그래픽스 강의노트 06 - 카메라와 Z-buffer

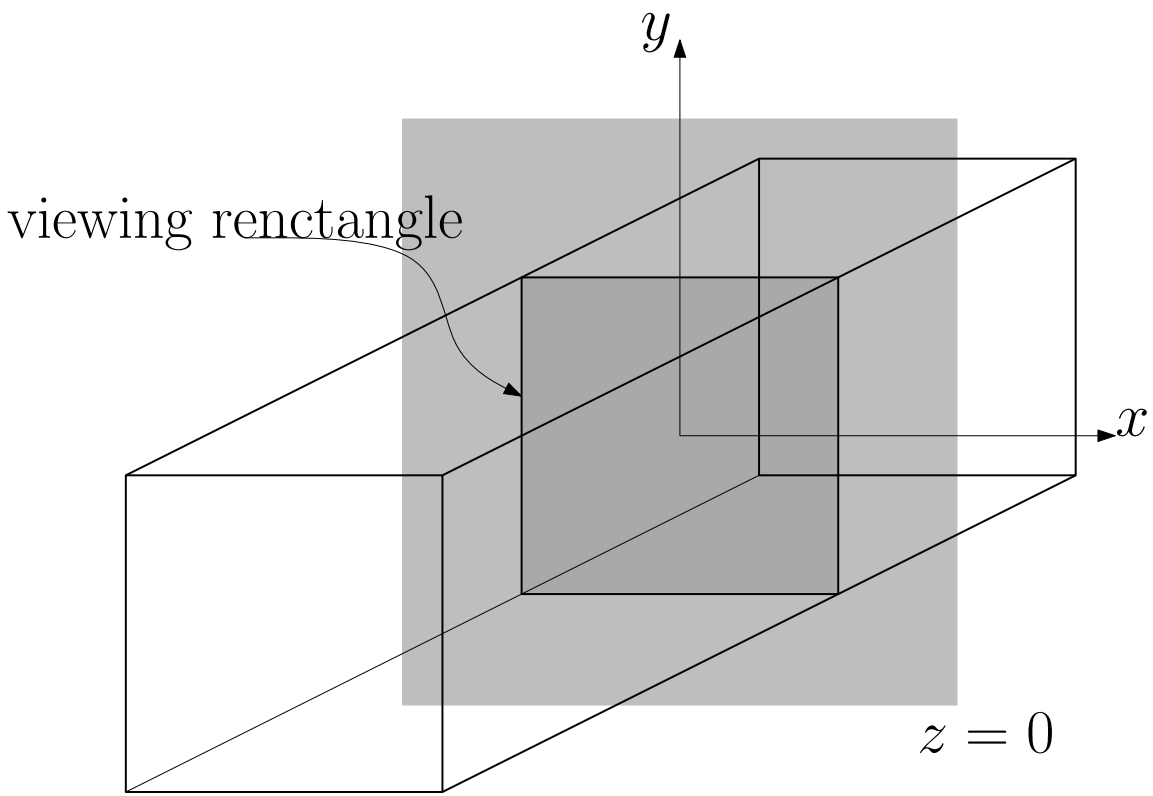
강영민

동명대학교

2021년 2학기

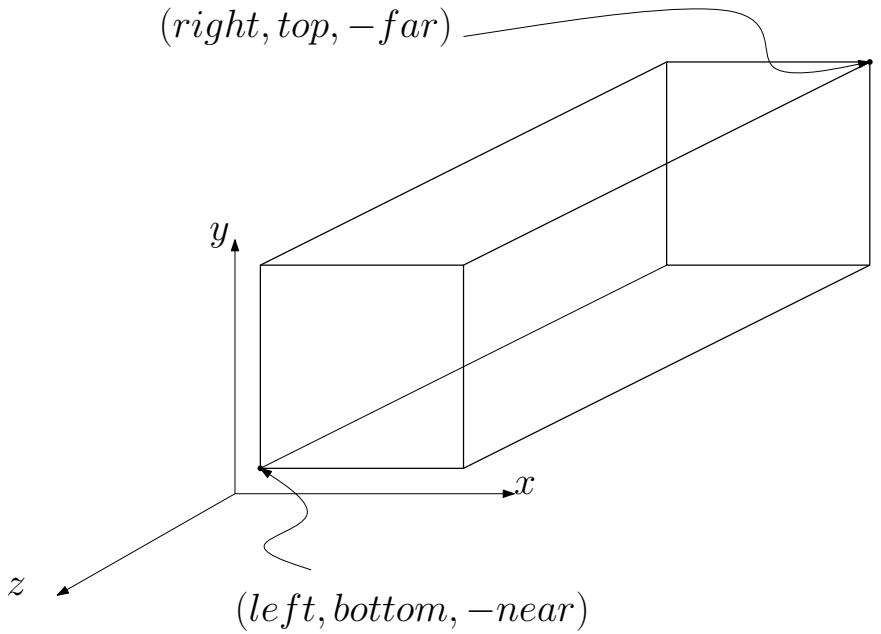
디폴트 카메라

- OpenGL의 디폴트 카메라는 원점 위치
- z축 음의 방향
- 원근 투영을 적용하지 않아 카메라에 잡히는 공간은 상자 형태



직교 투영 (orthographics projection) 설정

- 디폴트 카메라는 직교 투영 카메라
- 상자 모양의 가시화 공간내의 객체를 상자를 절단하는 면에 투영
- 이 상자의 위치와 길이를 변경하는 함수가 glOrtho




```
glOrtho(float left, float right, float bottom, float top, float near, float far);
```

- 결국 OpenGL의 디폴트 카메라는 다음과 같은 설정
  - glOrtho(-1,1, -1,1, -1,1);

원근 투영 (perspective projection) 설정

- 실제 카메라나 우리 눈은 원근이 없는 평행 투영이 불가능
- 멀리 있는 것은 작게 보이고 가까이 있는 것은 크게 보이는 이유는 원근투영
- 이러한 원근 투영을 설정하는 방법은 glFrustum과 gluPerspective
- glFrustum에 대한 이해는 뒤로 미루고 우선 직관적인 gluPerspective 함수를 먼저 이해

```
gluPerspective(float fovy, float Aspect, float near, float far);
```

- fovy는 y축 방향으로의 시야각을 도(degree)로 나타낸 것
- Aspect는 가시화 볼륨의 종횡비(aspect ratio)
- near는 카메라에서 상이 맺히는 가까운 평면까지의 거리
