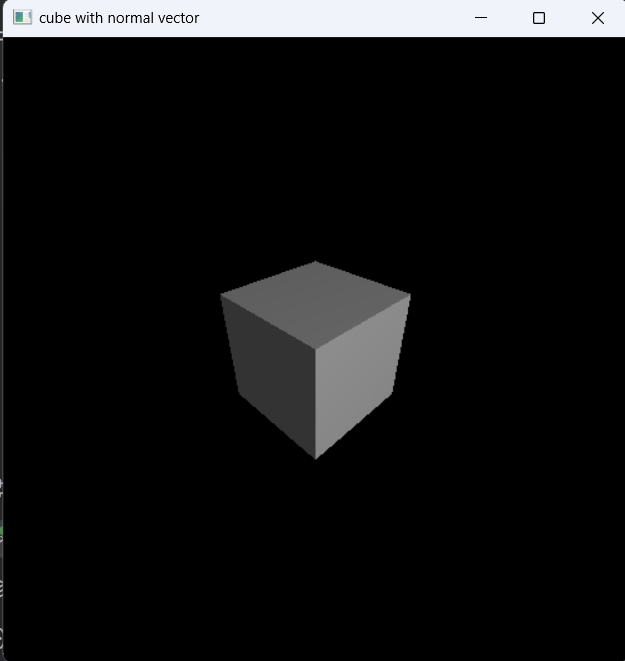
**컴퓨터그래픽스 Lab06 보고서**

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

**[과제 1]** Cube normal

**결과**



**코드**

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.

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) # 학생들이 작성

# 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 # 학생들이 작성

# 외적 벡터의 각 성분값을 외적벡터의 크기로 나눈다.

return normal

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

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

# 노멀벡터를 구하는 메소드에 인수로 줄 점 3개는 polygonNormal로 받은 숫자 4개 중 3개를 인덱스로 가지는 vertices 값이다.

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(): # 정육면체를 이루는 폴리곤 8개를 생성한다.

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)

def MyDisplay():

global myview

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

glLoadIdentity()

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, (1, 1, 1, 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))

gluLookAt(3.0, 3.0, 3.0, 0.0, 0.0, 0.0, 0.0, 1.0, 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('cube with normal vector')

glEnable(GL\_DEPTH\_TEST)

glEnable(GL\_LIGHTING)

glEnable(GL\_LIGHT0)

glEnable(GL\_NORMALIZE)

glShadeModel(GL\_SMOOTH)

glutReshapeFunc(myReshape)

glutDisplayFunc(MyDisplay)

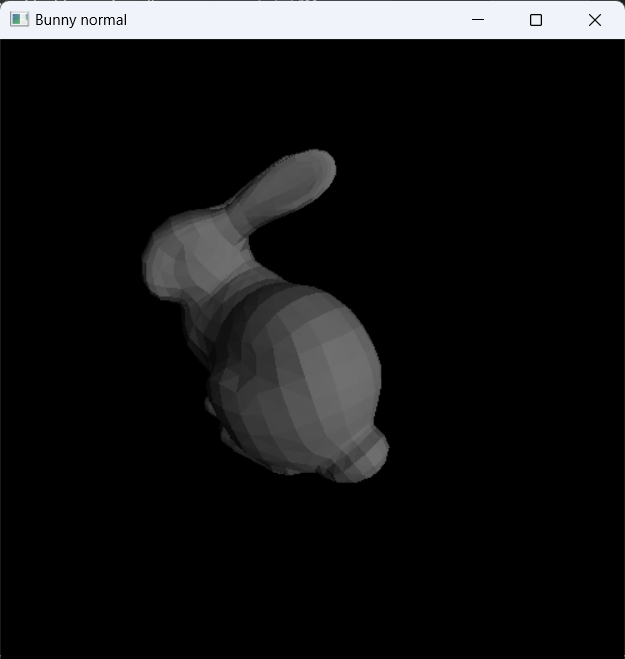
glutMainLoop()

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

main()

**[과제 2]** Bunny normal

**결과**



**코드**

from OpenGL.GL import \*

from OpenGL.GLUT import \*

from OpenGL.GLU import \*

from ObjLoader import \*

import numpy as np

myview = 3

xRot = 0.0

yRot = 0.0

obj = 0

def flatNormal(v1, v2, v3): # 노멀벡터를 구하는 메소드. x,y,z좌표를 가지는 점 3개를 인수로 받는다.

cross = np.cross(v3-v2,v1-v2) # 학생들이 작성

# 하나의 폴리곤 면 위에 위치한 두 벡터 v3-v2, v1-v2의 외적을 구한다.

length = np.linalg.norm(cross)

normal = (cross[0]/length, cross[1]/length,cross[2]/length) # 학생들이 작성

# 외적 벡터의 각 성분값을 외적벡터의 크기로 나눈다.

return normal

def loadRabbit():

global obj

index\_count = len(obj.vertex\_index)

normal = None

glBegin(GL\_POLYGON)

for i, vi in enumerate(obj.vertex\_index):

if i % 3 == 0:

v1 = np.array((obj.model[i \* 3]

, obj.model[i \* 3 + 1]

, obj.model[i \* 3 + 2]))

v2 = np.array((obj.model[(i + 1) \* 3]

, obj.model[(i + 1) \* 3 + 1]

, obj.model[(i + 1) \* 3 + 2]))

v3 = np.array((obj.model[(i + 2) \* 3]

, obj.model[(i + 2) \* 3 + 1]

, obj.model[(i + 2) \* 3 + 2]))

normal=flatNormal(v1,v2,v3) # 학생들이 작성

# 노멀벡터를 구하기 위한 인수로 각 폴리곤을 이루는 점 중 3개(v1~v3)를 준다.

if i > 0:

glEnd()

glBegin(GL\_POLYGON)

glNormal3fv((normal[0], normal[1], normal[2]))

glVertex3fv((obj.model[i \* 3]

, obj.model[i \* 3 + 1]

, obj.model[i \* 3 + 2]))

glEnd()

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))

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

glScalef(3, 3, 3)

glColor3f(1.0, 1.0, 1.0)

global obj

obj = ObjLoader()

obj.load\_model("res/bunny.obj")

loadRabbit()

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('Bunny normal')

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()