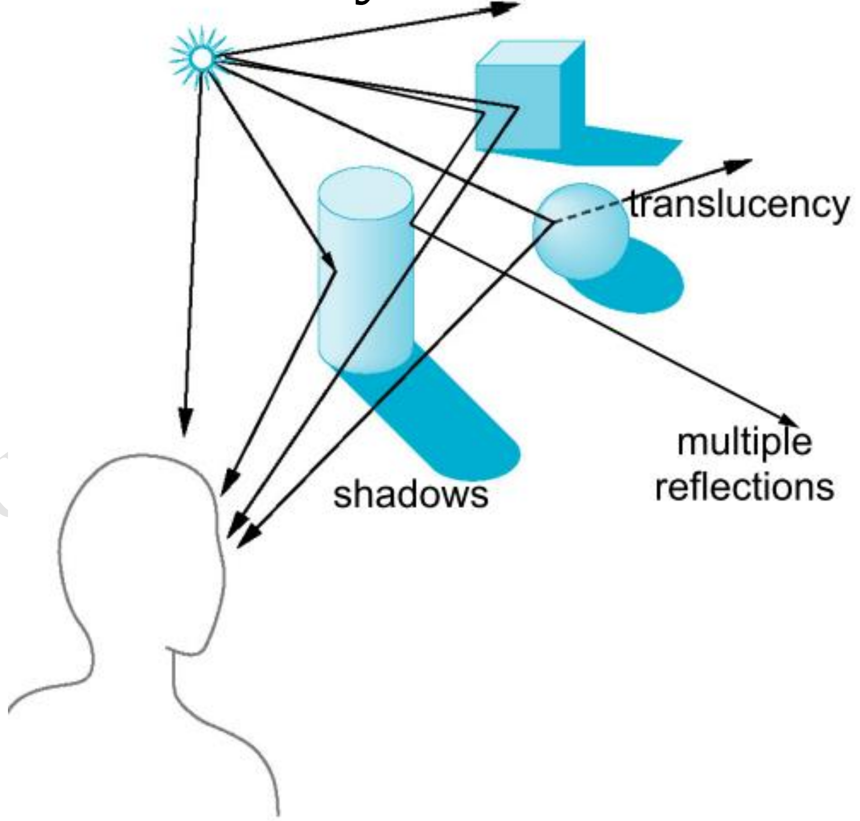
Advanced Computer Graphics

Lecture-08 Introduction to OpenGL-8

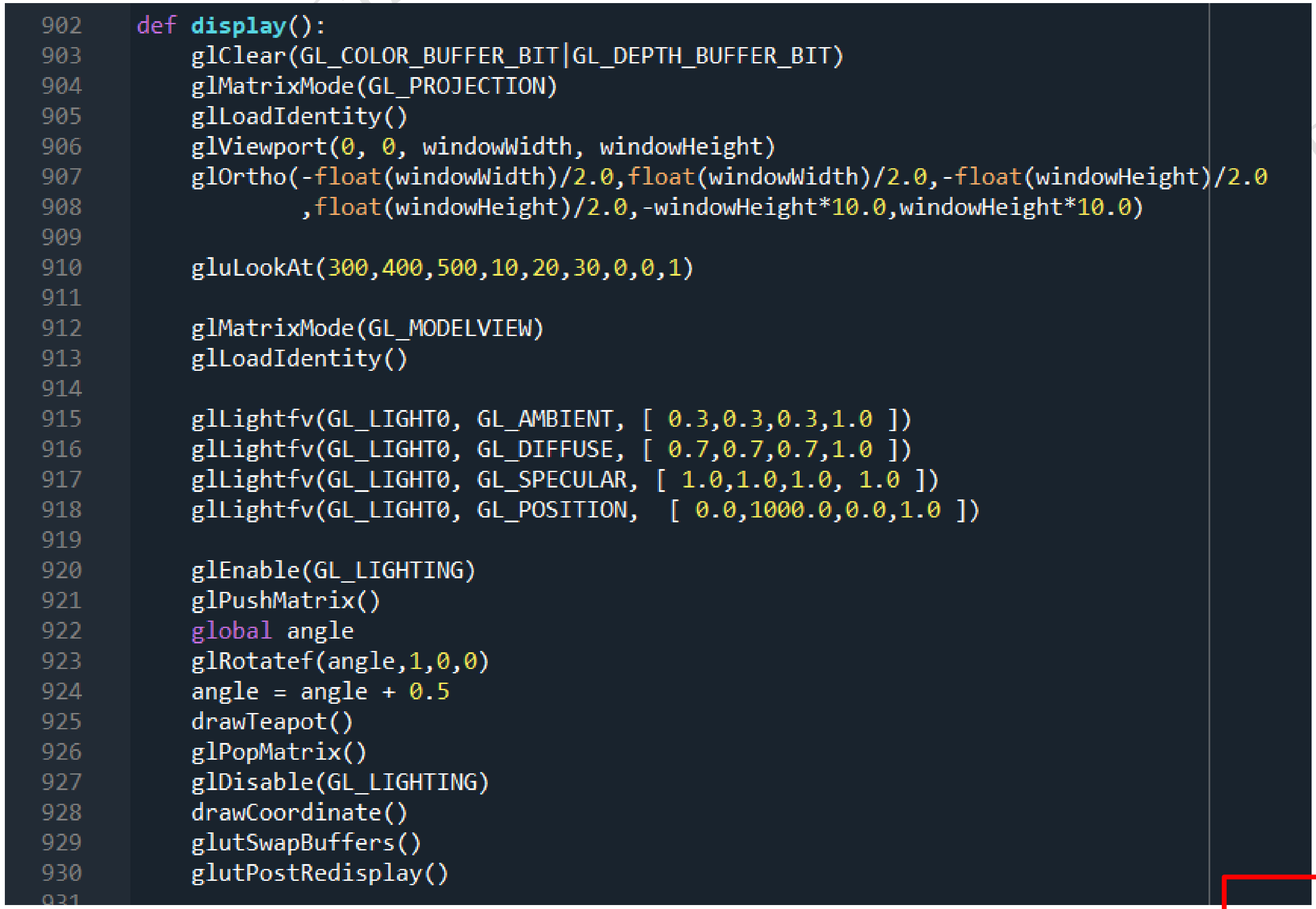
Tzung-Han Lin
National Taiwan University of Science and Technology
Graduate Institute of Color and Illumination Technology
e-mail: thl@mail.ntust.edu.tw

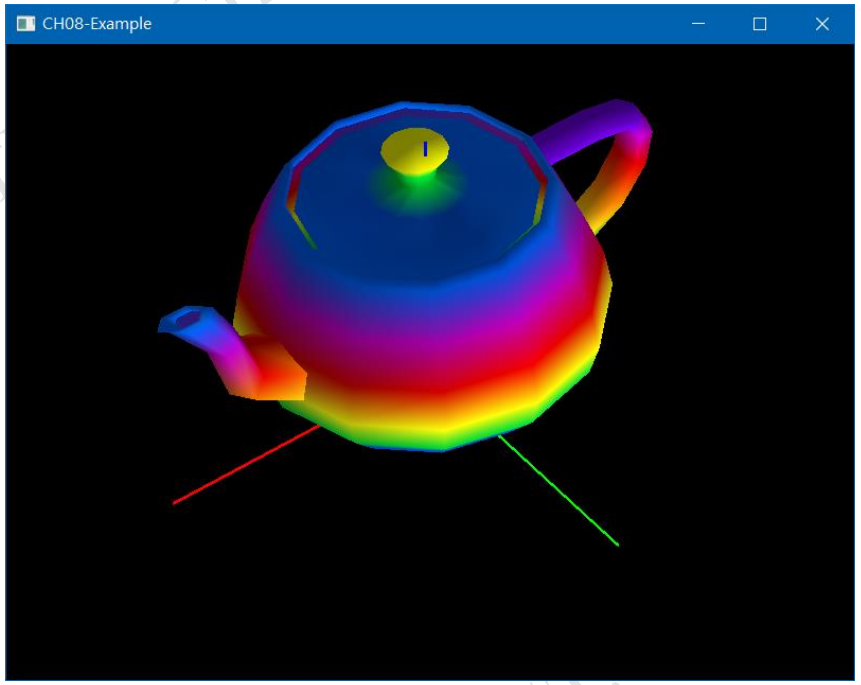
Render a “Color”

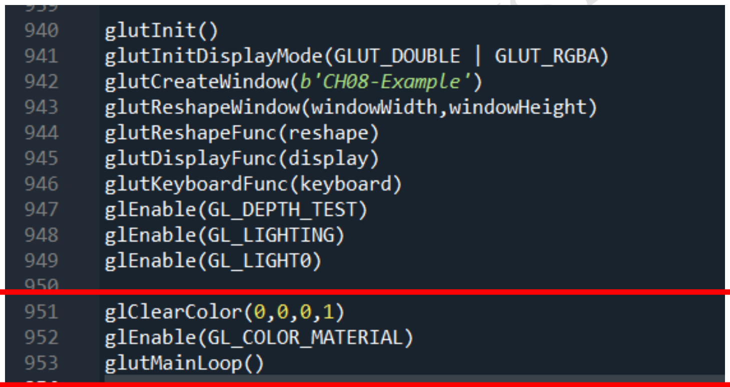
- What you see is the integration of “color of reflection” which is the combination of material properties (reflection for spectrum), light spectrum and human vision system.



Draw Color and Enable Color Material


```
def display():
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glViewport(0, 0, windowWidth, windowHeight)
    glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.0,-float(windowHeight)/2.0
            ,float(windowHeight)/2.0,-windowHeight*10.0,windowHeight*10.0)

    gluLookAt(300,400,500,10,20,30,0,0,1)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    glLightfv(GL_LIGHT0, GL_AMBIENT, [ 0.3,0.3,0.3,1.0 ])
    glLightfv(GL_LIGHT0, GL_DIFFUSE, [ 0.7,0.7,0.7,1.0 ])
    glLightfv(GL_LIGHT0, GL_SPECULAR, [ 1.0,1.0,1.0, 1.0 ])
    glLightfv(GL_LIGHT0, GL_POSITION,  [ 0.0,1000.0,0.0,1.0 ])

    glEnable(GL_LIGHTING)
    glPushMatrix()
    global angle
    glRotatef(angle,1,0,0)
    angle = angle + 0.5
    drawTeapot()
    glPopMatrix()
    glDisable(GL_LIGHTING)
    drawCoordinate()
    glutSwapBuffers()
    glutPostRedisplay()
```

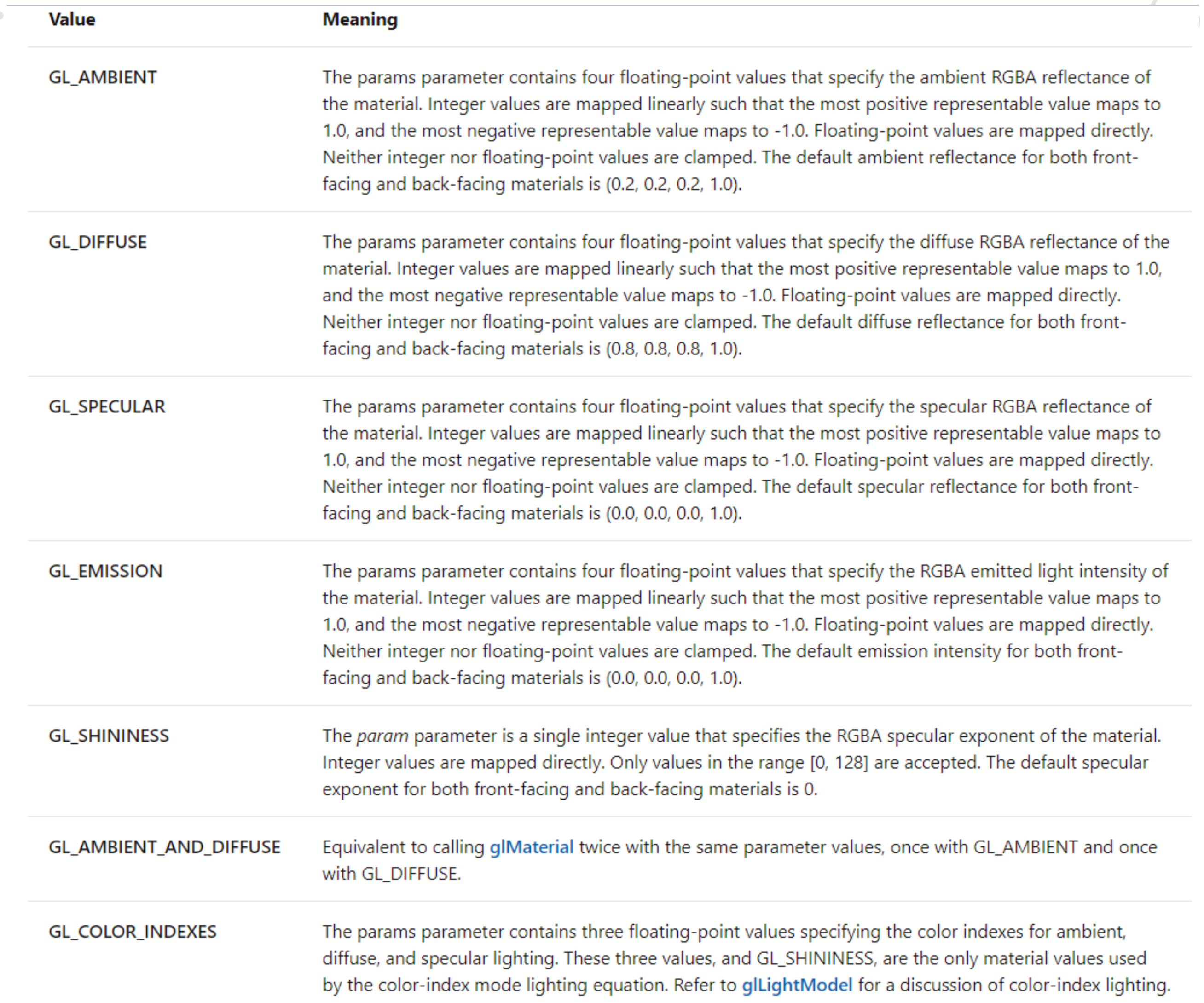



```
    glutInit()
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA)
    glutCreateWindow(b'CH08-Example')
    glutReshapeWindow(windowWidth,windowHeight)
    glutReshapeFunc(reshape)
    glutDisplayFunc(display)
    glutKeyboardFunc(keyboard)
    glEnable(GL_DEPTH_TEST)
    glEnable(GL_LIGHTING)
    glEnable(GL_LIGHT0)

    glClearColor(0,0,0,1)
    glEnable(GL_COLOR_MATERIAL)
    glutMainLoop()
```

