Advanced Computer Graphics

Lecture-08 Introduction to OpenGL-9

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

TAIWAN TECH National Taiwan University of Science and Technology

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

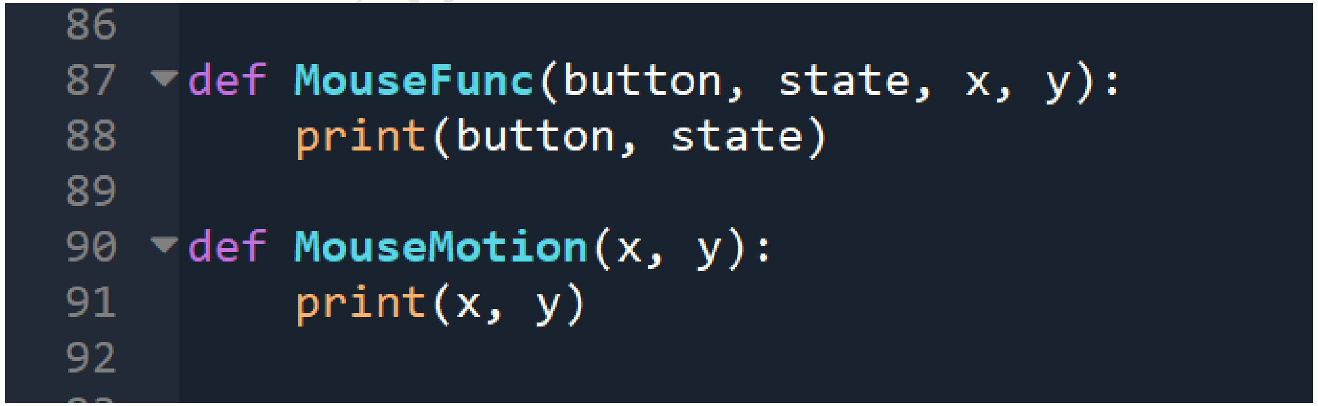
Mouse control

- Other similar device: Trackball / Joystick

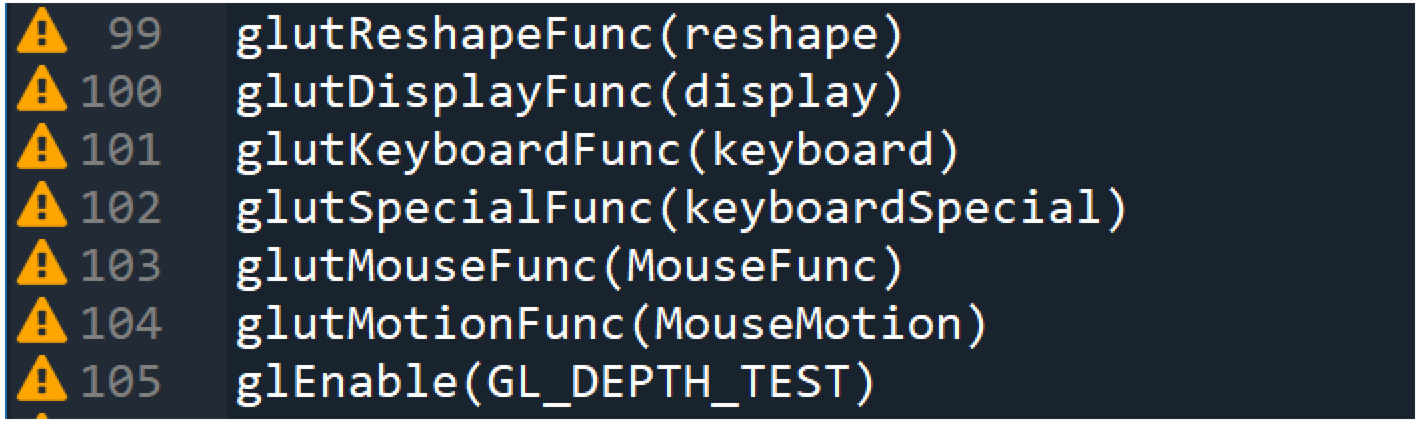
Mouse control

- glutMouseFunc(MouseFunc)
- glutMotionFunc(MouseMotion)

