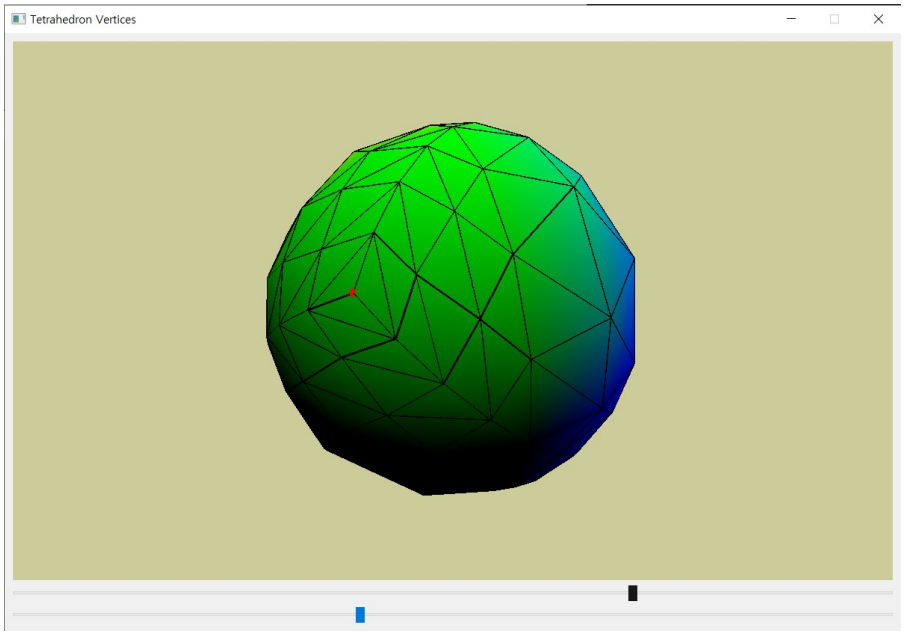
사면체 면을 분할해 좌표를 색상으로 지정하기

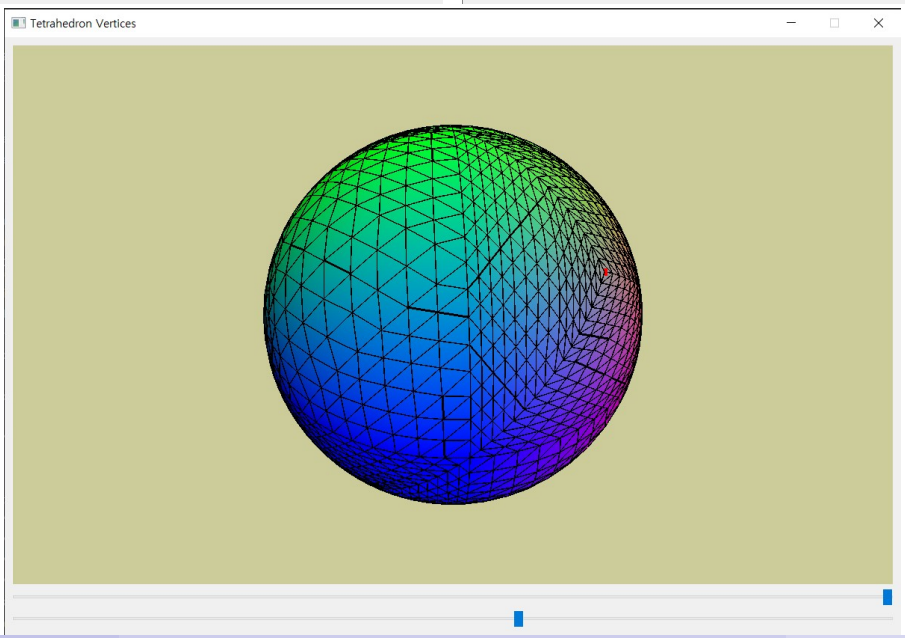
Lines 26–50 / 50

```
v12 /= l12
v20 /= l20
drawTriangle(v0, v01, v20, subdivision = subdivision-1)
drawTriangle(v01, v1, v12, subdivision = subdivision-1)
drawTriangle(v20, v12, v2, subdivision = subdivision-1)
drawTriangle(v01, v12, v20, subdivision = subdivision-1)
```

면 렌더링 결과


Tetrahedron Vertices


Tetrahedron Vertices


Tetrahedron Vertices