openGL LIGHT default value

Parameter Name	Default Value	Meaning
GL_AMBIENT	(0.0, 0.0, 0.0, 1.0)	ambient RGBA intensity of light
GL_DIFFUSE	(1.0, 1.0, 1.0, 1.0)	diffuse RGBA intensity of light
GL_SPECULAR	(1.0, 1.0, 1.0, 1.0)	specular RGBA intensity of light
GL_POSITION	(0.0, 0.0, 1.0, 0.0)	(*x*, *y*, *z*, *w*) position of light
GL_SPOT_DIRECTION	(0.0, 0.0, -1.0)	(*x*, *y*, *z*) direction of spotlight
GL_SPOT_EXPONENT	0.0	spotlight exponent
GL_SPOT_CUTOFF	180.0	spotlight cutoff angle
GL_CONSTANT_ATTENUATION	1.0	constant attenuation factor
GL_LINEAR_ATTENUATION	0.0	linear attenuation factor
GL_QUADRATIC_ATTENUATION	0.0	quadratic attenuation factor

openGL MATERIAL default value

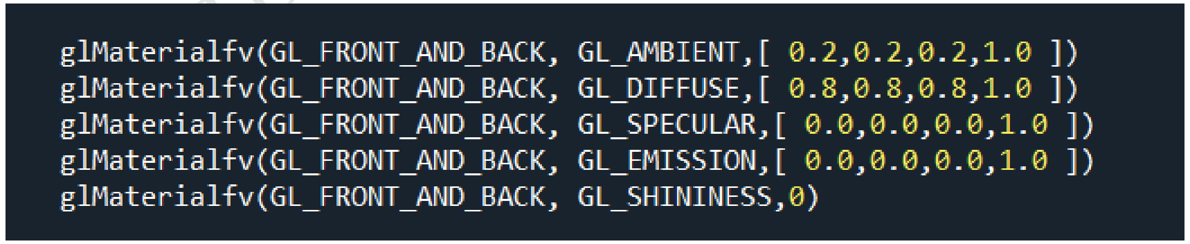


Value	Meaning
GL_AMBIENT	The params parameter contains four floating-point values that specify the ambient RGBA reflectance of the material. Integer values are mapped linearly such that the most positive representable value maps to 1.0, and the most negative representable value maps to -1.0. Floating-point values are mapped directly. Neither integer nor floating-point values are clamped. The default ambient reflectance for both front-facing and back-facing materials is (0.2, 0.2, 0.2, 1.0).
GL_DIFFUSE	The params parameter contains four floating-point values that specify the diffuse RGBA reflectance of the material. Integer values are mapped linearly such that the most positive representable value maps to 1.0, and the most negative representable value maps to -1.0. Floating-point values are mapped directly. Neither integer nor floating-point values are clamped. The default diffuse reflectance for both front-facing and back-facing materials is (0.8, 0.8, 0.8, 1.0).
GL_SPECULAR	The params parameter contains four floating-point values that specify the specular RGBA reflectance of the material. Integer values are mapped linearly such that the most positive representable value maps to 1.0, and the most negative representable value maps to -1.0. Floating-point values are mapped directly. Neither integer nor floating-point values are clamped. The default specular reflectance for both front-facing and back-facing materials is (0.0, 0.0, 0.0, 1.0).
GL_EMISSION	The params parameter contains four floating-point values that specify the RGBA emitted light intensity of the material. Integer values are mapped linearly such that the most positive representable value maps to 1.0, and the most negative representable value maps to -1.0. Floating-point values are mapped directly. Neither integer nor floating-point values are clamped. The default emission intensity for both front-facing and back-facing materials is (0.0, 0.0, 0.0, 1.0).
GL_SHININESS	The *param* parameter is a single integer value that specifies the RGBA specular exponent of the material. Integer values are mapped directly. Only values in the range [0, 128] are accepted. The default specular exponent for both front-facing and back-facing materials is 0.
GL_AMBIENT_AND_DIFFUSE	Equivalent to calling glMaterial twice with the same parameter values, once with GL_AMBIENT and once with GL_DIFFUSE.
GL_COLOR_INDEXES	The params parameter contains three floating-point values specifying the color indexes for ambient, diffuse, and specular lighting. These three values, and GL_SHININESS, are the only material values used by the color-index mode lighting equation. Refer to glLightModel for a discussion of color-index lighting.

https://docs.microsoft.com/en-us/windows/win32/opengl/glmaterialfv

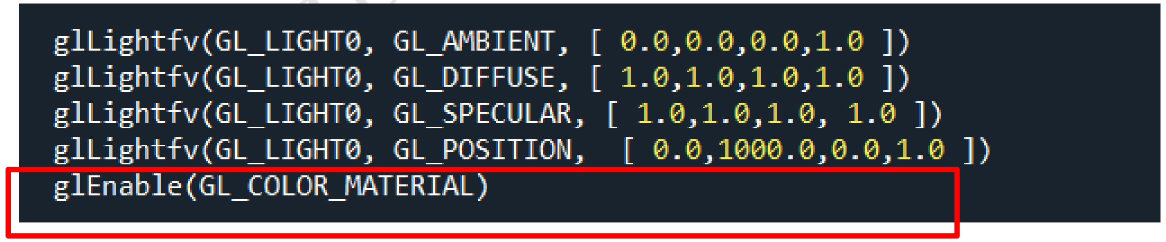
Material and Light in openGL

Material Property



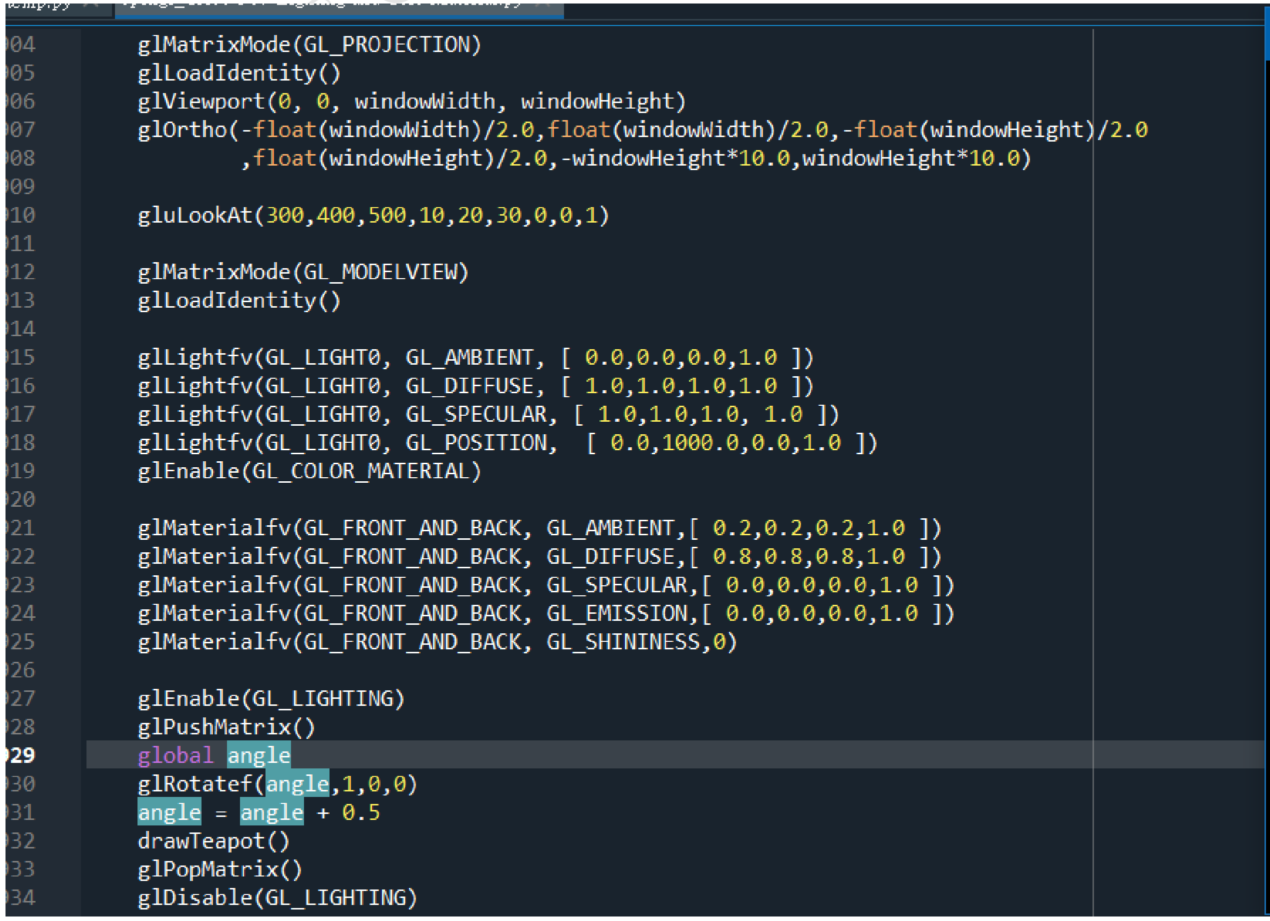
```
glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT,[ 0.2,0.2,0.2,1.0 ])
glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE,[ 0.8,0.8,0.8,1.0 ])
glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR,[ 0.0,0.0,0.0,1.0 ])
glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION,[ 0.0,0.0,0.0,1.0 ])
glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS,0)
```

Lighting

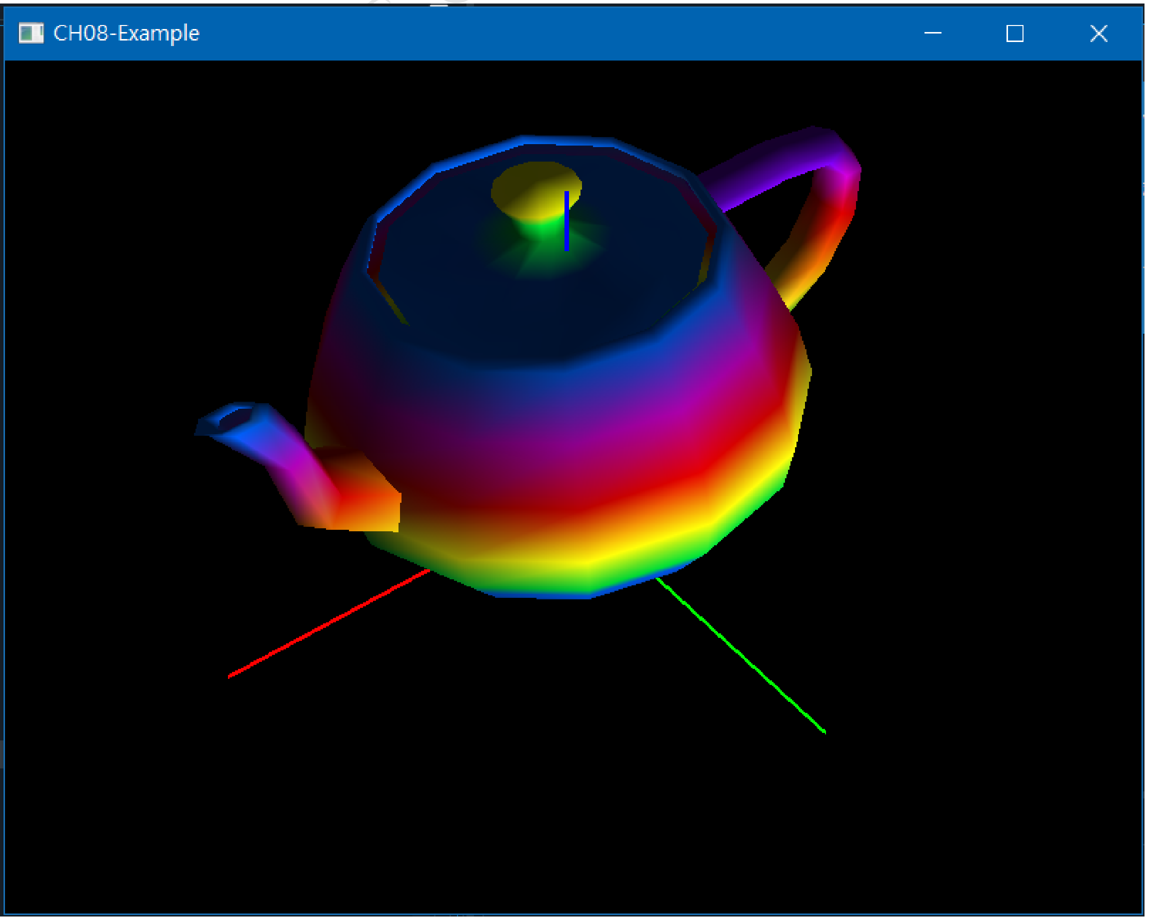


```
glLightfv(GL_LIGHT0, GL_AMBIENT, [ 0.0,0.0,0.0,1.0 ])
glLightfv(GL_LIGHT0, GL_DIFFUSE, [ 1.0,1.0,1.0,1.0 ])
glLightfv(GL_LIGHT0, GL_SPECULAR, [ 1.0,1.0,1.0, 1.0 ])
glLightfv(GL_LIGHT0, GL_POSITION,  [ 0.0,1000.0,0.0,1.0 ])
glEnable(GL_COLOR_MATERIAL)
```

Draw Color and Enable Color Material (default value)



```
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glViewport(0, 0, windowWidth, windowHeight)
    glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.0,-float(windowHeight)/2.0
            ,float(windowHeight)/2.0,-windowHeight*10.0,windowHeight*10.0)

    gluLookAt(300,400,500,10,20,30,0,0,1)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    glLightfv(GL_LIGHT0, GL_AMBIENT, [ 0.0,0.0,0.0,1.0 ])
    glLightfv(GL_LIGHT0, GL_DIFFUSE, [ 1.0,1.0,1.0,1.0 ])
    glLightfv(GL_LIGHT0, GL_SPECULAR, [ 1.0,1.0,1.0, 1.0 ])
    glLightfv(GL_LIGHT0, GL_POSITION,  [ 0.0,1000.0,0.0,1.0 ])
    glEnable(GL_COLOR_MATERIAL)

    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT,[ 0.2,0.2,0.2,1.0 ])
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE,[ 0.8,0.8,0.8,1.0 ])
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR,[ 0.0,0.0,0.0,1.0 ])
    glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION,[ 0.0,0.0,0.0,1.0 ])
    glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS,0)

    glEnable(GL_LIGHTING)
    glPushMatrix()
    global angle
    glRotatef(angle,1,0,0)
    angle = angle + 0.5
    drawTeapot()
    glPopMatrix()
    glDisable(GL_LIGHTING)
```