Mouse control (know the definition)

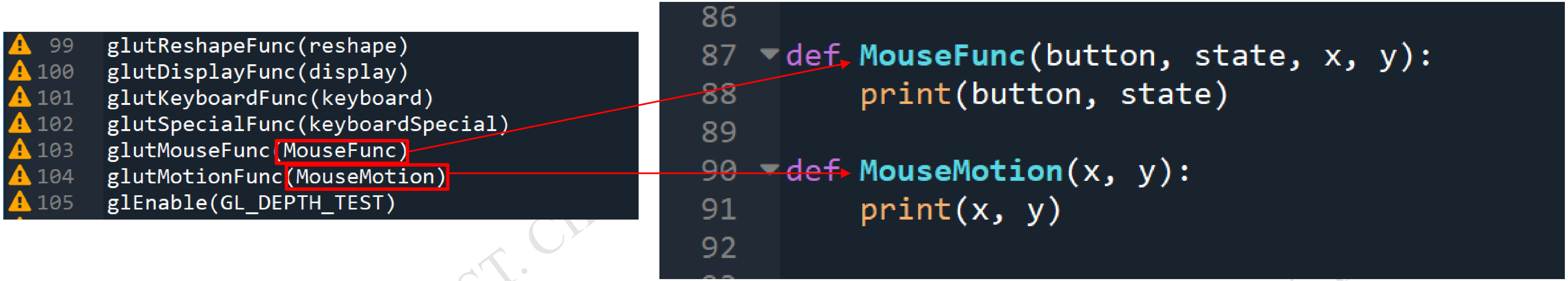


```python

def MouseFunc(button, state, x, y):
    print(button, state)

def MouseMotion(x, y):
    print(x, y)

```



```python
glutReshapeFunc(reshape)
glutDisplayFunc(display)
glutKeyboardFunc(keyboard)
glutSpecialFunc(keyboardSpecial)
glutMouseFunc(MouseFunc)
glutMotionFunc(MouseMotion)
glEnable(GL_DEPTH_TEST)
```

