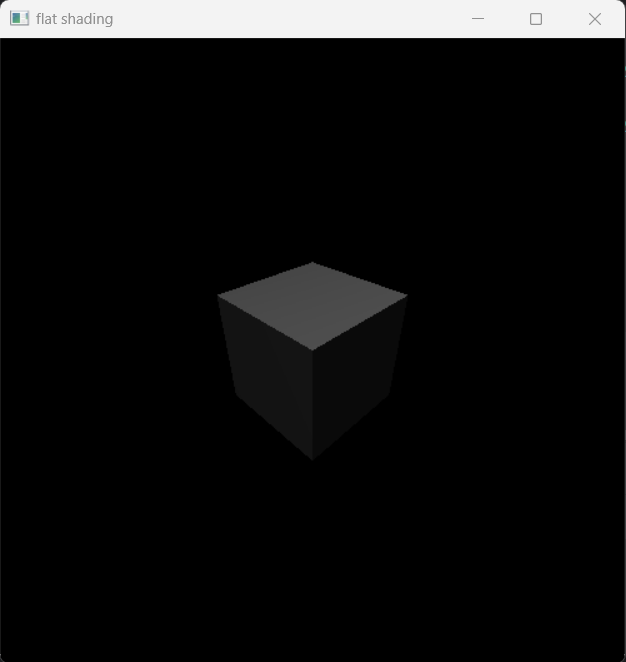
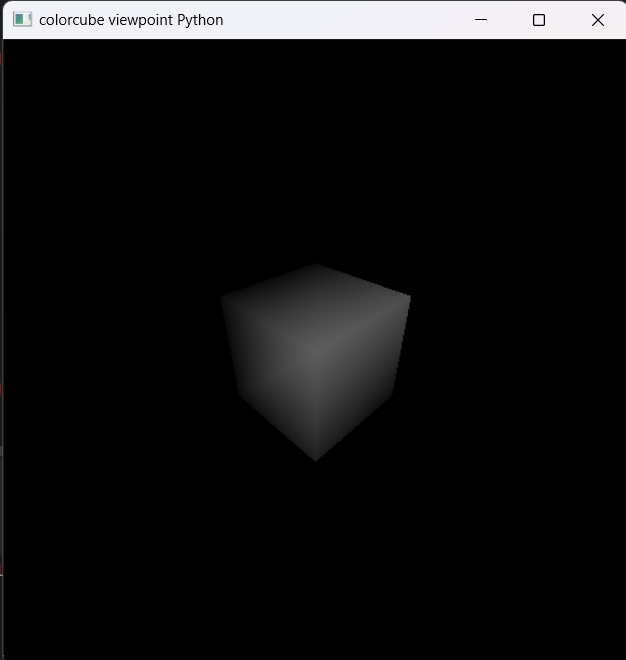
**컴퓨터그래픽스 Lab08 보고서**

|  |  |  |
| --- | --- | --- |
| **학번** | **이름** | **분반** |
| 2312282 | 임다희 | 003 |

**[과제]** flat shading, smooth shading 구현

**결과**

-flat shading -smooth shading



**코드**

1) flat shading (lab08\_2312282\_임다희\_01.py)

from OpenGL.GL import \*

from OpenGL.GLUT import \*

from OpenGL.GLU import \*

import numpy as np

vertices = ((-1.0, -1.0, -1.0), (1.0, -1.0, -1.0),

(1.0, 1.0, -1.0), (-1.0, 1.0, -1.0), (-1.0, -1.0, 1.0),

(1.0, -1.0, 1.0), (1.0, 1.0, 1.0), (-1.0, 1.0, 1.0))

# 정육면체를 이루는 8개 점의 좌표.

myview = 3

xRot = 0.0

yRot = 0.0

def flatNormal(v1, v2, v3):

# 노멀벡터를 구하는 메소드. x,y,z좌표를 가지는 점 3개를 인수로 받는다.

# 각 면의 노멀벡터(법선벡터)를 계산하기 위한 함수.

v1 = np.array(v1)

v2 = np.array(v2)

v3 = np.array(v3)

# 각 점을 3차원 벡터 v1~v3의 형태로 표현한다.

cross = np.cross(v3 - v2, v1 - v2) 정육면체의 한 면 위에 위치한 두 벡터 v3-v2, v1-v2의 외적을 구한다.

length = np.linalg.norm(cross) # np.cross를 통해 구한 외적 벡터의 크기를 구한다.

normal = cross / length# 외적 벡터의 각 성분값을 외적벡터의 크기로 나눈다.

return normal # normalize된 외적 벡터의 값을 리턴한다.

def polygonNormal(a, b, c, d):

normalvector = flatNormal(vertices[a], vertices[b], vertices[c])

# 한 면을 이루는 점 중 3개의 점으로 그 면의 노멀벡터를 구해 flat shading 진행.

# 면의 모든 점에 대해 동일한 노멀벡터가 적용된다.

glBegin(GL\_POLYGON)

glNormal3fv(normalvector)

glVertex3fv(vertices[a])

glNormal3fv(normalvector)

glVertex3fv(vertices[b])

glNormal3fv(normalvector)

glVertex3fv(vertices[c])

glNormal3fv(normalvector)

glVertex3fv(vertices[d])

glEnd()

def cubeFlat():

polygonNormal(0, 3, 2, 1)

polygonNormal(0, 1, 5, 4)

polygonNormal(5, 4, 7, 6)

polygonNormal(5, 1, 2, 6)

polygonNormal(7, 6, 2, 3)

polygonNormal(4, 7, 3, 0)