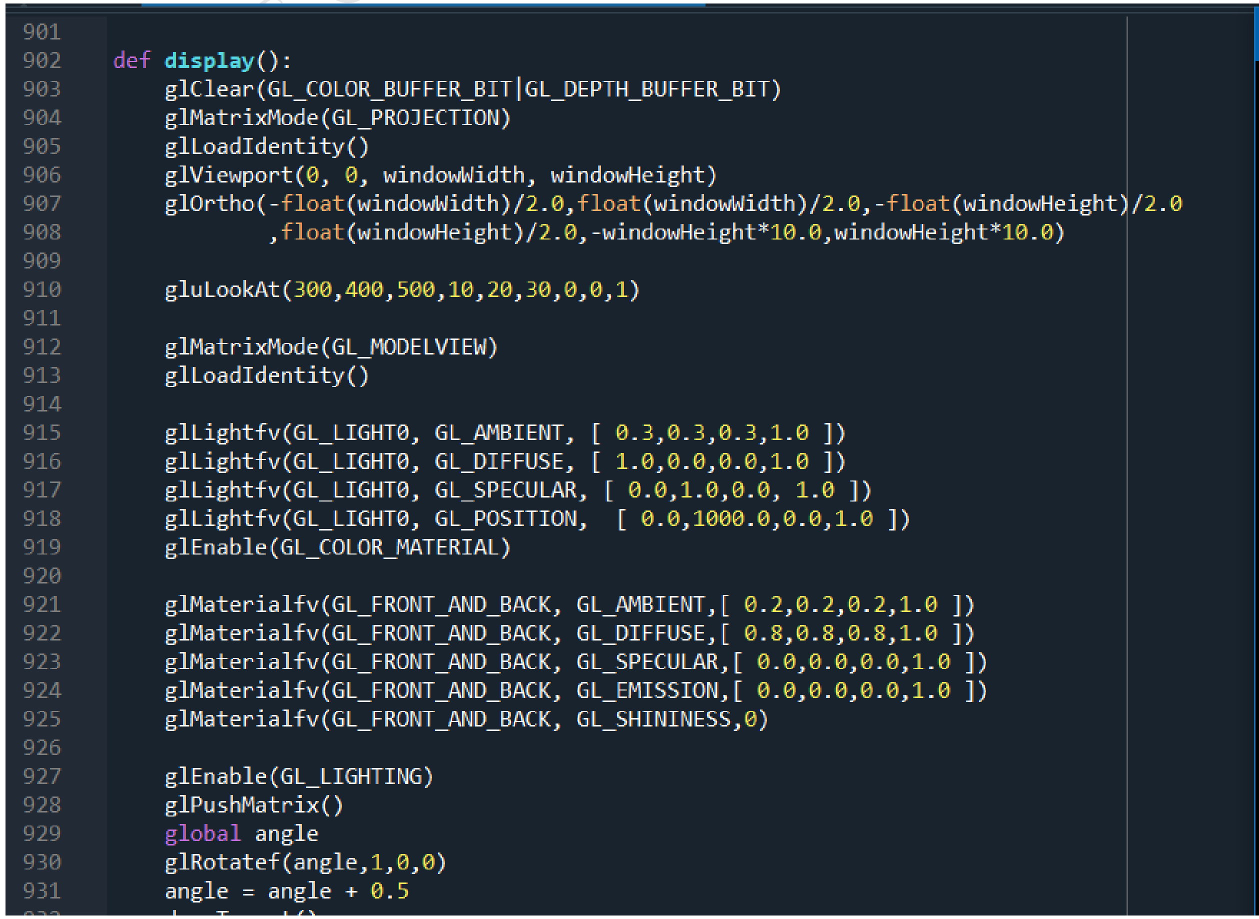
計算機對光反應的模型是經過簡化....

$$L_v = \int_{380}^{830} L_{e,\lambda} V(\lambda)\, d\lambda \qquad Q_\lambda = \bar{r}(\lambda)R + \bar{g}(\lambda)G + \bar{b}(\lambda)B$$

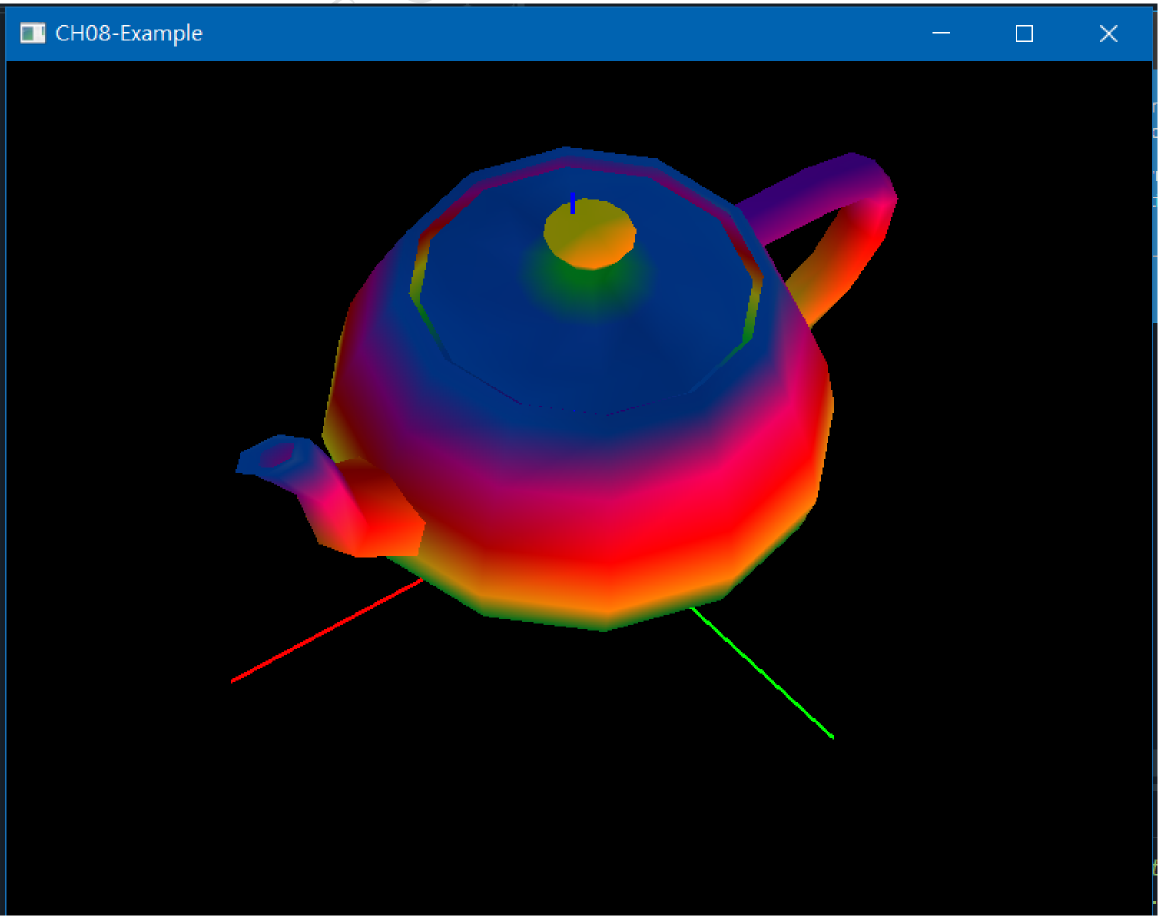
$$I = I_E + K_A I_{AL} + \sum_i [K_D (N \cdot L_i) I_i + K_S (V \cdot R_i)^n I_i]$$

如何決定材料特性？？？
電腦圖學對材質屬性大多假設為”等向””均質”
可透過BRDF測量獲得比較精確的物理模型

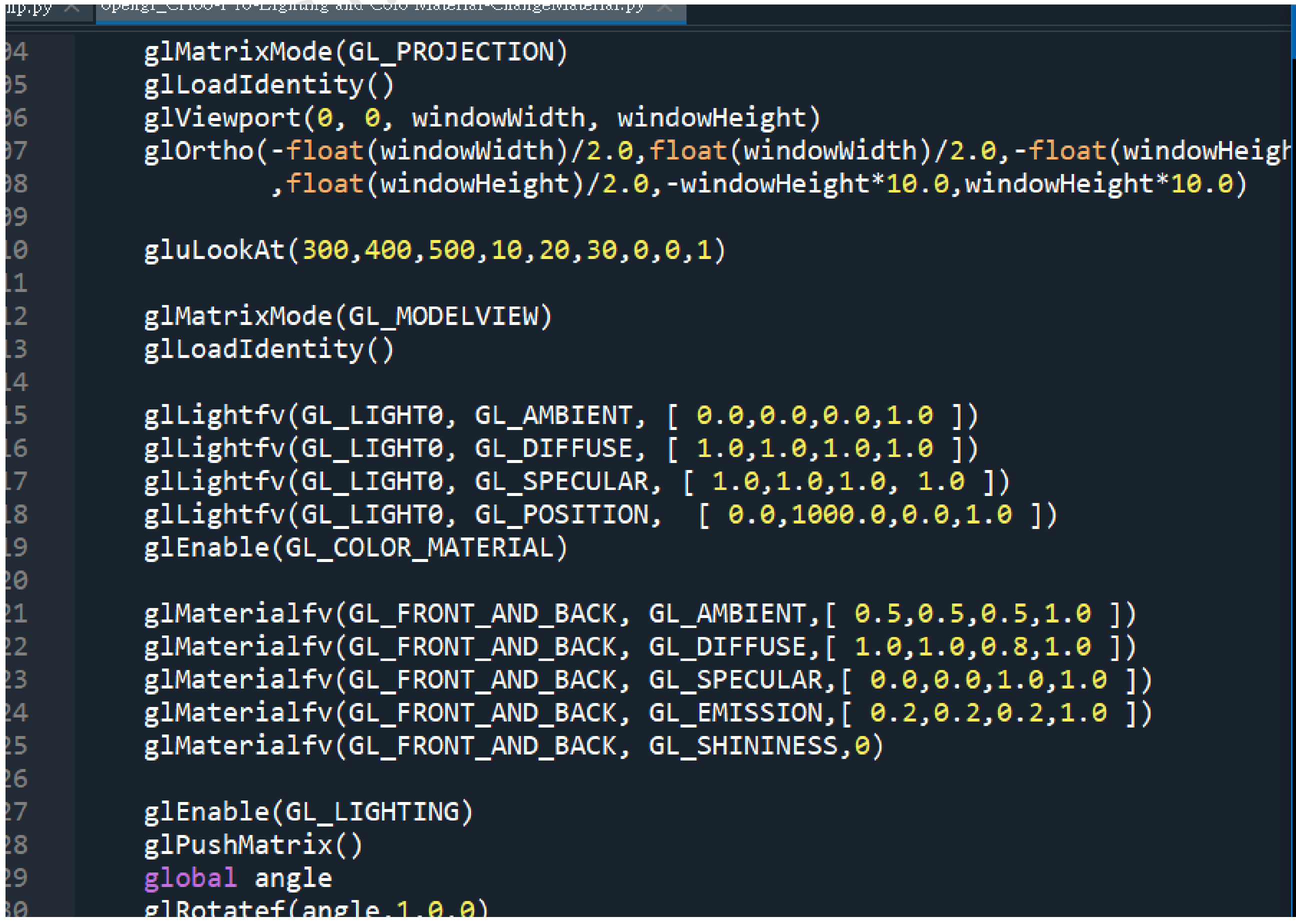
Change properties of LIGHT



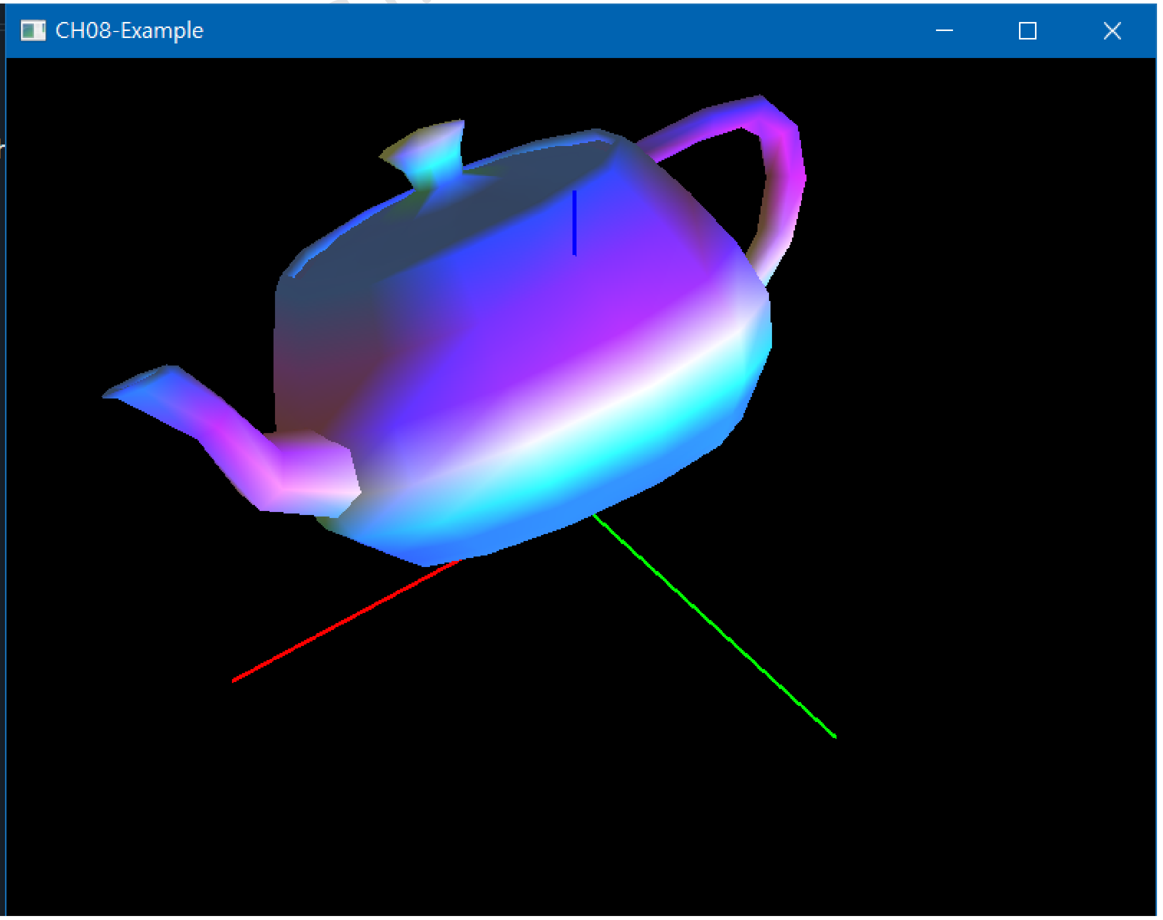
```
def display():
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glViewport(0, 0, windowWidth, windowHeight)
    glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.0,-float(windowHeight)/2.0
            ,float(windowHeight)/2.0,-windowHeight*10.0,windowHeight*10.0)

    gluLookAt(300,400,500,10,20,30,0,0,1)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    glLightfv(GL_LIGHT0, GL_AMBIENT, [ 0.3,0.3,0.3,1.0 ])
    glLightfv(GL_LIGHT0, GL_DIFFUSE, [ 1.0,0.0,0.0,1.0 ])
    glLightfv(GL_LIGHT0, GL_SPECULAR, [ 0.0,1.0,0.0, 1.0 ])
    glLightfv(GL_LIGHT0, GL_POSITION,  [ 0.0,1000.0,0.0,1.0 ])
    glEnable(GL_COLOR_MATERIAL)

    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT,[ 0.2,0.2,0.2,1.0 ])
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE,[ 0.8,0.8,0.8,1.0 ])
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR,[ 0.0,0.0,0.0,1.0 ])
    glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION,[ 0.0,0.0,0.0,1.0 ])
    glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS,0)

    glEnable(GL_LIGHTING)
    glPushMatrix()
    global angle
    glRotatef(angle,1,0,0)
    angle = angle + 0.5
```