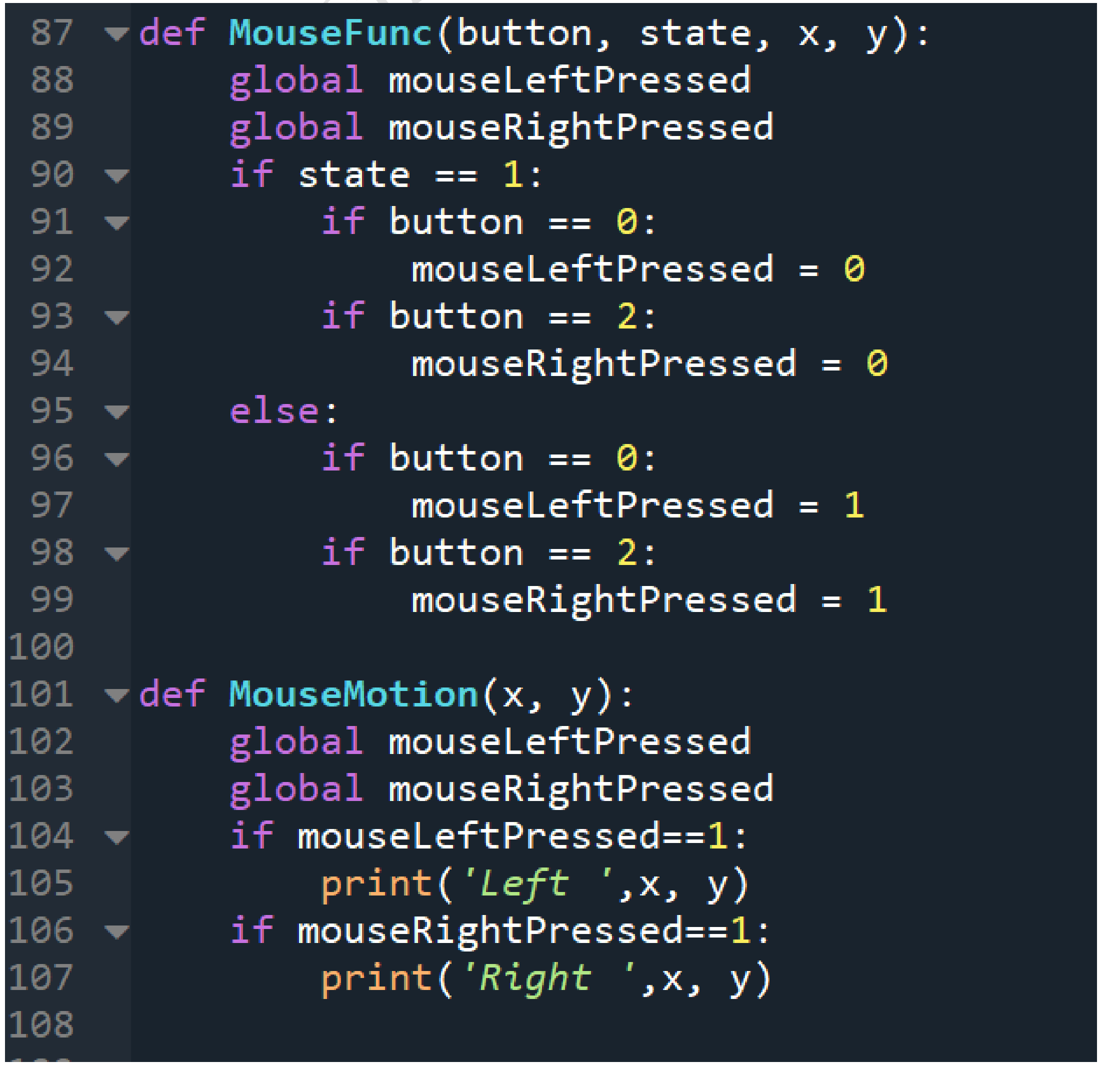
Mouse control (know the definition)



```
glutReshapeFunc(reshape)
glutDisplayFunc(display)
glutKeyboardFunc(keyboard)
glutSpecialFunc(keyboardSpecial)
glutMouseFunc(MouseFunc)
glutMotionFunc(MouseMotion)
glEnable(GL_DEPTH_TEST)
```

```
def MouseFunc(button, state, x, y):
    print(button, state)

def MouseMotion(x, y):
    print(x, y)

```

Mouse control (for translation and rotation)