- far는 가시화 공간을 결정하는 평면 중 카메라에서 가장 먼 쪽 평면과 카메라 사이의 거리
- 이 함수는 그리기 동작이 일어날 때마다 불리는 것이 아니라 초기에 한 번, 혹은 렌즈를 바꿀 필요가 있을 때에만 불린다.

행렬 모드(matrix mode)

- OpenGL은 세 종류의 행렬: 텍스처 행렬, 모델뷰 행렬, 투영행렬
- 모델뷰 행렬은 공간 내에서 가상 객체의 좌표를 변경
- 투영행렬은 가상 객체를 투영면에 옮겨 놓음

```
glMatrixMode(GL_PROJECTION)
glLoadIdentity()
gluPerspective(60, 1.0, 0.1, 100.0)

glMatrixMode(GL_MODELVIEW)
glLoadIdentity()
gluLookAt(1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0)

[[draw something]]
```

- 여기서는 gluPerspective와 gluLookAt이 어떠한 행렬 모드을 변경하는가 하는 것이 관심
- gluPerspective 함수는 투영의 특성을 변경하는 것이므로 투영행렬 모드
- gluLookAt은 카메라의 위치를 옮기는 것이고, 이는 바꾸어 말해 물체의 위치를 카메라 기준에서 옮기는 것이므로 모델뷰 행렬을 변경

카메라 위치 변경

- 카메라를 원하는 곳으로 옮겨주는 함수: gluLookAt

```
gluLookAt(float eye_x, float eye_y, float eye_z,
          float at_x, float at_y, float at_z,
          float up_x, float up_y, float up_z)
```

- 카메라의 위치를 옮겨 놓는 역학
- 카메라 이동의 반대로 물체를 옮겨 놓는 것
- 카메라의 위치는 매 프레임마다 변경될 수 있으므로 이 함수는 그리기 함수 내에서 매번 불리는 것이 일반적