Change properties of Material



```
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glViewport(0, 0, windowWidth, windowHeight)
    glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.0,-float(windowHeigh
            ,float(windowHeight)/2.0,-windowHeight*10.0,windowHeight*10.0)

    gluLookAt(300,400,500,10,20,30,0,0,1)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    glLightfv(GL_LIGHT0, GL_AMBIENT, [ 0.0,0.0,0.0,1.0 ])
    glLightfv(GL_LIGHT0, GL_DIFFUSE, [ 1.0,1.0,1.0,1.0 ])
    glLightfv(GL_LIGHT0, GL_SPECULAR, [ 1.0,1.0,1.0, 1.0 ])
    glLightfv(GL_LIGHT0, GL_POSITION,  [ 0.0,1000.0,0.0,1.0 ])
    glEnable(GL_COLOR_MATERIAL)

    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT,[ 0.5,0.5,0.5,1.0 ])
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE,[ 1.0,1.0,0.8,1.0 ])
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR,[ 0.0,0.0,1.0,1.0 ])
    glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION,[ 0.2,0.2,0.2,1.0 ])
    glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS,0)

    glEnable(GL_LIGHTING)
    glPushMatrix()
    global angle
    glRotatef(angle,1,0,0)
```



Keyboard / Mouse / trackball control

Keyboard / mouse

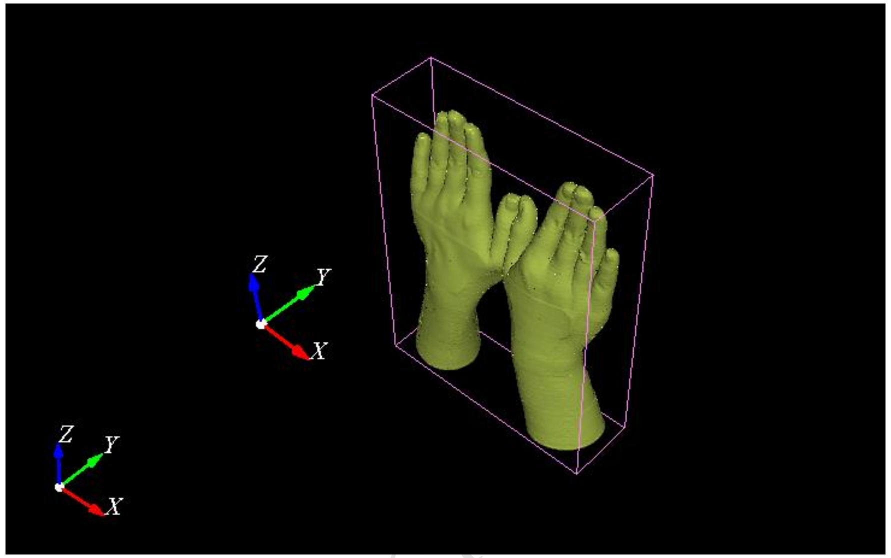
trackball

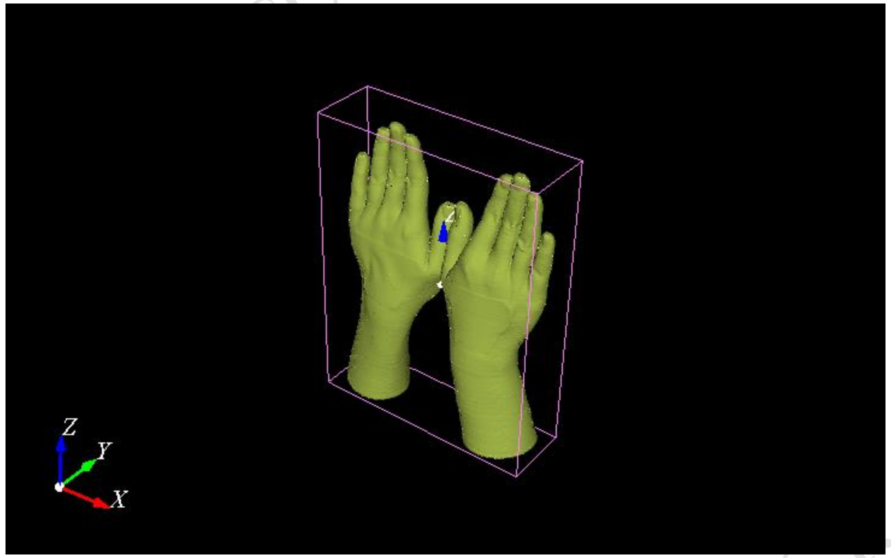
6D pen

3D mouse

joystick

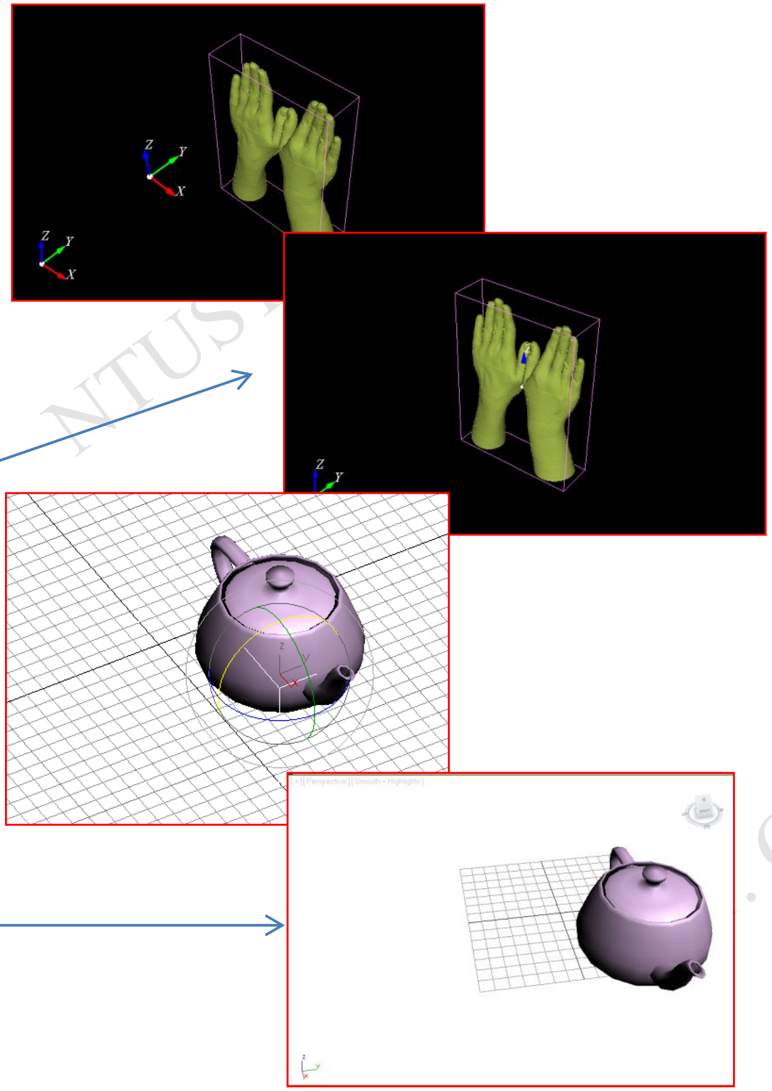
Define your rule and strategy





Trackerball

- 1. 相機固定位置
  2. 旋轉物體(針對原點)
- 1. 相機位置固定
  2. 移動物體到原點
  3. 旋轉物體(針對原點)
  SDI
- 1. 相機位置固定
  2. 偵測物體的中心
  3. 針對物體中心旋轉
- 1. 相機位置與方向針對觀看物體改變

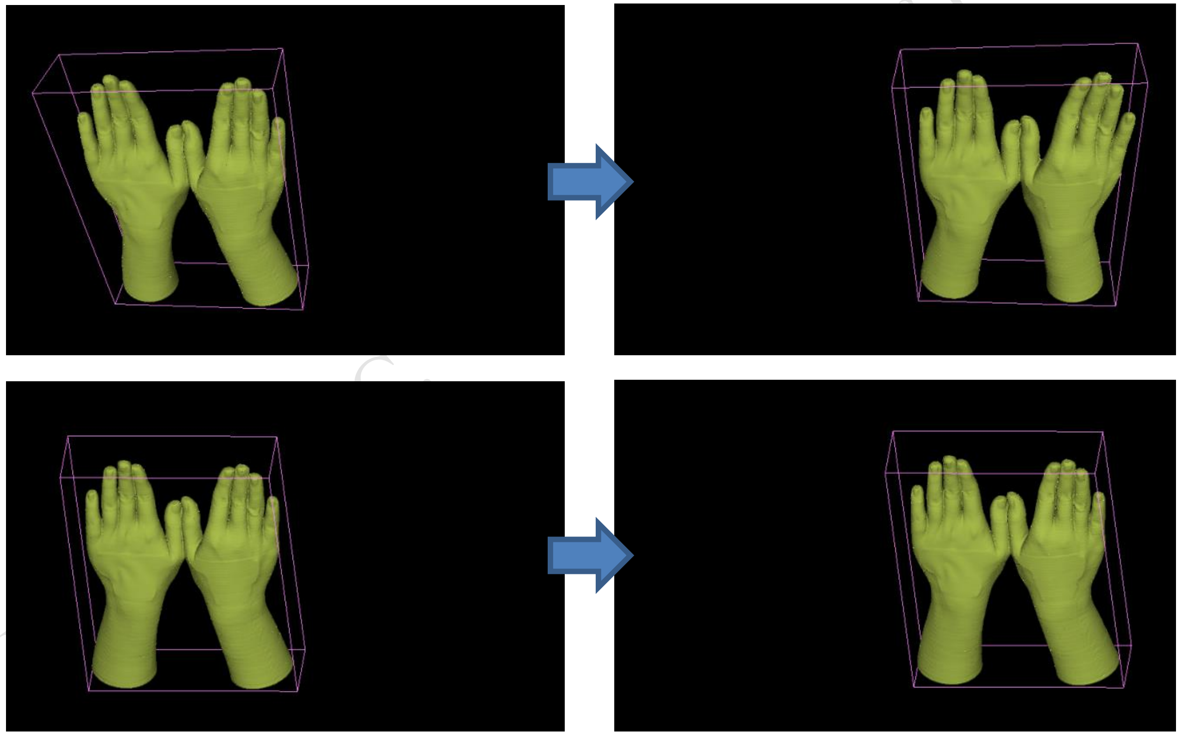


平移效果

1. 相機固定位置
2. 平移物體(針對原點)
3. 或平移物體到原點再平移

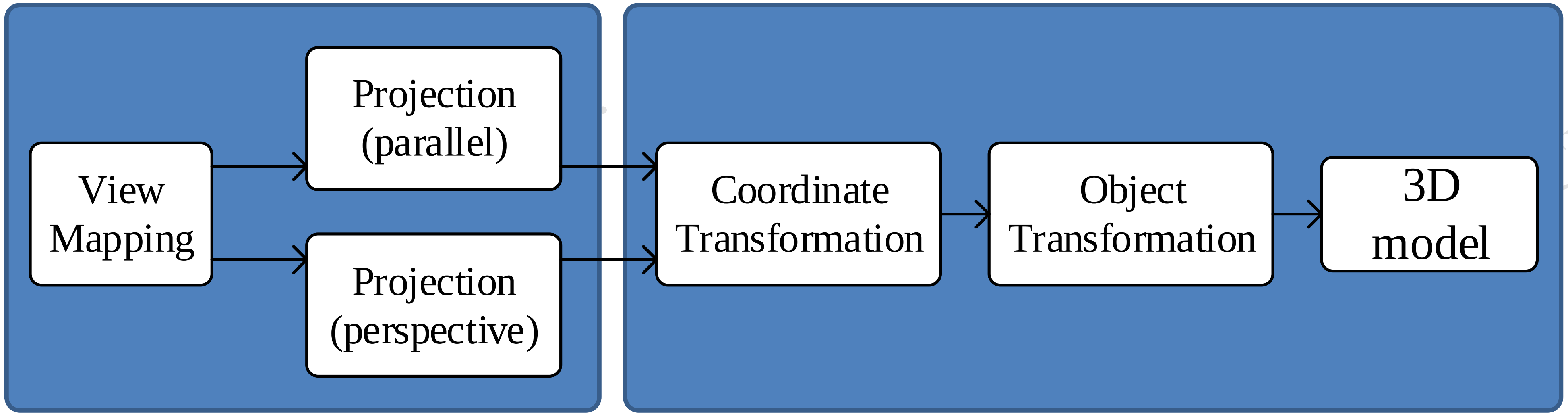
1. 根據相機UV方向移動(相機位置改變)
2. 物體保持不動

平行投影(滑鼠移動比例固定)
透視投影(滑鼠移動比例問題)

