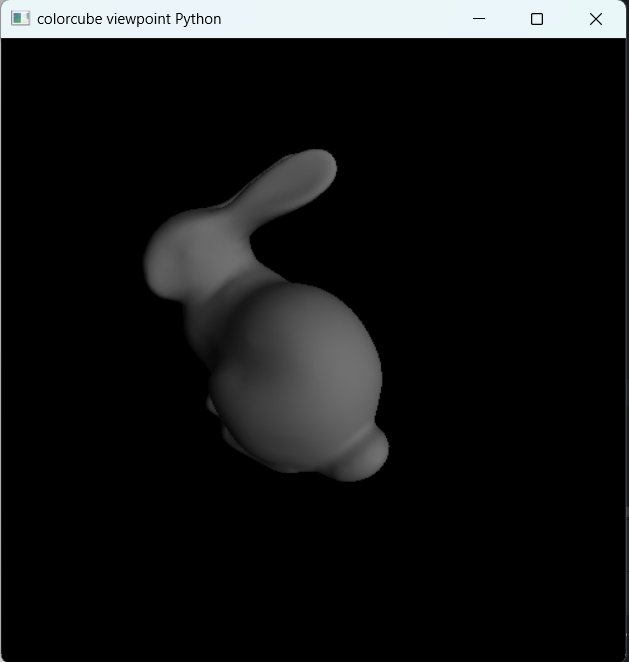
**컴퓨터그래픽스 Lab09 보고서**

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

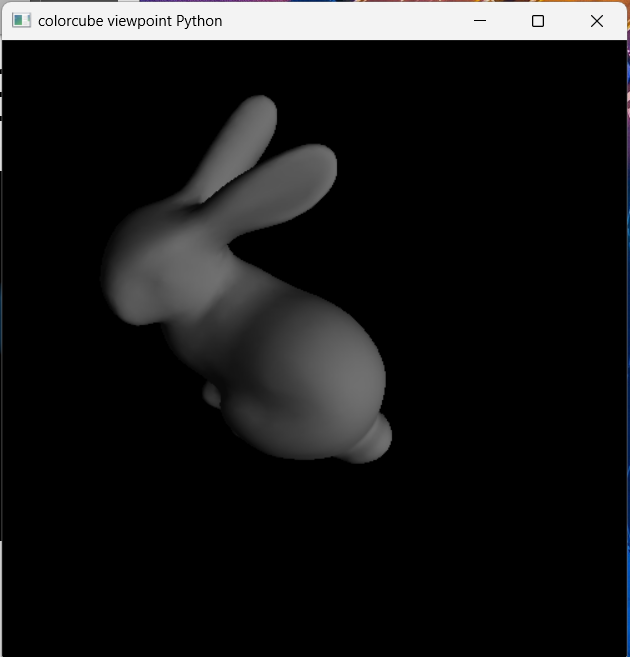
**[과제 1]** Smooth Shading 구현 (3종류)

**결과**

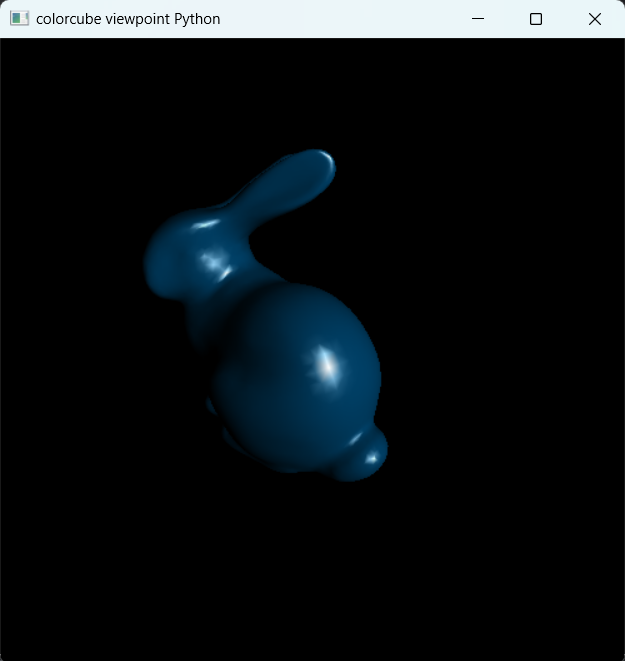
1) lab09\_2312282\_임다희\_01.py



2) lab09\_2312282\_임다희\_02.py



3) lab09\_2312282\_임다희\_03.py



**코드**

1) lab09\_2312282\_임다희\_01.py

#토끼 모델에 smooth shading을 적용한다.