카메라를 설정하는 기본적인 프로그램

Lines 1–25 / 50

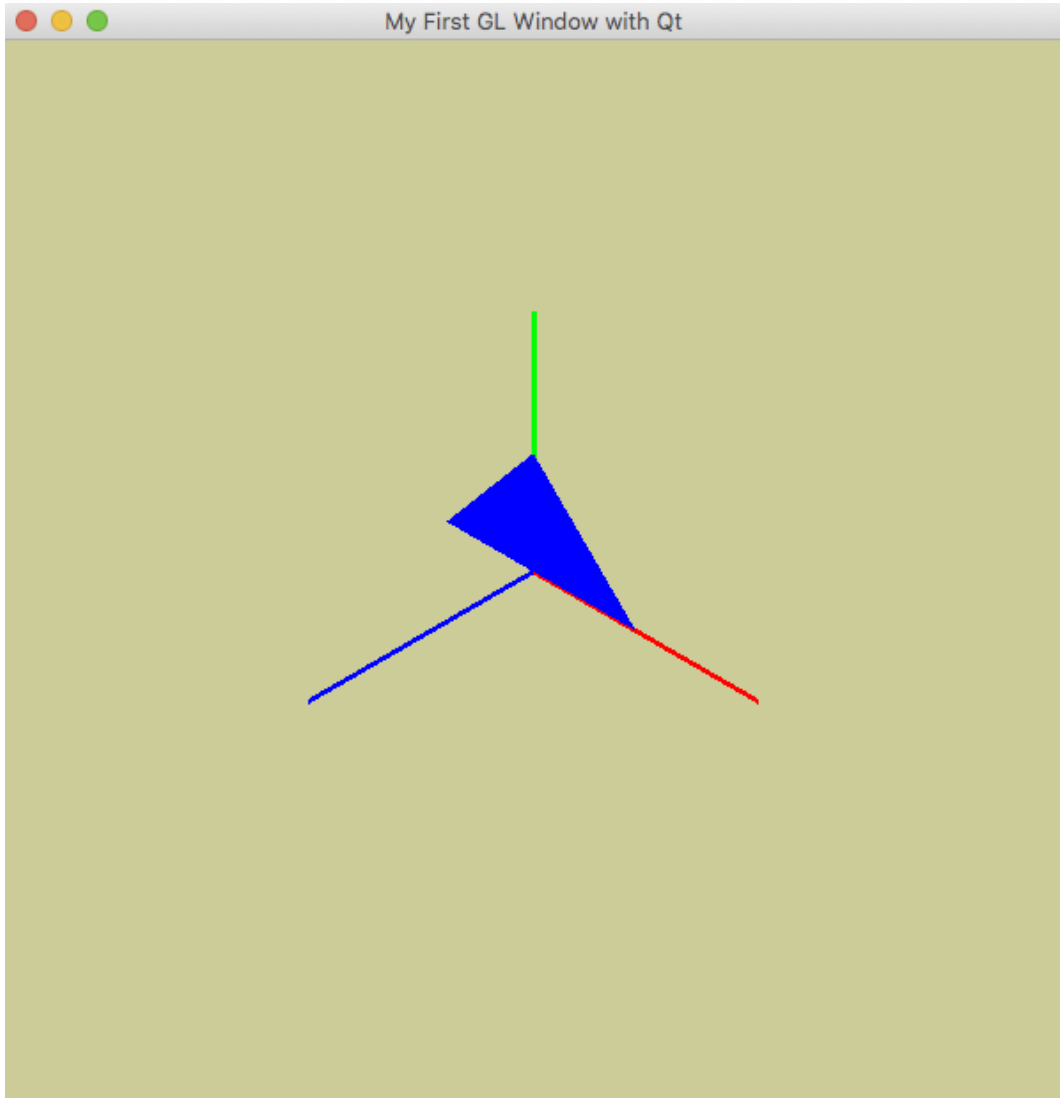
```
from OpenGL.GL import *
from OpenGL.GLU import *
import sys
from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget, QOpenGLWidget

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)
        glLineWidth(3)

    def resizeGL(self, width, height):
        # 카메라의 투영 특성을 여기서 설정
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        asp = width/height
        gluPerspective(60, asp, 0.01, 100)

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

카메라를 설정하는 기본적인 프로그램

Lines 26–50 / 50

```
        gluLookAt(2,2,2, 0,0,0, 0,1,0)