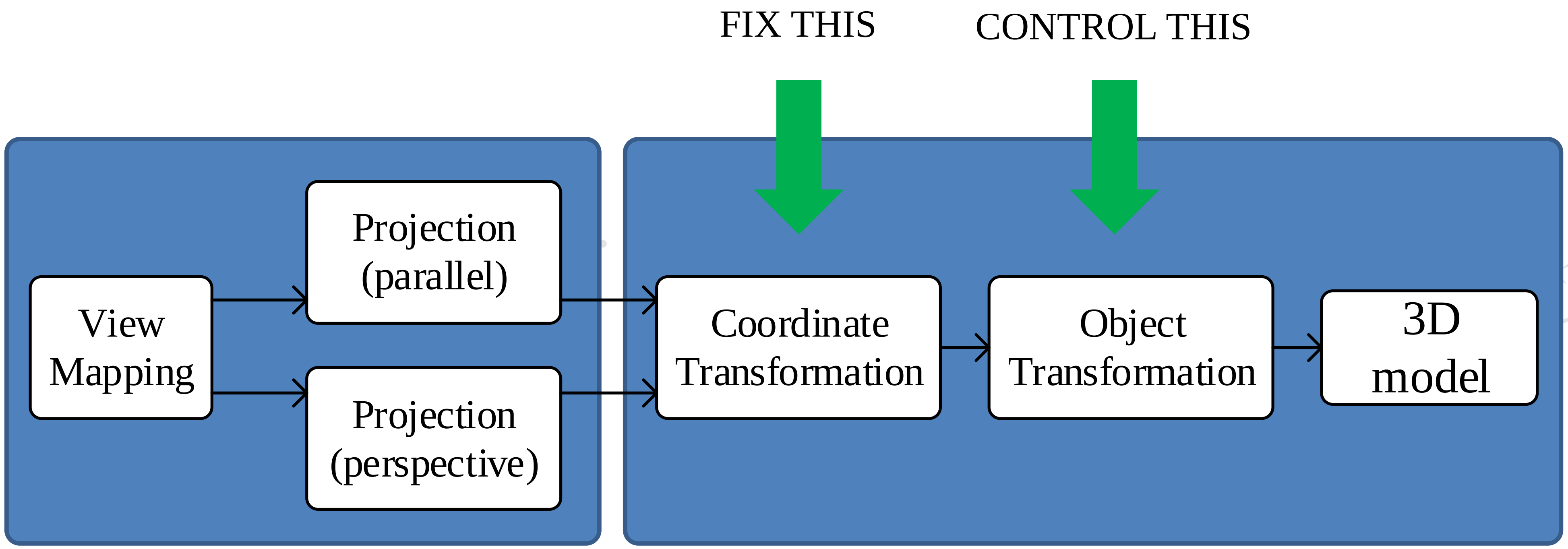


What you control in pipeline?

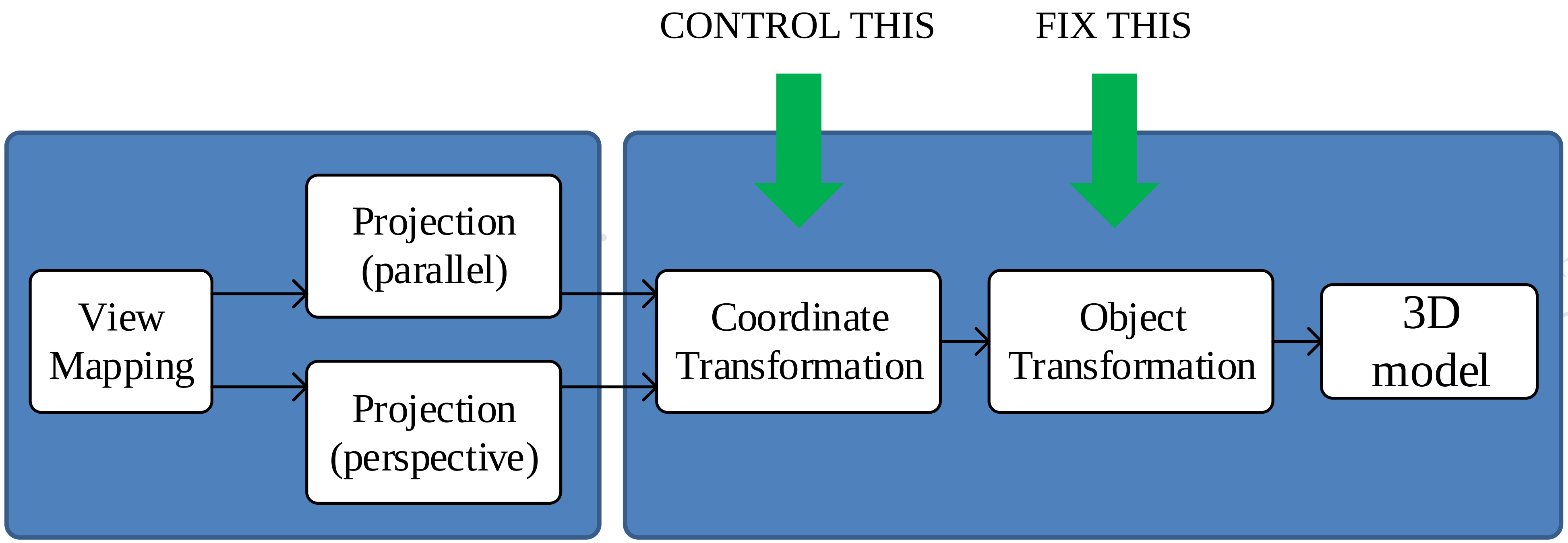
- To know the bounding box of object (data range)
- To control camera? (coordinate transformation)
- To control object? (object transformation)



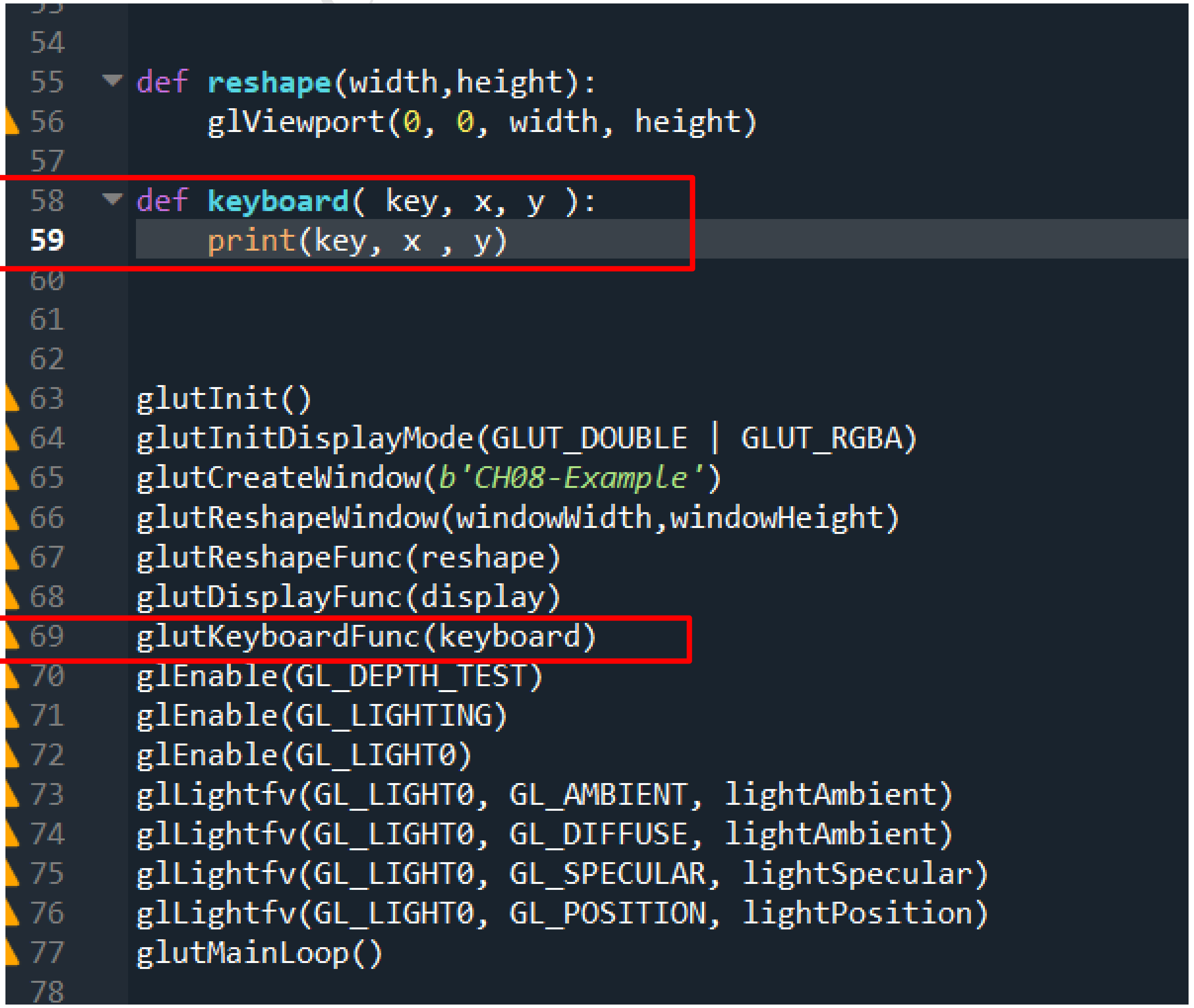
Control Object transformation



Control "Camera" (coordinate transformation)

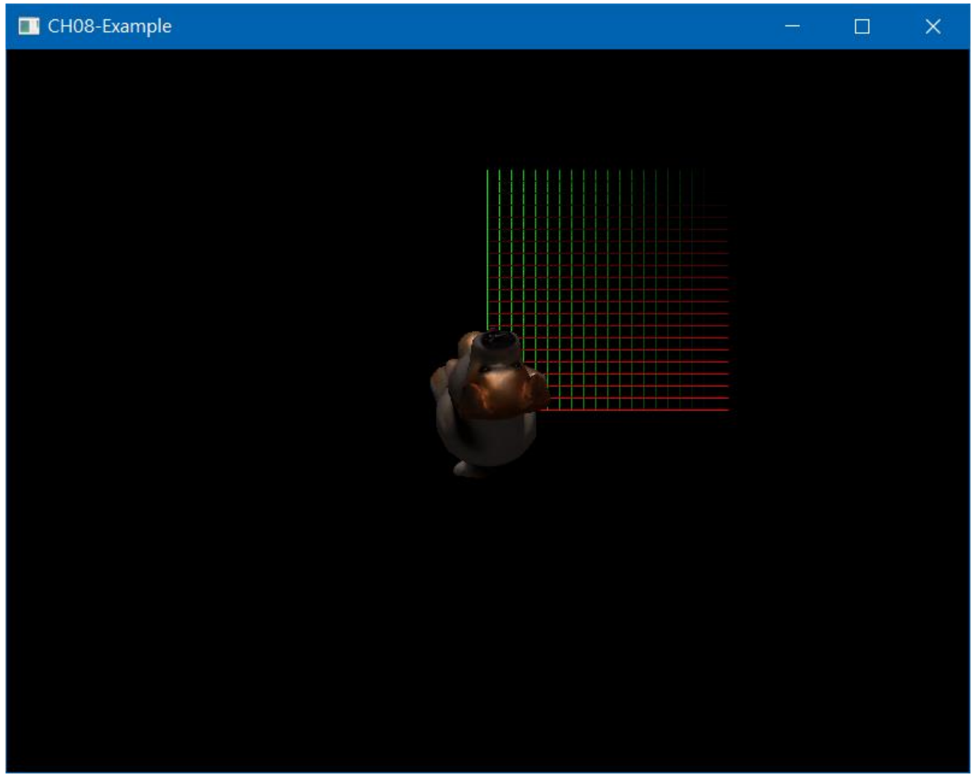


Keyboard: Check "value" of key in your device

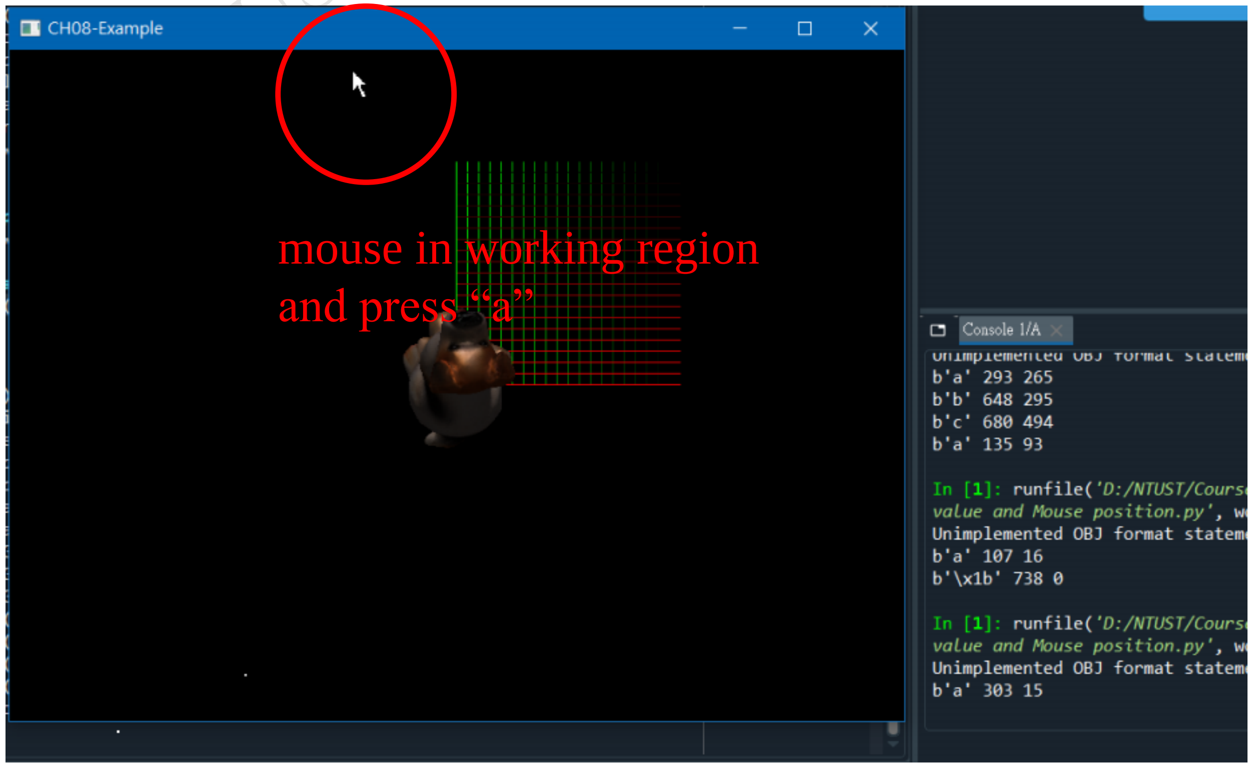


```
def reshape(width,height):
    glViewport(0, 0, width, height)

def keyboard( key, x, y ):
    print(key, x , y)



glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA)
glutCreateWindow(b'CH08-Example')
glutReshapeWindow(windowWidth,windowHeight)
glutReshapeFunc(reshape)
glutDisplayFunc(display)
glutKeyboardFunc(keyboard)
glEnable(GL_DEPTH_TEST)
glEnable(GL_LIGHTING)
glEnable(GL_LIGHT0)
glLightfv(GL_LIGHT0, GL_AMBIENT, lightAmbient)
glLightfv(GL_LIGHT0, GL_DIFFUSE, lightAmbient)
glLightfv(GL_LIGHT0, GL_SPECULAR, lightSpecular)
glLightfv(GL_LIGHT0, GL_POSITION, lightPosition)
glutMainLoop()
```