from OpenGL.GL import \*

from OpenGL.GLUT import \*

from OpenGL.GLU import \*

from ObjLoader import \*

myview =3

xRot = 0

yRot = 0

obj = 0

def loadRabbit():

global obj

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

for i in range(index\_count):

# 폴리곤을 생성한다.

# START TRIANGLE

if i % 3 == 0:

# 3의 배수 번째 인덱스의 점에서 하나의 폴리곤 생성 시작.

glBegin(GL\_POLYGON)

glNormal3fv((obj.model[i \* 3 + index\_count \* 3 + index\_count \* 2 ]

, obj.model[i \* 3 + index\_count \* 3 + index\_count \* 2 + 1]

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

))

# Smooth Shading을 위해 각 점마다 고유한 법선벡터 하나를 계산해 적용한다.

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

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

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

# 점의 x,y,z좌표를 glVertex3fv에 넣어 폴리곤을 구성하는 점에 해당 점을 포함시킨다.

# END TRIANGLE

if i % 3 == 2:

# 3의 배수+2번째 인덱스의 점에서 폴리곤 생성 종료.

glEnd()

def MyDisplay():

global myview

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

#glMatrixMode(GL\_MODELVIEW)

glLoadIdentity()

#광원을 넣는다.

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

#주변광. r,g,b 알파값이 각각 0.2,0.2,0.2,1.0

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

#확산광. r,g,b,알파값이 각각 0.5, 0.5,0.5,1.0

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

#광원의 위치. x,y,z값이 각각 8.0,0.0,8.0

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

#카메라의 위치(3,3,3), 카메라가 바라보는 중심 위치(0,0,0), 카메라의 up vector(0,1,0).

glRotatef(xRot, 1.0, 0.0, 0.0)

glRotatef(yRot, 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\_smooth.obj")

#토끼 모델을 로드한다.

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('colorcube viewpoint Python')

glEnable(GL\_DEPTH\_TEST)

glEnable(GL\_LIGHTING)

glEnable(GL\_LIGHT0)

glEnable(GL\_NORMALIZE)

#glShadeModel(GL\_SMOOTH)

glutReshapeFunc(myReshape)

glutDisplayFunc(MyDisplay)

glutAttachMenu(GLUT\_RIGHT\_BUTTON)

glutMainLoop()

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

main()

2) lab09\_2312282\_임다희\_02.py

#토끼 모델에 Smooth shading을 적용하고, 모델을 회전시켜 카메라상에서 보이는 모습을 변경시킨다.

from OpenGL.GL import \*

from OpenGL.GLUT import \*

from OpenGL.GLU import \*

from ObjLoader import \*

myview =3

xRot = 17.5 #모델의 x축 기준 회전 각도를 나타내는 변수. 17.5도

yRot = 10 # 모델의 y축 기준 회전 각도를 나타내는 변수. 10도

zRot = -17.5 #모델의 z축 기준 회전 각도를 나타내는 변수. -17.5도

obj = 0

def loadRabbit():

global obj

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

for i in range(index\_count):

# START TRIANGLE

if i % 3 == 0:

# 3의 배수 번째 인덱스의 점에서 하나의 폴리곤 생성 시작.

glBegin(GL\_POLYGON)

glNormal3fv((obj.model[i \* 3 + index\_count \* 3 + index\_count \* 2 ]

, obj.model[i \* 3 + index\_count \* 3 + index\_count \* 2 + 1]

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

))