        # 그려진 프레임버퍼를 화면으로 송출
        glBegin(GL_LINES)
        glColor3f(1, 0, 0); glVertex3f(0, 0, 0); glVertex3f(1, 0, 0)
        glColor3f(0, 1, 0); glVertex3f(0, 0, 0); glVertex3f(0, 1, 0)
        glColor3f(0, 0, 1); glVertex3f(0, 0, 0); glVertex3f(0, 0, 1)
        glEnd()

        glBegin(GL_TRIANGLES)
        glVertex3f(-0.5,0,0); glVertex3f(0.5,0,0); glVertex3f(0,0.5,0)
        glEnd()

        glFlush()

def main(argv = []):
    app = QApplication(argv)
    window = MyGLWindow()
    window.setWindowTitle('My First GL Window with Qt')
    window.setFixedSize(600, 600)
    window.show()
    sys.exit(app.exec_())

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

카메라 조작 결과



카메라 애니메이션

Lines 1–25 / 75

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

import sys

from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget, QOpenGLWidget
from PyQt5.QtCore import QBasicTimer
import math

def drawAxes():
    glBegin(GL_LINES)
    glColor3f(1, 0, 0)
    glVertex3f(0, 0, 0); glVertex3f(1, 0, 0)
    glColor3f(0, 1, 0)
    glVertex3f(0, 0, 0); glVertex3f(0, 1, 0)
    glColor3f(0, 0, 1)
    glVertex3f(0, 0, 0); glVertex3f(0, 0, 1)
    glEnd()

class MyGLWindow(QOpenGLWidget):

    def __init__(self, parent=None):
        super(MyGLWindow, self).__init__(parent)
        self._timer = QBasicTimer()                 # 타이머 생성
        self._timer.start(int(1000 / 60), self)     # 초당 60 프레임
```

카메라 애니메이션

Lines 26–50 / 75

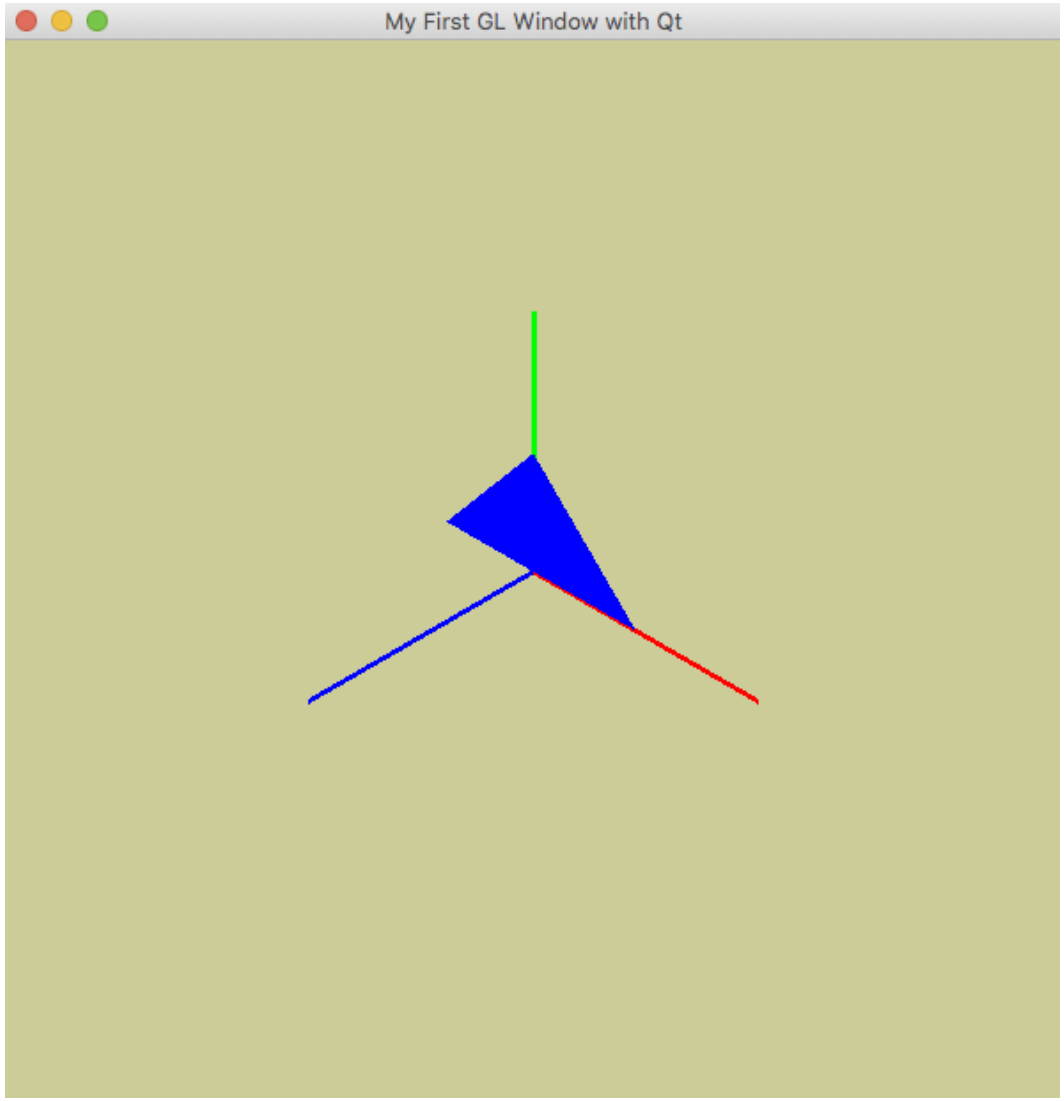
```
        self.angle = 0.0