# flat shading 설정이 적용된 폴리곤을 만든다.

def MyDisplay():

global myview

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

glLoadIdentity()

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, (0.2,0.2,0.2, 1.0))

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, (0.7, 0.7, 0.7, 1.0))

glLightfv(GL\_LIGHT0, GL\_POSITION, (8.0, 0.0, 8.0, 1.0))

gluLookAt(3.0, 3.0, 3.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0)

cubeFlat()

glutSwapBuffers()

def myReshape(w, h):

glViewport(0, 0, w, h)

glMatrixMode(GL\_PROJECTION)

glLoadIdentity()

# glFrustum (left, right, bottom, top, near distance, far distance)

if w <= h:

glFrustum(-2.0, 2.0, -2.0 \* float(h) / float(w), 2.0 \* float(h) / float(w), 2.0, 20.0)

else:

glFrustum(-2.0, 2.0, -2.0 \* float(w) / float(h), 2.0 \* float(w) / float(h), 2.0, 20.0)

glMatrixMode(GL\_MODELVIEW)

def main():

glutInit(sys.argv)

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH)

glutInitWindowSize(500, 500)

glutCreateWindow('flat shading')

glEnable(GL\_DEPTH\_TEST)

glEnable(GL\_LIGHTING)

glEnable(GL\_LIGHT0)

glEnable(GL\_NORMALIZE)

glutReshapeFunc(myReshape)

glutDisplayFunc(MyDisplay)

glutMainLoop()

if \_\_name\_\_ == "\_\_main\_\_":

main()

2) smooth shading (lab08\_2312282\_임다희\_02.py)

from OpenGL.GL import \*

from OpenGL.GLUT import \*

from OpenGL.GLU import \*

import numpy as np

vertices = ((-1.0, -1.0, -1.0), (1.0, -1.0, -1.0),

(1.0, 1.0, -1.0), (-1.0, 1.0, -1.0), (-1.0, -1.0, 1.0),

(1.0, -1.0, 1.0), (1.0, 1.0, 1.0), (-1.0, 1.0, 1.0))

# 정육면체를 이루는 8개 점의 좌표.

normals = [

tuple(np.array(v) / np.linalg.norm(v))

for v in vertices

]

# smooth shading을 위해 각 점에서의 노멀벡터를 구한다.

# 점의 노멀벡터=점에 인접한 면들의 노멀벡터를 더하여 정규화한 값 이다.

#정점을 포함하는 면의 법선벡터를 더하면 정점의 좌표와 동일한 결과가 나오므로,

# 정점의 좌표를 바로 정규화함으로서 점의 법선벡터를 얻을 수 있다.

# vertice 내의 8개 점의 좌표에서 이와 같이 실행하여 점들의 normal벡터를 저장하는 normals 배열을 얻는다.

myview = 3

xRot = 0.0

yRot = 0.0

def polygonSmooth(a, b, c, d):

glBegin(GL\_POLYGON)

glNormal3fv(normals[a])

glVertex3fv(vertices[a])

glNormal3fv(normals[b])

glVertex3fv(vertices[b])

glNormal3fv(normals[c])

glVertex3fv(vertices[c])

glNormal3fv(normals[d])

glVertex3fv(vertices[d])

# 각 점마다 앞에서 계산한 고유한 법선벡터를 적용한다.

glEnd()

def cubeSmooth():

polygonSmooth(0, 3, 2, 1)

polygonSmooth(0, 1, 5, 4)

polygonSmooth(5, 4, 7, 6)

polygonSmooth(5, 1, 2, 6)

polygonSmooth(7, 6, 2, 3)

polygonSmooth(4, 7, 3, 0)

# smooth shading 설정이 적용된 폴리곤을 만든다.

def MyDisplay():

global myview

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

glLoadIdentity()

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, (0.2,0.2,0.2, 1.0))

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, (0.5, 0.5, 0.5, 1.0))

glLightfv(GL\_LIGHT0, GL\_POSITION, (8.0, 0.0, 8.0, 1.0))

if myview == 0:

gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0)

elif myview == 1:

gluLookAt(0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -1.0)

elif myview == 2:

gluLookAt(5.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0)

elif myview == 3:

gluLookAt(3.0, 3.0, 3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0)

glColor3f(1.0, 1.0, 1.0)

glRotate(90, 0, 0, 1)

cubeSmooth()

glutSwapBuffers()

def myReshape(w, h):

glViewport(0, 0, w, h)

glMatrixMode(GL\_PROJECTION)

glLoadIdentity()

# glFrustum (left, right, bottom, top, near distance, far distance)

if w <= h:

glFrustum(-2.0, 2.0, -2.0 \* float(h) / float(w), 2.0 \* float(h) / float(w), 2.0, 20.0)

else:

glFrustum(-2.0, 2.0, -2.0 \* float(w) / float(h), 2.0 \* float(w) / float(h), 2.0, 20.0)

glMatrixMode(GL\_MODELVIEW)

def main():

glutInit(sys.argv)

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH)

glutInitWindowSize(500, 500)

glutCreateWindow('colorcube viewpoint Python')

glEnable(GL\_DEPTH\_TEST)

glEnable(GL\_LIGHTING)

glEnable(GL\_LIGHT0)

glEnable(GL\_NORMALIZE)

glutReshapeFunc(myReshape)

glutDisplayFunc(MyDisplay)

glutAttachMenu(GLUT\_RIGHT\_BUTTON)

glutMainLoop()

if \_\_name\_\_ == "\_\_main\_\_":

main()