# Smooth Shading을 위해 각 점마다 고유한 법선벡터 하나를 계산해 적용한다.

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

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

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

# 점의 x,y,z좌표를 glVertex3fv에 넣어 폴리곤을 구성하는 점에 해당 점을 포함시킨다.

# END TRIANGLE

if i % 3 == 2:

# 3의 배수+2번째 인덱스의 점에서 폴리곤 생성 종료.

glEnd()

def MyDisplay():

global myview

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

#glMatrixMode(GL\_MODELVIEW)

glLoadIdentity()

# 광원을 넣는다.

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

# 주변광. r,g,b 알파값이 각각 0.2,0.2,0.2,1.0

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

# 확산광. r,g,b,알파값이 각각 0.5, 0.5,0.5,1.0

glLightfv(GL\_LIGHT0, GL\_POSITION, (5.0, -1.0, 1.5, 1.0))

# 광원의 위치. x,y,z값이 각각 5.0,-1.0,1.5.

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

# 카메라의 위치(3,3,3), 카메라가 바라보는 중심 위치(0,0,0), 카메라의 up vector(0,1,0).

glRotatef(xRot, 1.0, 0.0, 0.0)

glRotatef(yRot, 0.0, 1.0, 0.0)

glRotatef(zRot, 0.0, 0.0, 1.0)

#glRotatef를 통해 사전에 정의한 변수 값만큼 모델을 x,y,z축 기준으로 회전시킨다.

#x축으로 17.5도, y축으로 10도, z축으로 -17.5도만큼 회전하였다.

glScalef(3, 3, 3)

glColor3f(1.0, 1.0, 1.0)

global obj

obj = ObjLoader()

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

# 토끼 모델을 로드한다.

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('colorcube viewpoint Python')

glEnable(GL\_DEPTH\_TEST)

glEnable(GL\_LIGHTING)

glEnable(GL\_LIGHT0)

glEnable(GL\_NORMALIZE)

#glShadeModel(GL\_SMOOTH)

glutReshapeFunc(myReshape)

glutDisplayFunc(MyDisplay)

glutAttachMenu(GLUT\_RIGHT\_BUTTON)

glutMainLoop()

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

main()

3) lab09\_2312282\_임다희\_03.py

#토끼 모델에 Smooth shading을 적용하고, lighting을 통해 모델의 색과 표면 특성을 표현한다.

from OpenGL.GL import \*

from OpenGL.GLUT import \*

from OpenGL.GLU import \*

from ObjLoader import \*

myview =3

#광원의 특성(세기)을 나타내는 계수를 정의한다.

light\_ambient = (0.1, 0.1, 0.1, 1.0)

# Ia : 광원의 주변광 세기를 조절한다. 주변광의 r,g,b값이 각각 0.1,0.1,0.1

light\_diffuse = (0, 0.5, 0.5, 1.0)

# Id : 광원의 확산광 세기를 조절한다. 확산광의 r,g,b값이 각각 0,0.5,0.5

light\_specular = (1, 1, 1, 1.0)

# Is : 광원의 경면광 세기를 조절한다. 경면광의 r,g,b값이 각각 1,1,1

light\_position1 = (10, 0.0, 10.0, 1.0)

# 광원의 위치. x,y,z값이 각각 10,0,10인 시점에 광원이 위치한다.

#물체의 특성(빛을 반사하는 정도)을 나타내는 계수를 정의한다.

no\_mat = (0.0, 0.0, 0.0, 1.0)

#no\_mat: 물체가 특정 빛을 반사하지 않는 경우에 계수로 넣어주는 값.

mat\_ambient = (0.0, 0.0, 1.0, 1.0)