    def initializeGL(self):
        # OpenGL 그리기를 수행하기 전에 각종 상태값을 초기화
        glClearColor(0.8, 0.8, 0.6, 1.0)
        glLineWidth(3)

    def resizeGL(self, width, height):
        # 카메라의 투영 특성을 여기서 설정
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        asp = width/height
        gluPerspective(60, asp, 0.01, 100)

    def paintGL(self):
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

        glMatrixMode(GL_MODELVIEW)
        glLoadIdentity()
        gluLookAt(2.0 * math.cos(self.angle), 1.0, 2.0 * math.sin(self.angle)
    ,
                  0,0,0, 0,1,0)

        glBegin(GL_TRIANGLES)
        glVertex3f(-0.5, 0, 0)
        glVertex3f( 0.5, 0, 0)
```

카메라 애니메이션

Lines 51–75 / 75

```
            glVertex3f( 0, 0.5, 0)
            glEnd()
            drawAxes()

            # 그려진 프레임버퍼를 화면으로 송출
            glFlush()

        def timerEvent(self, QTimerEvent):
            self.angle += 0.01
            self.update()

def main(argv = []):
    app = QApplication(argv)
    window = MyGLWindow()
    window.setWindowTitle('My First GL Window with Qt')
    window.setFixedSize(600, 600)
    window.show()
    sys.exit(app.exec_())

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

카메라 애니메이션 결과



(깊이버퍼 없이) 두 삼각형 그리기

Lines 1–25 / 100

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

import sys

from PyQt5.QtWidgets import QApplication, QMainWindow, QWidget, QOpenGLWidget
from PyQt5.QtCore import QBasicTimer
import math

def drawAxes():
    glBegin(GL_LINES)
    glColor3f(1, 0, 0)
    glVertex3f(0, 0, 0); glVertex3f(1, 0, 0)
    glColor3f(0, 1, 0)
    glVertex3f(0, 0, 0); glVertex3f(0, 1, 0)
    glColor3f(0, 0, 1)
    glVertex3f(0, 0, 0); glVertex3f(0, 0, 1)
    glEnd()

class MyGLWindow(QOpenGLWidget):

    def __init__(self, parent=None):
        super(MyGLWindow, self).__init__(parent)
        self._timer = QBasicTimer()                 # 타이머 생성
        self._timer.start(int(1000 / 60), self)     # 초당 60 프레임
```

(깊이버퍼 없이) 두 삼각형 그리기

Lines 26–50 / 100