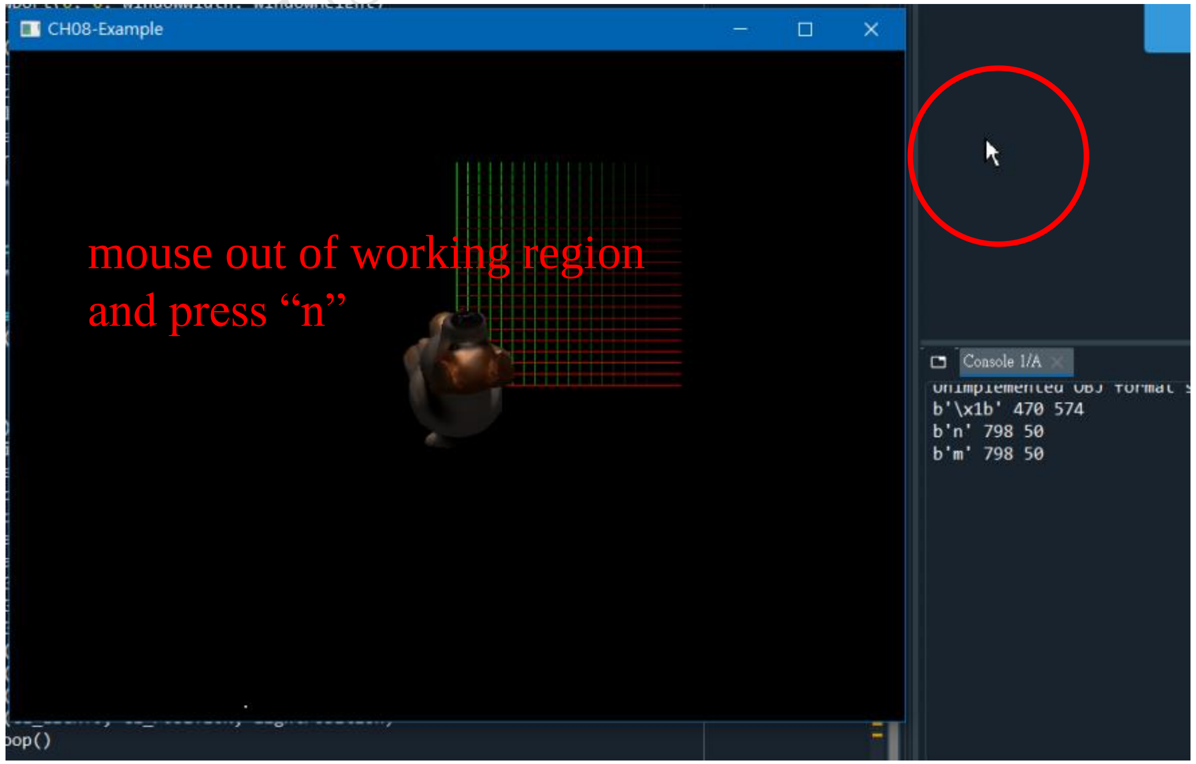
Make sure the dialog is ACTIVE, thus
"key" of keyboard will be recognized by your program



Keyboard: Check “value” of key in your device

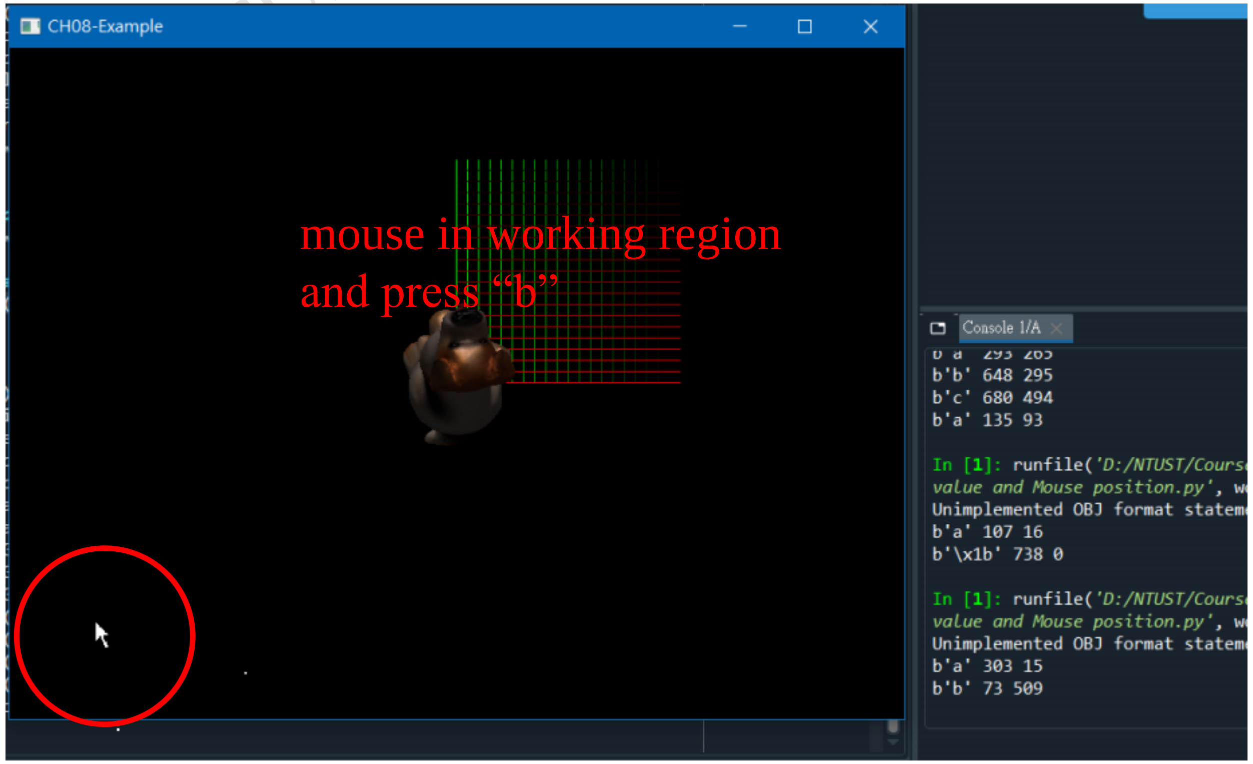


```
Unimplemented OBJ format statem
b'a' 293 265
b'b' 648 295
b'c' 680 494
b'a' 135 93

In [1]: runfile('D:/NTUST/Cours
value and Mouse position.py', w
Unimplemented OBJ format statem
b'a' 107 16
b'\x1b' 738 0

In [1]: runfile('D:/NTUST/Cours
value and Mouse position.py', w
Unimplemented OBJ format statem
b'a' 303 15
```

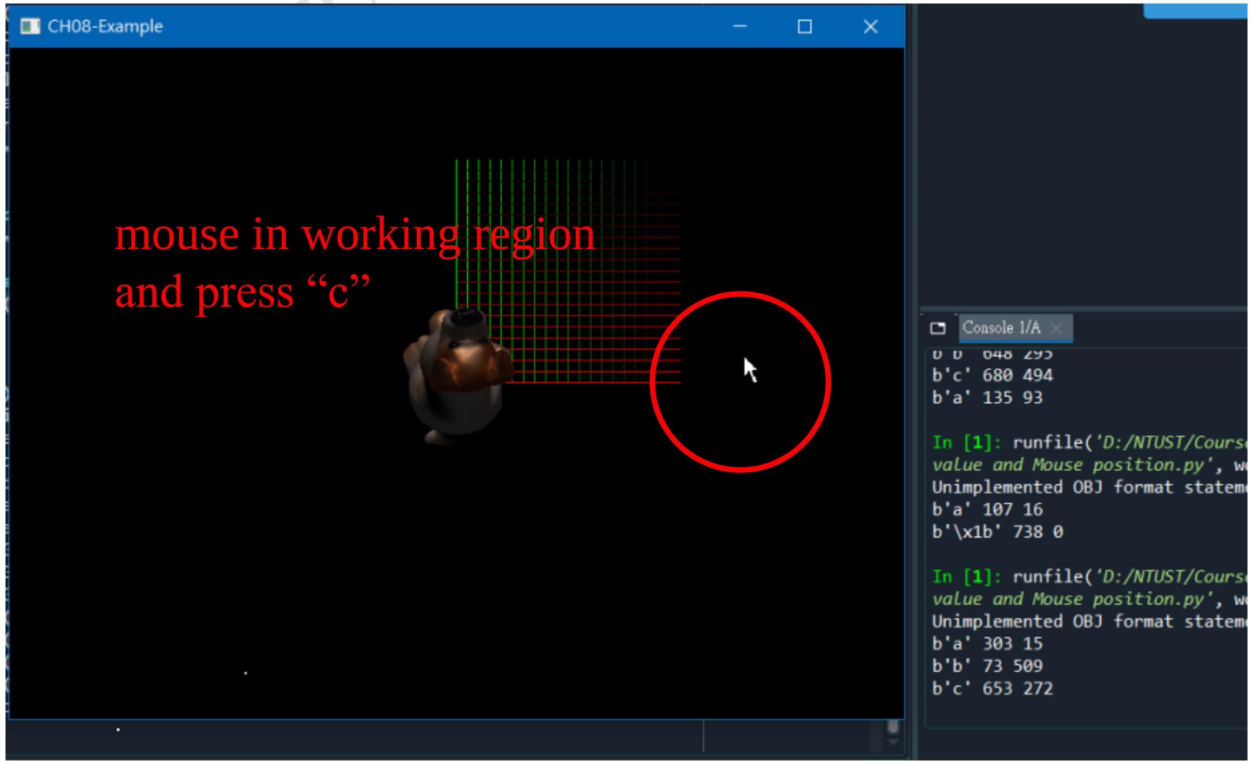


```
Unimplemented OBJ format
b'\x1b' 470 574
b'n' 798 50
b'm' 798 50
```

Keyboard: Check "value" of key in your device

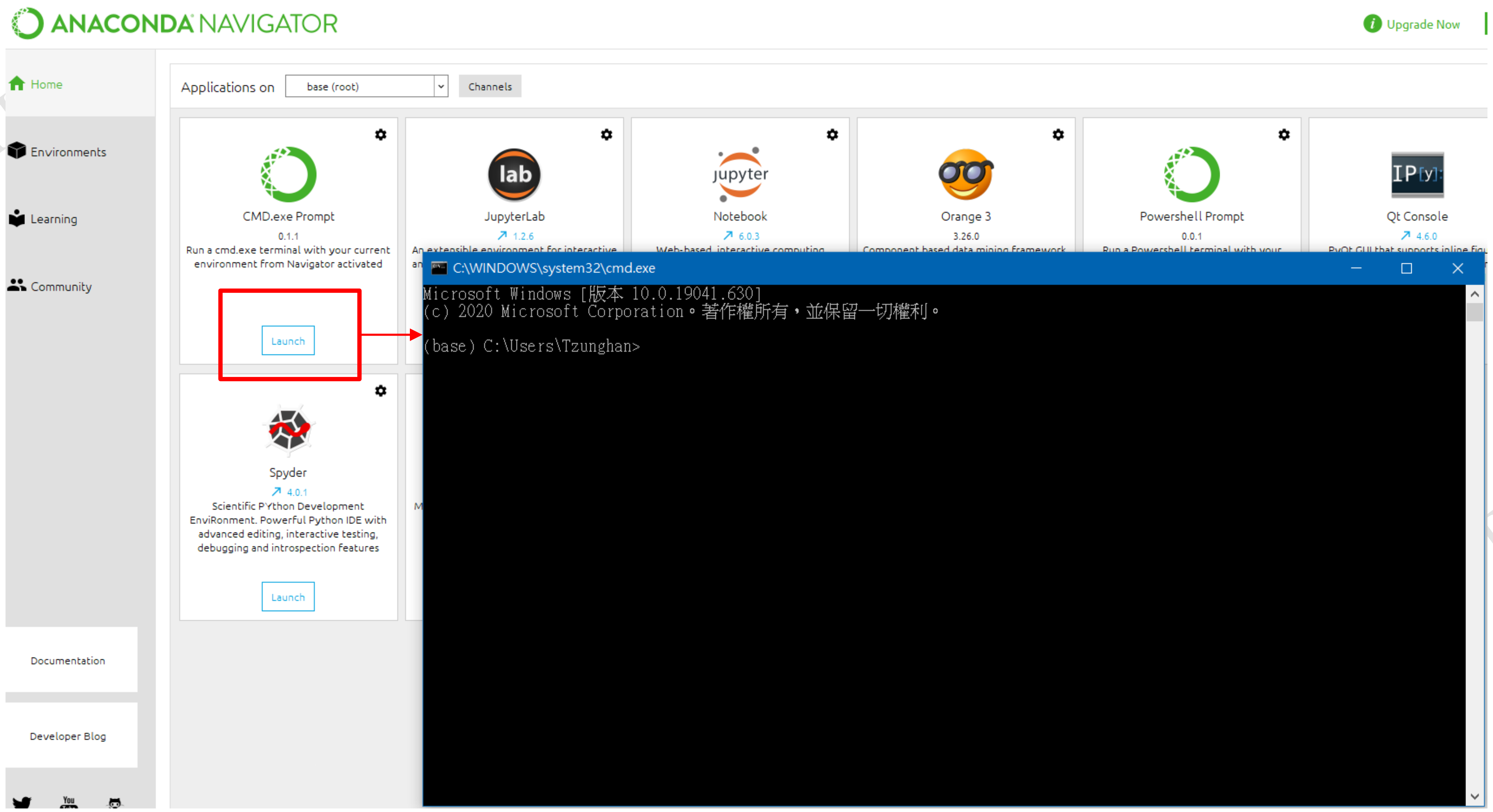


```
b'a' 293 265
b'b' 648 295
b'c' 680 494
b'a' 135 93

In [1]: runfile('D:/NTUST/Cours
value and Mouse position.py', w
Unimplemented OBJ format statem
b'a' 107 16
b'\x1b' 738 0

In [1]: runfile('D:/NTUST/Cours
value and Mouse position.py', w
Unimplemented OBJ format statem
b'a' 303 15
b'b' 73 509
```