# Ka : 물체의 주변광 계수를 조절한다. 물체에 의해 반사되는 주변광의 r,g,b값이 각각 0.0,0.0,1.0

mat\_diffuse = (0.0, 0.5, 0.8, 1.0) #kd

# Kd : 물체의 확산광 계수를 조절한다. 물체에 의해 반사되는 확산광의 r,g,b값이 각각 0.0,0.5,0.8

mat\_specular = (1.0, 1.0, 1.0, 1.0) #Ks

# Ks : 물체의 경면광 계수를 조절한다. 물체에 의해 반사되는 경면광의 r,g,b값이 각각 1.0, 1.0,1.0

no\_shininess = 0.0 # Shininess Coefficient

# 물체의 광택 계수를 조정한다. 광택 계수가 0인 경우

low\_shininess = 10.0 # Shininess Coefficient

# 물체의 광택 계수를 조정한다. 광택 계수가 10인 경우

high\_shininess = 100 # Shininess Coefficient

# 물체의 광택 계수를 조정한다. 광택 계수가 100인 경우

xRot = 0.0

yRot = 0.0

obj = 0

def loadRabbit():

global obj

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

for i in range(index\_count):

# 폴리곤을 생성한다.

# START TRIANGLE

if i % 3 == 0:

# 3의 배수 번째 인덱스의 점에서 하나의 폴리곤 생성 시작.

glBegin(GL\_POLYGON)

glNormal3fv((obj.model[i \* 3 + index\_count \* 3 + index\_count \* 2 ]

, obj.model[i \* 3 + index\_count \* 3 + index\_count \* 2 + 1]

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

))

# Smooth Shading을 위해 각 점마다 고유한 법선벡터 하나를 계산해 적용한다.

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

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

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

# 점의 x,y,z좌표를 glVertex3fv에 넣어 폴리곤을 구성하는 점에 해당 점을 포함시킨다.

# END TRIANGLE

if i % 3 == 2:

# 3의 배수+2번째 인덱스의 점에서 폴리곤 생성 종료.

glEnd()

def MyDisplay():

global myview

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

#glMatrixMode(GL\_MODELVIEW)

glLoadIdentity()

#광원을 넣는다.

#사전에 정의한 계수의 주변광, 확산광, 경면광, 광원의 위치를 적용한다.

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, light\_ambient)

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse)

glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular)

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position1)

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

# 카메라의 위치(3,3,3), 카메라가 바라보는 중심 위치(0,0,0), 카메라의 up vector(0,1,0).

glRotatef(xRot, 1.0, 0.0, 0.0)

glRotatef(yRot, 0.0, 1.0, 0.0)

glScalef(3, 3, 3)

glColor3f(1.0, 1.0, 1.0)

# 토끼 모델에 사전에 정의한 물체의 특성을 입힌다.

glMaterialfv(GL\_FRONT, GL\_AMBIENT, no\_mat)

#물체의 주변광 반사 계수를 no\_mat, 즉 (0,0,0,1)으로 지정한다.

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse)

#물체의 확산광 반사 계수를 mat\_diffuse, 즉(0.0,0.5,0.8,1.0)으로 지정한다.

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular)

#물체의 경면광 반사 계수를 mat\_specular, 즉(1.0,1.0,1.0)으로 지정한다.

glMaterialf(GL\_FRONT, GL\_SHININESS, high\_shininess)

#광택 계수를 high\_shiniess(100)으로 지정한다.

#해당 토끼 모델은 확산광, 경면광만을 반사하고 이들은 각각 0.0,0.5,0.8/1.0,1.0,1.0 의 rgb 값을 가지는 빛으로 표현된다.

#물체의 광택 계수는 100으로, 표면이 반짝이는 성질을 가지고 있다.

global obj

obj = ObjLoader()

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

# 토끼 모델을 로드한다.

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('colorcube viewpoint Python')

glEnable(GL\_DEPTH\_TEST)

glEnable(GL\_LIGHTING)

glEnable(GL\_LIGHT0)

glEnable(GL\_NORMALIZE)

#glShadeModel(GL\_SMOOTH)

glutReshapeFunc(myReshape)

glutDisplayFunc(MyDisplay)

glutAttachMenu(GLUT\_RIGHT\_BUTTON)

glutMainLoop()