```
        self.angle = 0.0

    def initializeGL(self):
        # OpenGL 그리기를 수행하기 전에 각종 상태값을 초기화
        glClearColor(0.8, 0.8, 0.6, 1.0)
        glLineWidth(3)
        glEnable(GL_DEPTH_TEST)

    def resizeGL(self, width, height):
        # 카메라의 투영 특성을 여기서 설정
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        asp = width/height
        gluPerspective(60, asp, 0.01, 100)

    def paintGL(self):
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

        glMatrixMode(GL_MODELVIEW)
        glLoadIdentity()
        gluLookAt(2.0 * math.cos(self.angle), 1.0, 2.0 * math.sin(self.angle)
    ,
                  0,0,0, 0,1,0)

        glBegin(GL_TRIANGLES)
        glColor3f(1, 1, 0)
```

(깊이버퍼 없이) 두 삼각형 그리기

Lines 51–75 / 100

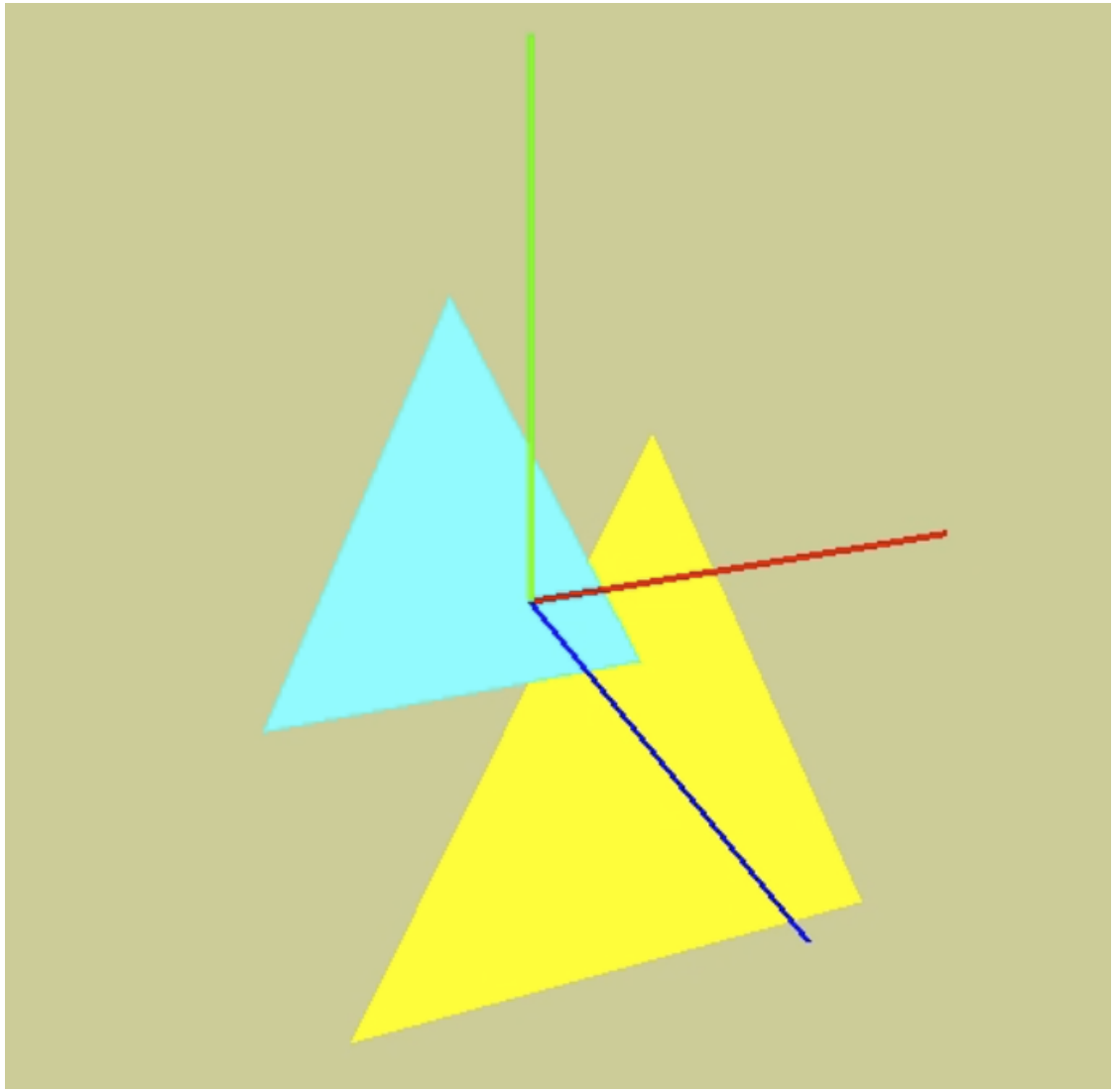
```
            glVertex3f(.0, .5, .5)
            glVertex3f(-.5, -.5, .5)
            glVertex3f(.5, -.5, .5)
            glColor3f(0, 1, 1)
            glVertex3f(.0, .5, -.5)
            glVertex3f(-.5, -.5, -.5)
            glVertex3f(.5, -.5, -.5)
            glEnd()
            glFlush()

            drawAxes()