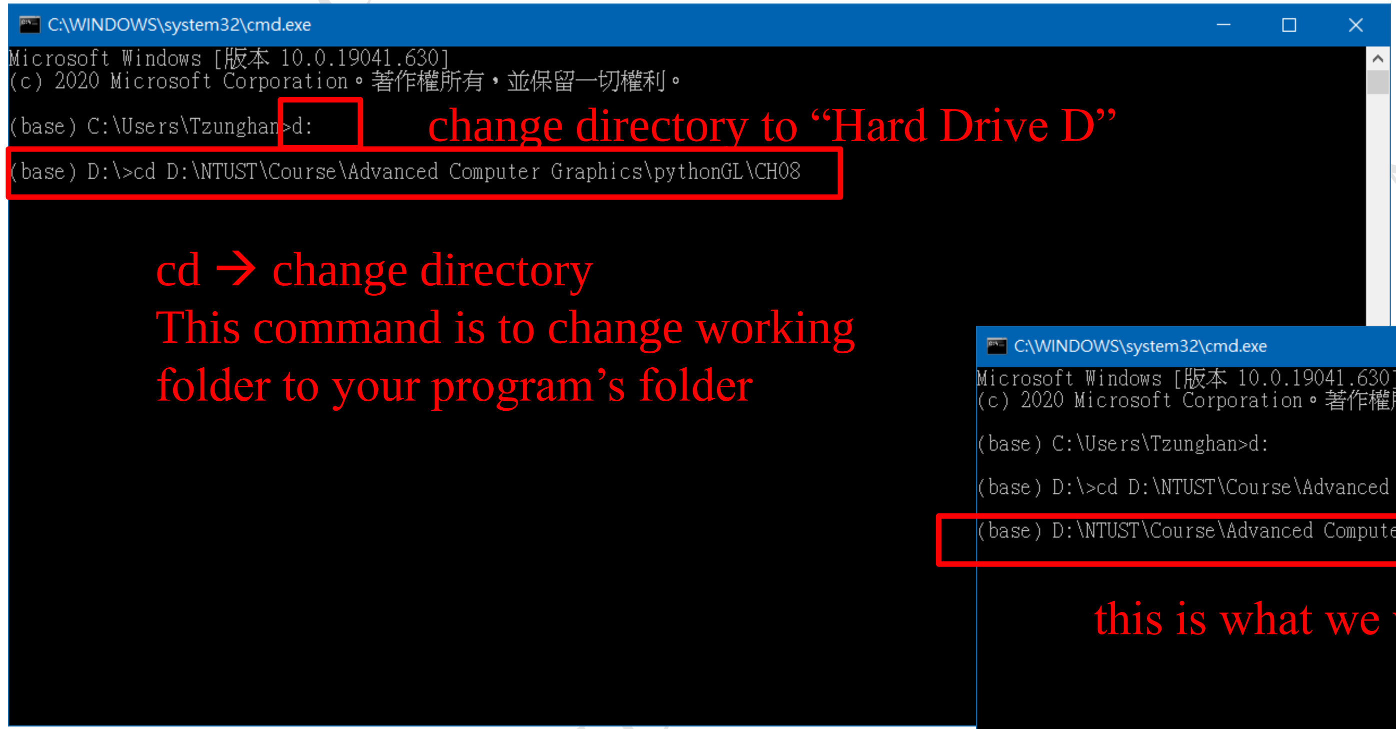
```
b'b' 648 295
b'c' 680 494
b'a' 135 93

In [1]: runfile('D:/NTUST/Cours
value and Mouse position.py', w
Unimplemented OBJ format statem
b'a' 107 16
b'\x1b' 738 0

In [1]: runfile('D:/NTUST/Cours
value and Mouse position.py', w
Unimplemented OBJ format statem
b'a' 303 15
b'b' 73 509
b'c' 653 272
```

Launch console mode and execute your program


ANACONDA NAVIGATOR
Upgrade Now
Home
Environments
Learning
Community
Documentation
Developer Blog
Applications on
base (root)
Channels
CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated
Launch
JupyterLab
1.2.6
Notebook
6.0.3
Orange 3
3.26.0
Powershell Prompt
0.0.1
Qt Console
4.6.0
Spyder
4.0.1
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch
C:\WINDOWS\system32\cmd.exe

```
Microsoft Windows [版本 10.0.19041.630]
(c) 2020 Microsoft Corporation。著作權所有，並保留一切權利。

(base) C:\Users\Tzunghan>
```

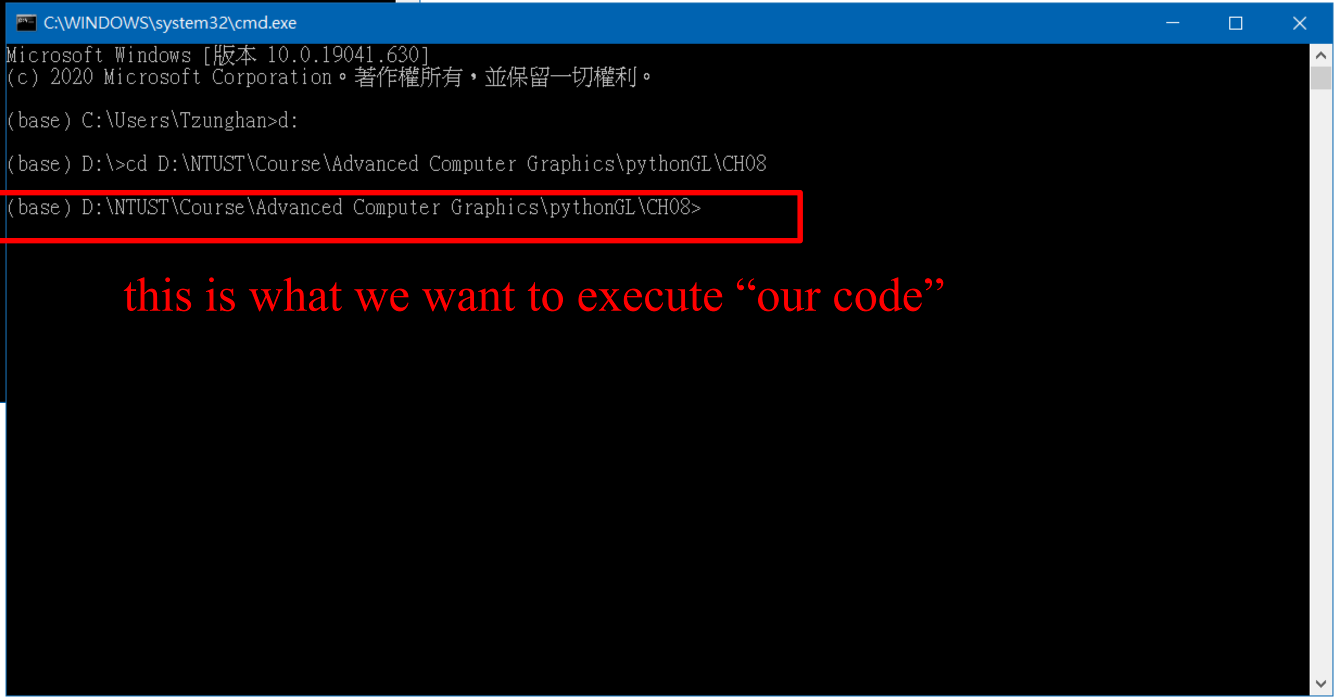
Launch console mode and execute your program



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.19041.630]
(c) 2020 Microsoft Corporation。著作權所有，並保留一切權利。

(base) C:\Users\Tzunghan>d:

(base) D:\>cd D:\NTUST\Course\Advanced Computer Graphics\pythonGL\CH08
```

change directory to “Hard Drive D”

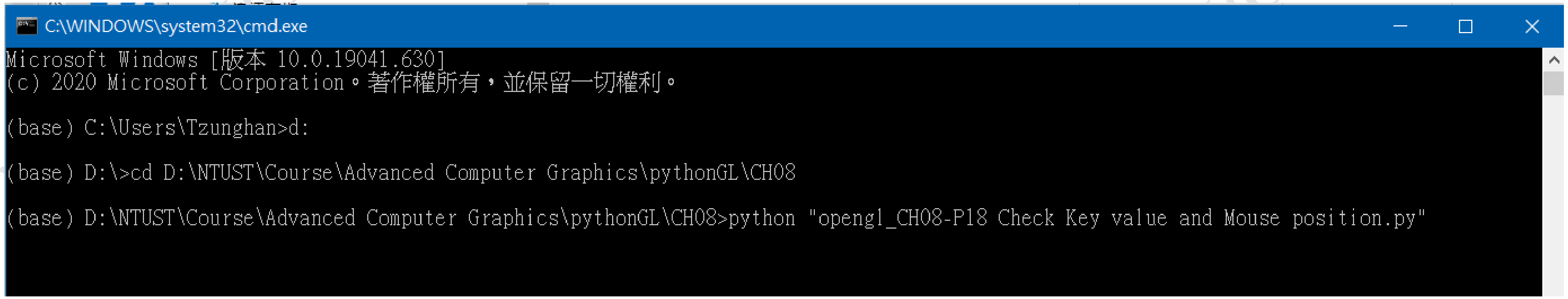
cd → change directory
This command is to change working folder to your program’s folder



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.19041.630]
(c) 2020 Microsoft Corporation。著作權所有，並保留一切權利。

(base) C:\Users\Tzunghan>d:

(base) D:\>cd D:\NTUST\Course\Advanced Computer Graphics\pythonGL\CH08

(base) D:\NTUST\Course\Advanced Computer Graphics\pythonGL\CH08>
```

this is what we want to execute “our code”

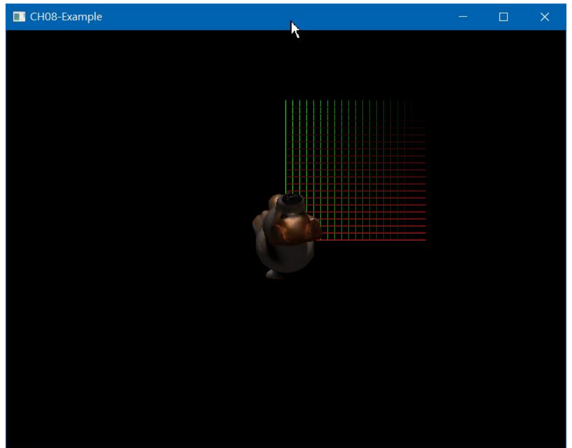
Launch console mode and execute your program


C:\WINDOWS\system32\cmd.exe

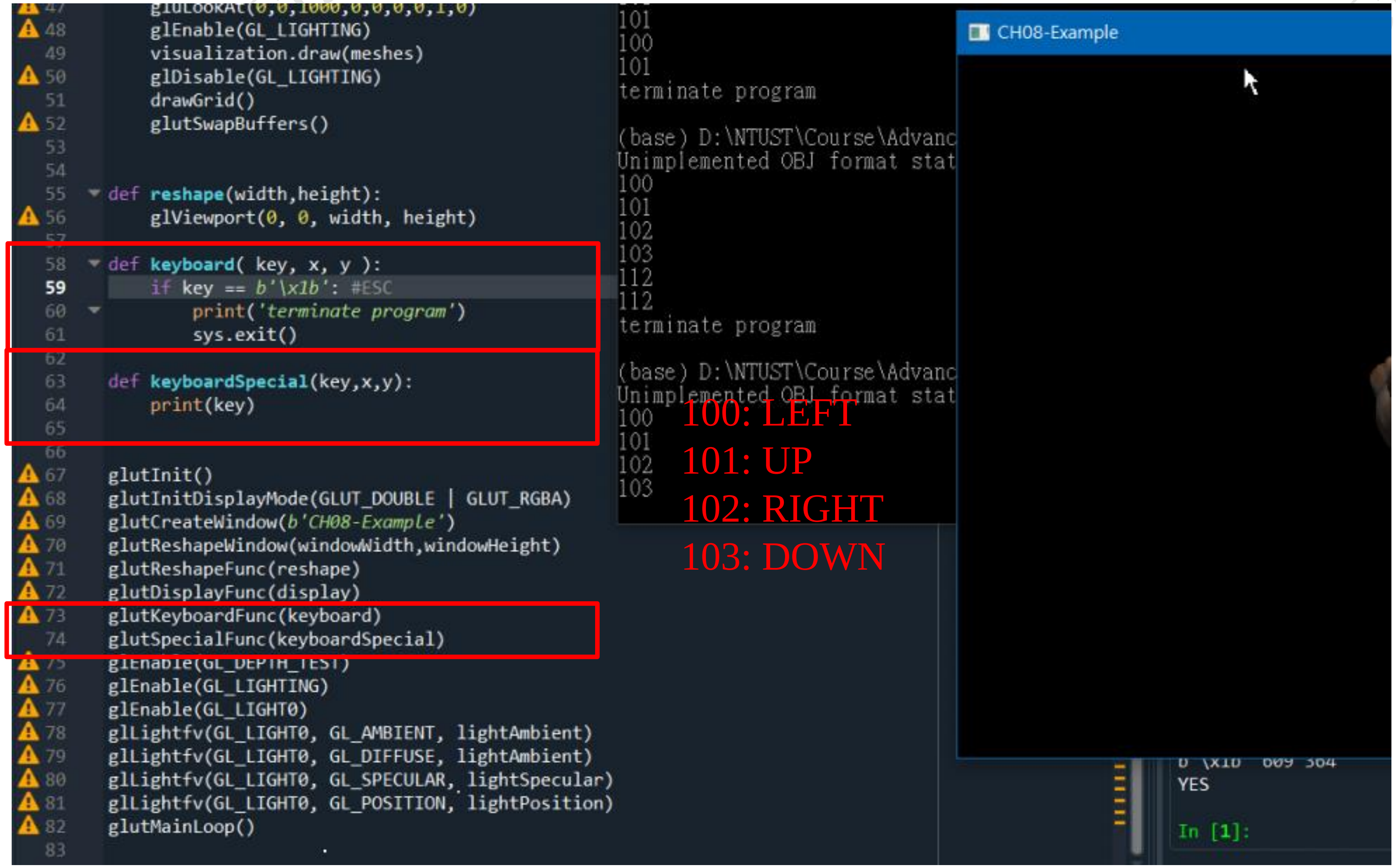
```
Microsoft Windows [版本 10.0.19041.630]
(c) 2020 Microsoft Corporation。著作權所有，並保留一切權利。

(base) C:\Users\Tzunghan>d:

(base) D:\>cd D:\NTUST\Course\Advanced Computer Graphics\pythonGL\CH08

(base) D:\NTUST\Course\Advanced Computer Graphics\pythonGL\CH08>python "opengl_CH08-P18 Check Key value and Mouse position.py"
```