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

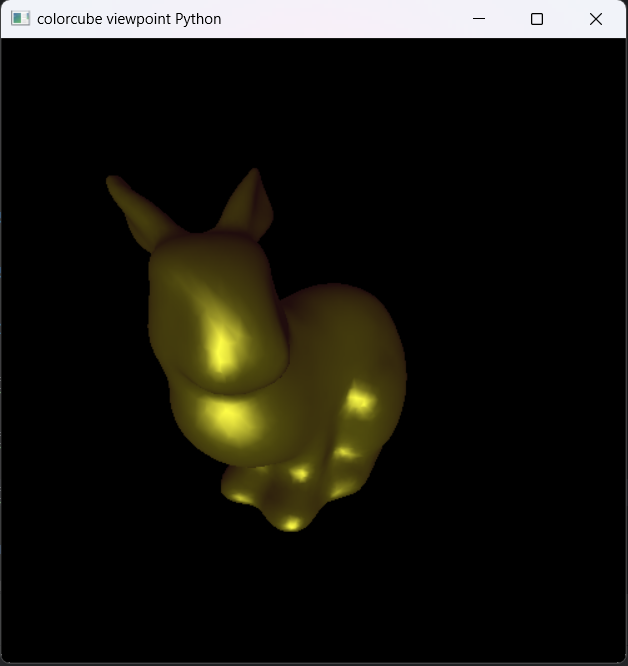
main()

#loadRabbit()

**[과제 2]** Transform, viewing transform, lighting 중 한 가지 이상 활용한 bunny 모델 렌더링

**결과**

4) lab09\_2312282\_임다희\_04.py



**코드**

4) lab09\_2312282\_임다희\_04.py

#토끼 모델에 Smooth shading을 적용하고,

#model transform, viewing transform, lighting을 활용하여 렌더링한다.

from OpenGL.GL import \*

from OpenGL.GLUT import \*

from OpenGL.GLU import \*

from ObjLoader import \*

myview =3

#광원의 특성(세기)을 나타내는 계수를 정의한다(lighting 활용).

light\_ambient = (0.5, 0.5, 0.5, 1.0)

# Ia : 광원의 주변광 세기를 조절한다. 주변광의 r,g,b값이 각각 0.5,0.5,0.5

light\_diffuse = (0.75, 0.5, 0.0, 1.0)

# Id : 광원의 확산광 세기를 조절한다. 확산광의 r,g,b값이 각각 0.75, 0.5,0.0

light\_specular = (0.75, 1, 0.5, 1.0)

# Is : 광원의 경면광 세기를 조절한다. 경면광의 r,g,b값이 각각 0.75,1,0.5

light\_position1 = (0.0, -5.0, 10.0, 1.0)

# 광원의 위치. x,y,z값이 각각 0.0,-5.0,10.0인 시점에 광원이 위치한다.

#물체의 특성(빛을 반사하는 정도)을 나타내는 계수를 정의한다.(lighting 활용).

no\_mat = (0.0, 0.0, 0.0, 1.0)

#no\_mat: 물체가 특정 빛을 반사하지 않는 경우에 계수로 넣어주는 값.

mat\_ambient = (0.25, 0.1, 0.1, 1.0)

# Ka : 물체의 주변광 계수를 조절한다. 물체에 의해 반사되는 주변광의 r,g,b값이 각각 0.25,0.1,0.1

mat\_diffuse = (0.25, 0.5, 0.25, 1.0)

# Kd : 물체의 확산광 계수를 조절한다. 물체에 의해 반사되는 확산광의 r,g,b값이 각각 0.25,0.5,0.25

mat\_specular = (1.0, 0.75, 0.5, 1.0) #Ks

# Ks : 물체의 경면광 계수를 조절한다. 물체에 의해 반사되는 경면광의 r,g,b값이 각각 1.0, 0.75,0.5

no\_shininess = 0.0 # Shininess Coefficient

# 광원의 광택 계수를 조정한다. 광택 계수가 0인 경우

low\_shininess = 25 # Shininess Coefficient

# 광원의 광택 계수를 조정한다. 광택 계수가 25인 경우

high\_shininess = 100 # Shininess Coefficient

# 광원의 광택 계수를 조정한다. 광택 계수가 100인 경우

#x,y,z축 기준으로 모델을 회전시키기 위한 계수를 정의한다(model transform 활용).