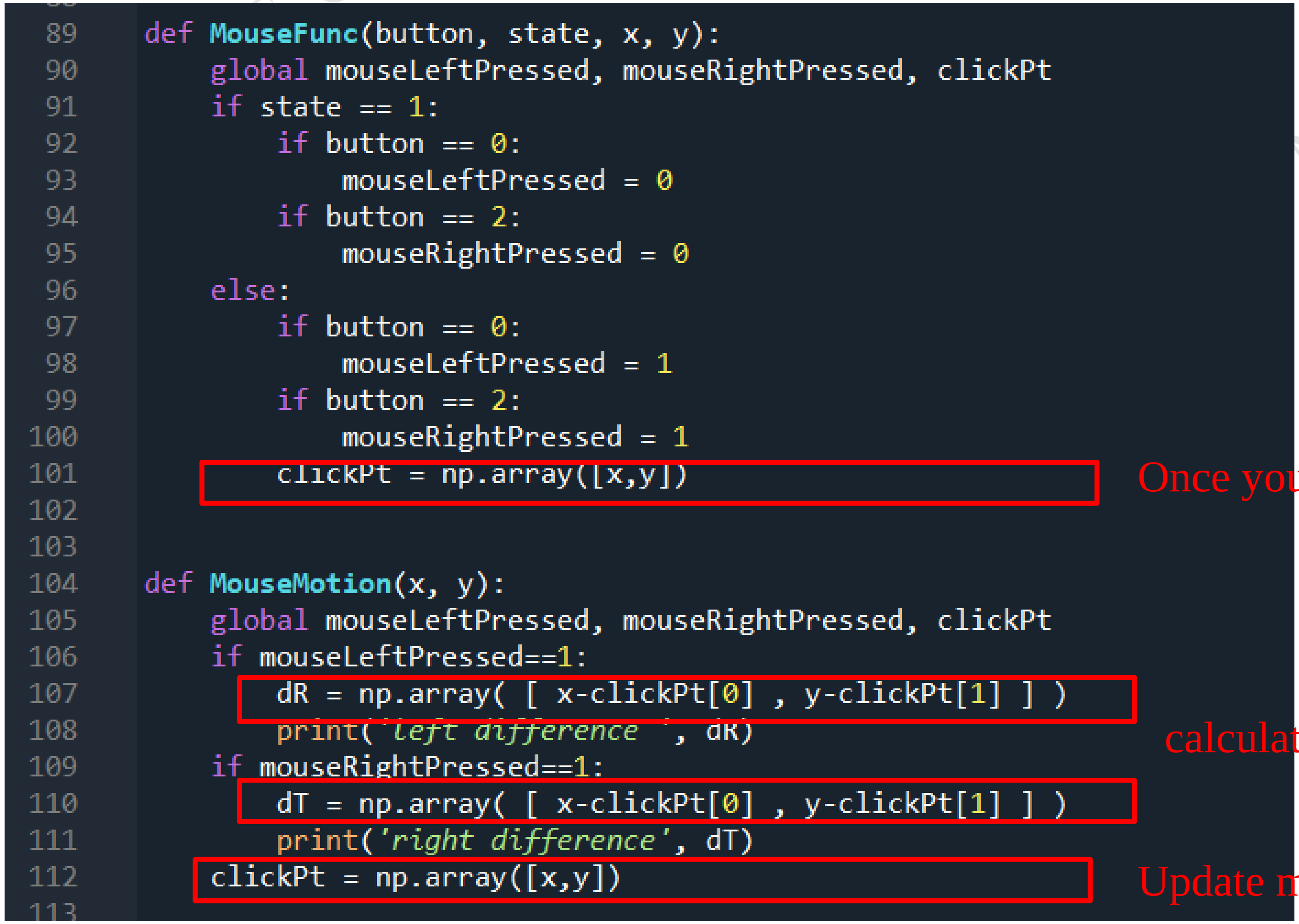


```
def MouseFunc(button, state, x, y):
    global mouseLeftPressed
    global mouseRightPressed
    if state == 1:
        if button == 0:
            mouseLeftPressed = 0
        if button == 2:
            mouseRightPressed = 0
    else:
        if button == 0:
            mouseLeftPressed = 1
        if button == 2:
            mouseRightPressed = 1

def MouseMotion(x, y):
    global mouseLeftPressed
    global mouseRightPressed
    if mouseLeftPressed==1:
        print('Left ',x, y)
    if mouseRightPressed==1:
        print('Right ',x, y)
```

Use global variables to store the status of “Press”

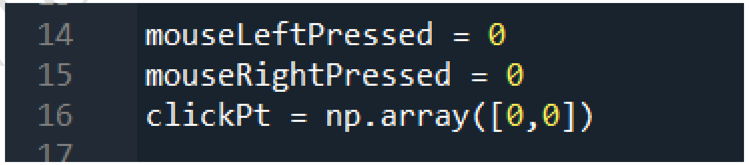
Mouse control (for translation and rotation)

Store the difference vector after clicking and dragging



```python
def MouseFunc(button, state, x, y):
    global mouseLeftPressed, mouseRightPressed, clickPt
    if state == 1:
        if button == 0:
            mouseLeftPressed = 0
        if button == 2:
            mouseRightPressed = 0
    else:
        if button == 0:
            mouseLeftPressed = 1
        if button == 2:
            mouseRightPressed = 1
        clickPt = np.array([x,y])


def MouseMotion(x, y):
    global mouseLeftPressed, mouseRightPressed, clickPt
    if mouseLeftPressed==1:
        dR = np.array( [ x-clickPt[0] , y-clickPt[1] ] )
        print('left difference', dR)
    if mouseRightPressed==1:
        dT = np.array( [ x-clickPt[0] , y-clickPt[1] ] )
        print('right difference', dT)
    clickPt = np.array([x,y])
```

global variables



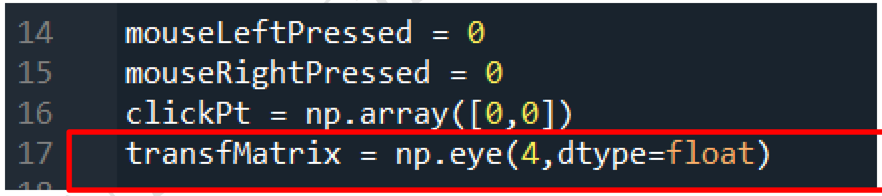
```python
mouseLeftPressed = 0
mouseRightPressed = 0
clickPt = np.array([0,0])
```