- to run your program type in the following line, then enter

python "opengl_CH08-P18 Check Key value and Mouse position.py"



Arrow key (as special key)



```
    gluLookAt(0,0,1000,0,0,0,0,1,0)
    glEnable(GL_LIGHTING)
    visualization.draw(meshes)
    glDisable(GL_LIGHTING)
    drawGrid()
    glutSwapBuffers()


def reshape(width,height):
    glViewport(0, 0, width, height)

def keyboard( key, x, y ):
    if key == b'\x1b': #ESC
        print('terminate program')
        sys.exit()

def keyboardSpecial(key,x,y):
    print(key)


glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA)
glutCreateWindow(b'CH08-Example')
glutReshapeWindow(windowWidth,windowHeight)
glutReshapeFunc(reshape)
glutDisplayFunc(display)
glutKeyboardFunc(keyboard)
glutSpecialFunc(keyboardSpecial)
glEnable(GL_DEPTH_TEST)
glEnable(GL_LIGHTING)
glEnable(GL_LIGHT0)
glLightfv(GL_LIGHT0, GL_AMBIENT, lightAmbient)
glLightfv(GL_LIGHT0, GL_DIFFUSE, lightAmbient)
glLightfv(GL_LIGHT0, GL_SPECULAR, lightSpecular)
glLightfv(GL_LIGHT0, GL_POSITION, lightPosition)
glutMainLoop()
```

```
101
100
101
terminate program

(base) D:\NTUST\Course\Advanc
Unimplemented OBJ format stat
100
101
102
103
112
112
terminate program

(base) D:\NTUST\Course\Advanc
Unimplemented OBJ format stat
100
101
102
103
```

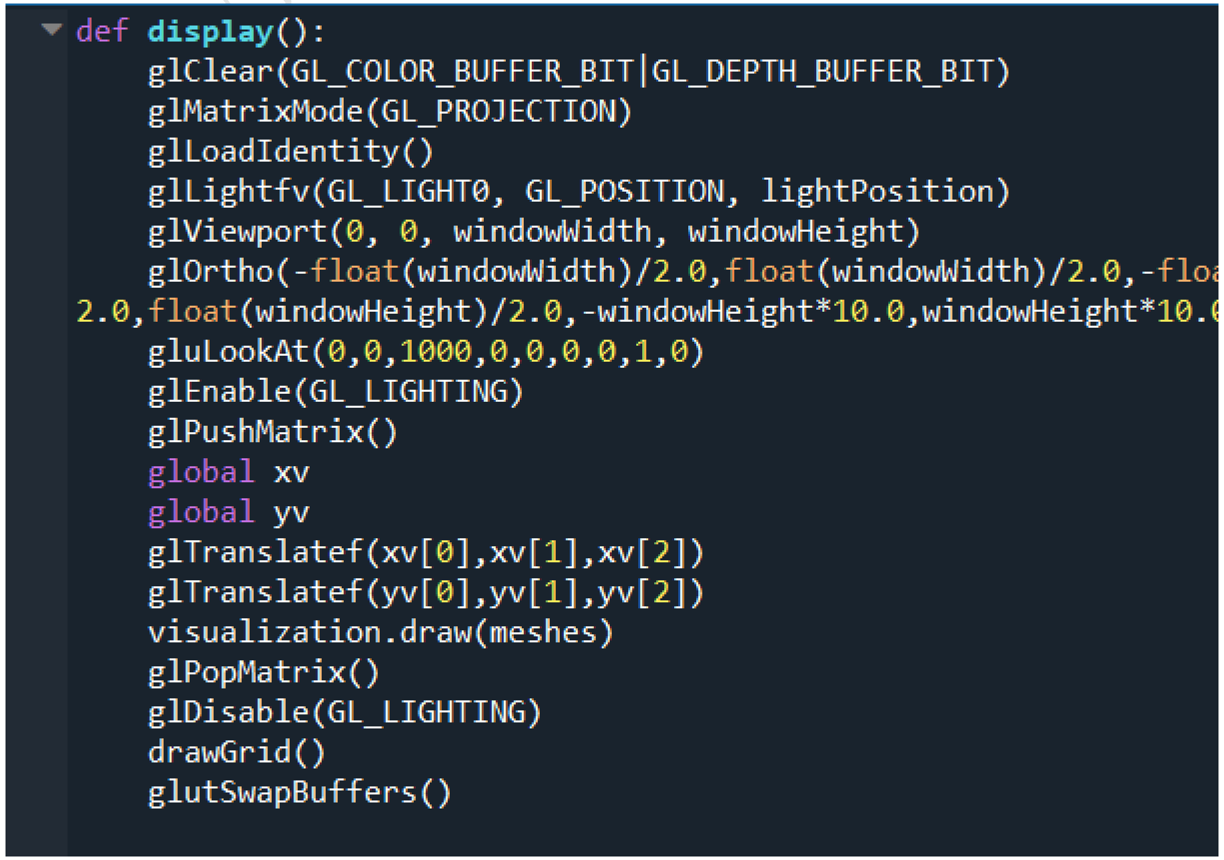
CH08-Example

```
b'\x1b' 609 364
YES

In [1]:
```

100: LEFT
101: UP
102: RIGHT
103: DOWN

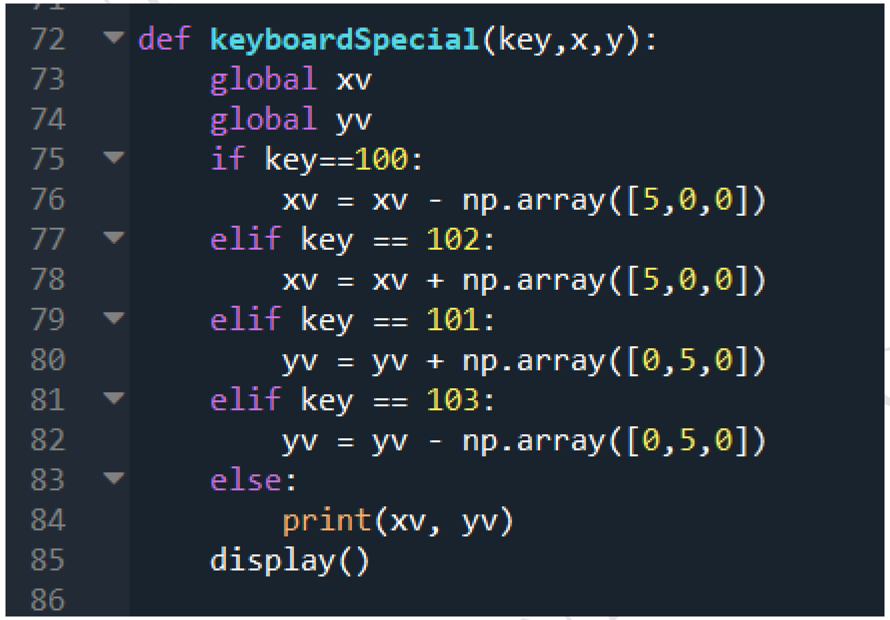
Translate Objects (by arrow key)

Apply vectors on Object



```
def display():
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glLightfv(GL_LIGHT0, GL_POSITION, lightPosition)
    glViewport(0, 0, windowWidth, windowHeight)
    glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.0,-floa
2.0,float(windowHeight)/2.0,-windowHeight*10.0,windowHeight*10.0
    gluLookAt(0,0,1000,0,0,0,0,1,0)
    glEnable(GL_LIGHTING)
    glPushMatrix()
    global xv
    global yv
    glTranslatef(xv[0],xv[1],xv[2])
    glTranslatef(yv[0],yv[1],yv[2])
    visualization.draw(meshes)
    glPopMatrix()
    glDisable(GL_LIGHTING)
    drawGrid()
    glutSwapBuffers()
```

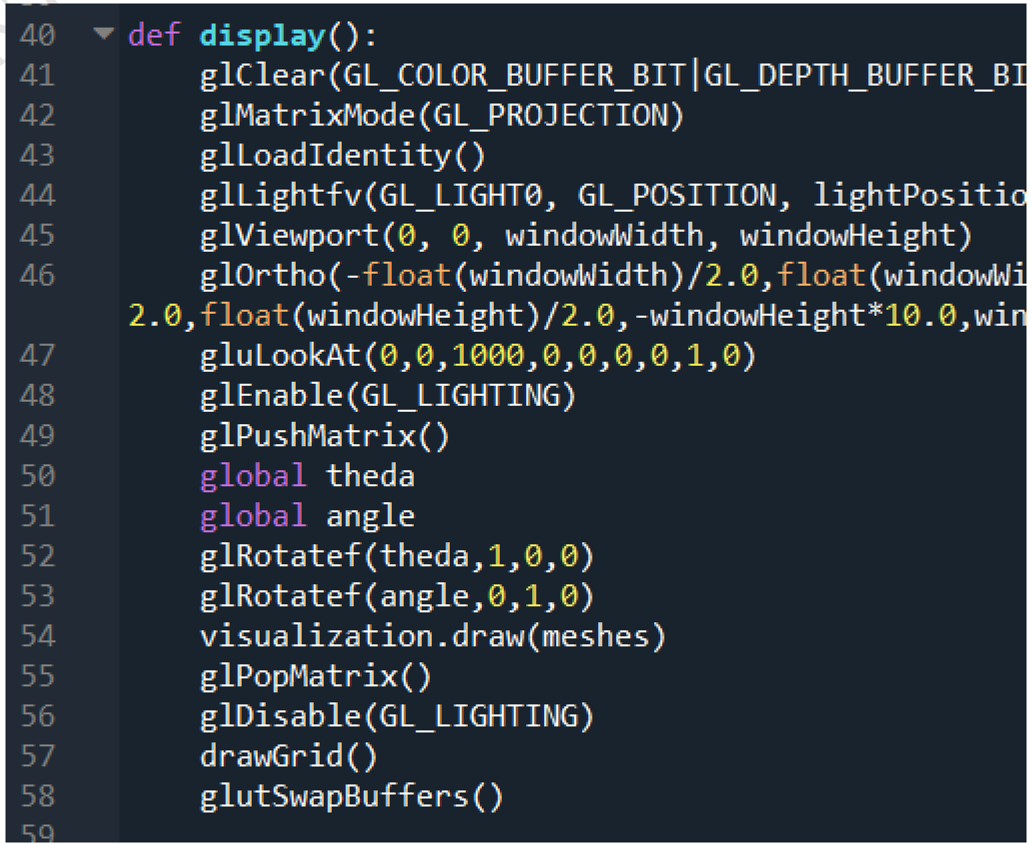
Change statuses of vectors in Keyboard



```
def keyboardSpecial(key,x,y):
    global xv
    global yv
    if key==100:
        xv = xv - np.array([5,0,0])
    elif key == 102:
        xv = xv + np.array([5,0,0])
    elif key == 101:
        yv = yv + np.array([0,5,0])
    elif key == 103:
        yv = yv - np.array([0,5,0])
    else:
        print(xv, yv)
    display()
```

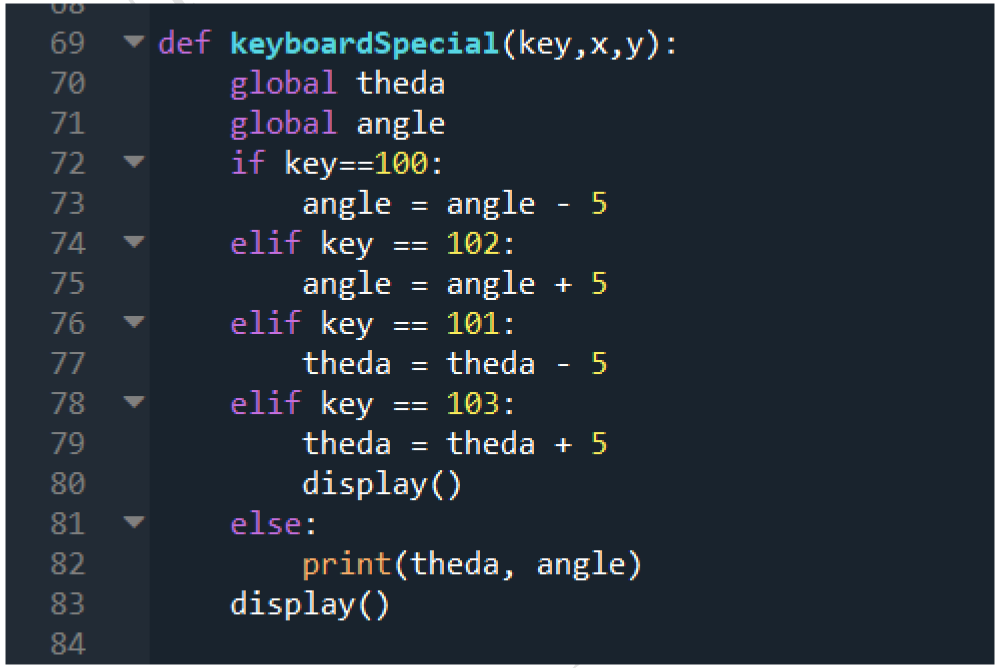
Rotate Objects (by arrow key)

Apply angles' values on Object



```
def display():
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BI
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glLightfv(GL_LIGHT0, GL_POSITION, lightPositio
    glViewport(0, 0, windowWidth, windowHeight)
    glOrtho(-float(windowWidth)/2.0,float(windowWi
2.0,float(windowHeight)/2.0,-windowHeight*10.0,win
    gluLookAt(0,0,1000,0,0,0,0,1,0)
    glEnable(GL_LIGHTING)
    glPushMatrix()
    global theda
    global angle
    glRotatef(theda,1,0,0)
    glRotatef(angle,0,1,0)
    visualization.draw(meshes)
    glPopMatrix()
    glDisable(GL_LIGHTING)
    drawGrid()
    glutSwapBuffers()
```

Change statuses of angles in Keyboard



```
def keyboardSpecial(key,x,y):
    global theda
    global angle
    if key==100:
        angle = angle - 5
    elif key == 102:
        angle = angle + 5
    elif key == 101:
        theda = theda - 5
    elif key == 103:
        theda = theda + 5
        display()
    else:
        print(theda, angle)
    display()
```

色彩與照明科技研究所
Graduate Institute of
Color and Illumination Technology

Find us on:
facebook