xRot = 0.0 #모델의 x축 기준 회전 각도를 나타내는 변수.

yRot = 60.0 #모델의 y축 기준 회전 각도를 나타내는 변수.

zRot= 0.0 #모델의 z축 기준 회전 각도를 나타내는 변수.

obj = 0

def loadRabbit():

global obj

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

for i in range(index\_count):

# START TRIANGLE

if i % 3 == 0:

# 3의 배수 번째 인덱스의 점에서 하나의 폴리곤 생성 시작.

glBegin(GL\_POLYGON)

glNormal3fv((obj.model[i \* 3 + index\_count \* 3 + index\_count \* 2 ]

, obj.model[i \* 3 + index\_count \* 3 + index\_count \* 2 + 1]

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

))

# Smooth Shading을 위해 각 점마다 고유한 법선벡터 하나를 계산해 적용한다.

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

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

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

# 점의 x,y,z좌표를 glVertex3fv에 넣어 폴리곤을 구성하는 점에 해당 점을 포함시킨다.

# END TRIANGLE

if i % 3 == 2:

# 3의 배수+2번째 인덱스의 점에서 폴리곤 생성 종료.

glEnd()

def MyDisplay():

global myview

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

#glMatrixMode(GL\_MODELVIEW)

glLoadIdentity()

# 광원을 넣는다.

# 사전에 정의한 계수의 주변광, 확산광, 경면광, 광원의 위치를 적용한다.

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, light\_ambient)

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse)

glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular)

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position1)

gluLookAt(1.0, 2.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0)

# 카메라의 위치(1,2,5), 카메라가 바라보는 중심 위치(0,0,0), 카메라의 up vector(0,1,0).

# 카메라의 위치를 이동시켜 (1,2,5) 좌표에서 바라본 모델 모습을 얻는다. (viewing transform 활용)

glRotatef(xRot, 1.0, 0.0, 0.0)

glRotatef(yRot, 0.0, 1.0, 0.0)

# x, y축 기준으로 물체를 사전에 정의한 계수만큼 회전시킨다.

# 최종적으로 y축 기준으로 60도만큼 돌아간 모델을 얻을 수 있다.

glScalef(3, 3, 3)

glColor3f(1.0, 1.0, 1.0)

~~# 토끼 모델에 사전에 정의한 물체의 특성을 입힌다.~~

glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat\_ambient)

# 물체의 주변광 반사 계수를 mat\_ambient, 즉 (0.25, 0.1, 0.1, 1.0)으로 지정한다.

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse)

# 물체의 확산광 반사 계수를 mat\_diffuse, 즉 (0.25, 0.5, 0.25, 1.0)으로 지정한다.

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular)

# 물체의 경면광 반사 계수를 mat\_specular, 즉 (1.0, 0.75, 0.5, 1.0) 으로 지정한다.

glMaterialf(GL\_FRONT, GL\_SHININESS, low\_shininess)

# 광택 계수를 low\_shiniess(25)으로 지정한다.

# 해당 토끼 모델은 주변광, 확산광, 경면광만을 반사하고

# 이들은 각각 0.25,0.1,0.1/0.25,0.5,0.25/1.0,0.75,0.5 의 rgb 값을 가지는 빛으로 표현된다.

# 물체의 광택 계수는 25로, 표면이 조금 반짝이는 성질을 가지고 있다.

global obj

obj = ObjLoader()

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

# 토끼 모델을 로드한다.

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('colorcube viewpoint Python')

glEnable(GL\_DEPTH\_TEST)

glEnable(GL\_LIGHTING)

glEnable(GL\_LIGHT0)

glEnable(GL\_NORMALIZE)

#glShadeModel(GL\_SMOOTH)

glutReshapeFunc(myReshape)

glutDisplayFunc(MyDisplay)

glutAttachMenu(GLUT\_RIGHT\_BUTTON)

glutMainLoop()

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

main()

#loadRabbit()