            # 그려진 프레임버퍼를 화면으로 송출
            glFlush()

        def timerEvent(self, QTimerEvent):
            self.angle += 0.01
            self.update()

    def main(argv = []):
        app = QApplication(argv)
        window = MyGLWindow()
        window.setWindowTitle('My First GL Window with Qt')
        window.setFixedSize(600, 600)
        window.show()
```

(깊이버퍼 없이) 두 삼각형 그리기

Lines 76–100 / 100

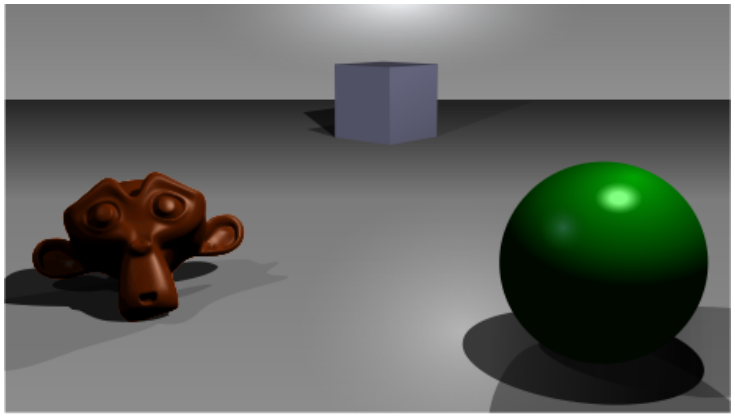
```
    sys.exit(app.exec_())

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

(깊이버퍼 없이) 두 삼격형 그리기

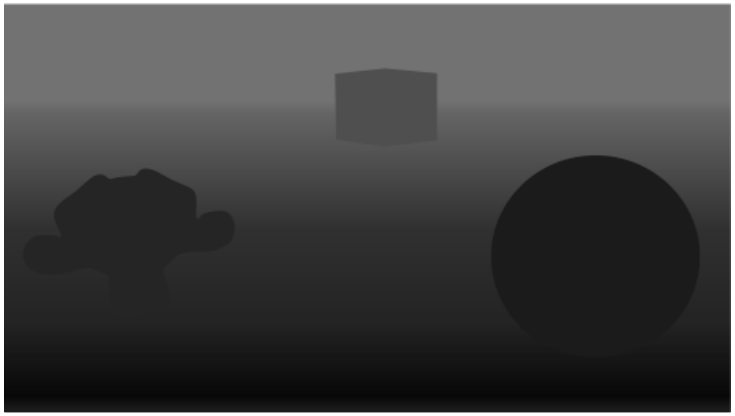


깊이 버퍼 (depth buffer)

- 깊이 버퍼 혹은 z-버퍼
  - 색상 버퍼(color buffer)는 그려지는 영상의 각 화소별 색을 기록
  - 우리가 최종적으로 보게되는 것은 이 색상 버퍼의 내용
  - 색상 버퍼를 구성할 때 물체의 가려짐을 고려하여 가려진 물체를 그리지 않도록 하는 버퍼가 깊이 버퍼
  - 색상이 아니라 카메라에서 그려지는 픽셀까지의 거리를 저장하는 버퍼



A simple three-dimensional scene



Z-buffer representation

깊이 버퍼의 사용

- 기본적으로 “깊이 테스트” 작업을 수행하지 않는 것이 디폴트
- 깊이 테스트를 수행하기 위해서는 다음과 같이 상태를 변경
  - glEnable(GL_DEPTH_TEST);
- 또한 매번 그리기가 이뤄지면 그려진 내용에 따라 깊이 버퍼의 내용이 더러워져 있는 상태
- 새로운 그리기를 하기 전에 색상 버퍼를 깨끗하게 지우듯이 깊이 버퍼의 내용도 깨끗하게 지우는 다음과 같은 코드가 필요
  - glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

깊이 버퍼와 이중 버퍼 사용 결과