Once you click mouse

calculation the difference (for drag motion)

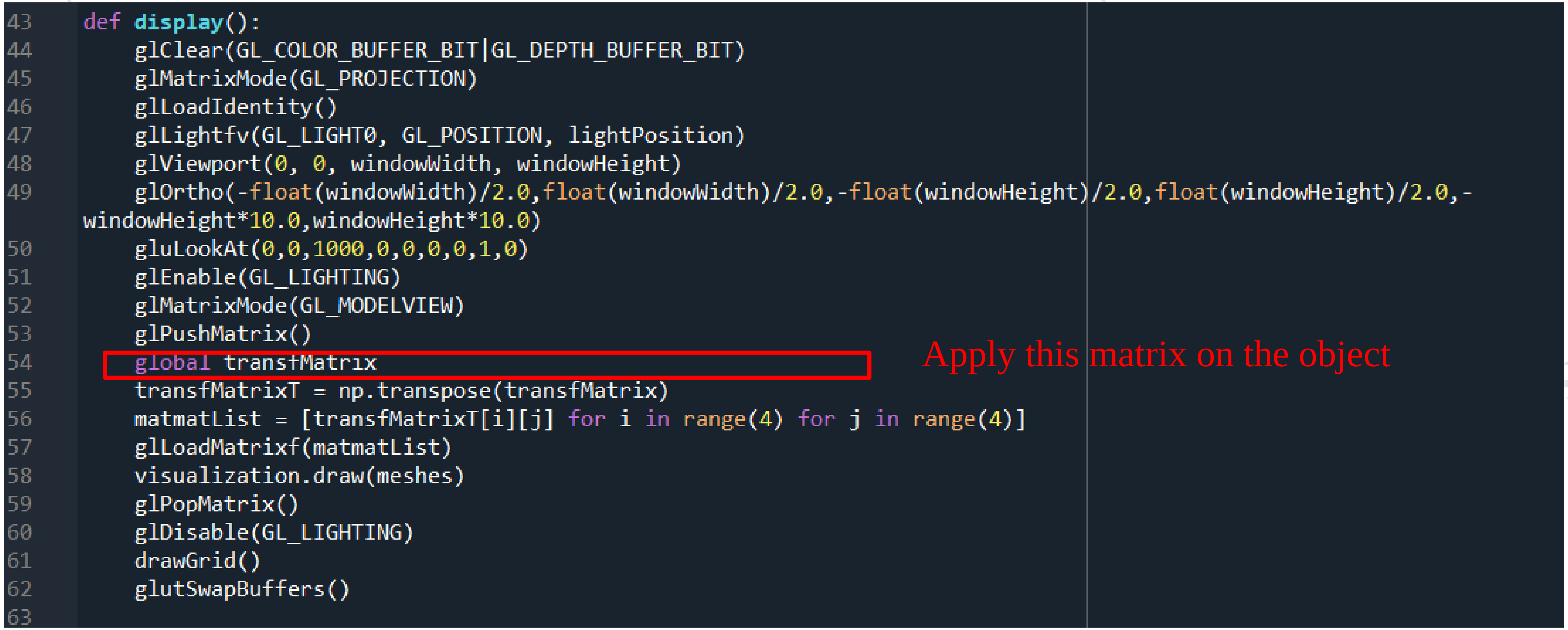
Update mouse position

Mouse control: Right-button for Translation-1



```python
mouseLeftPressed = 0
mouseRightPressed = 0
clickPt = np.array([0,0])
transfMatrix = np.eye(4,dtype=float)
```

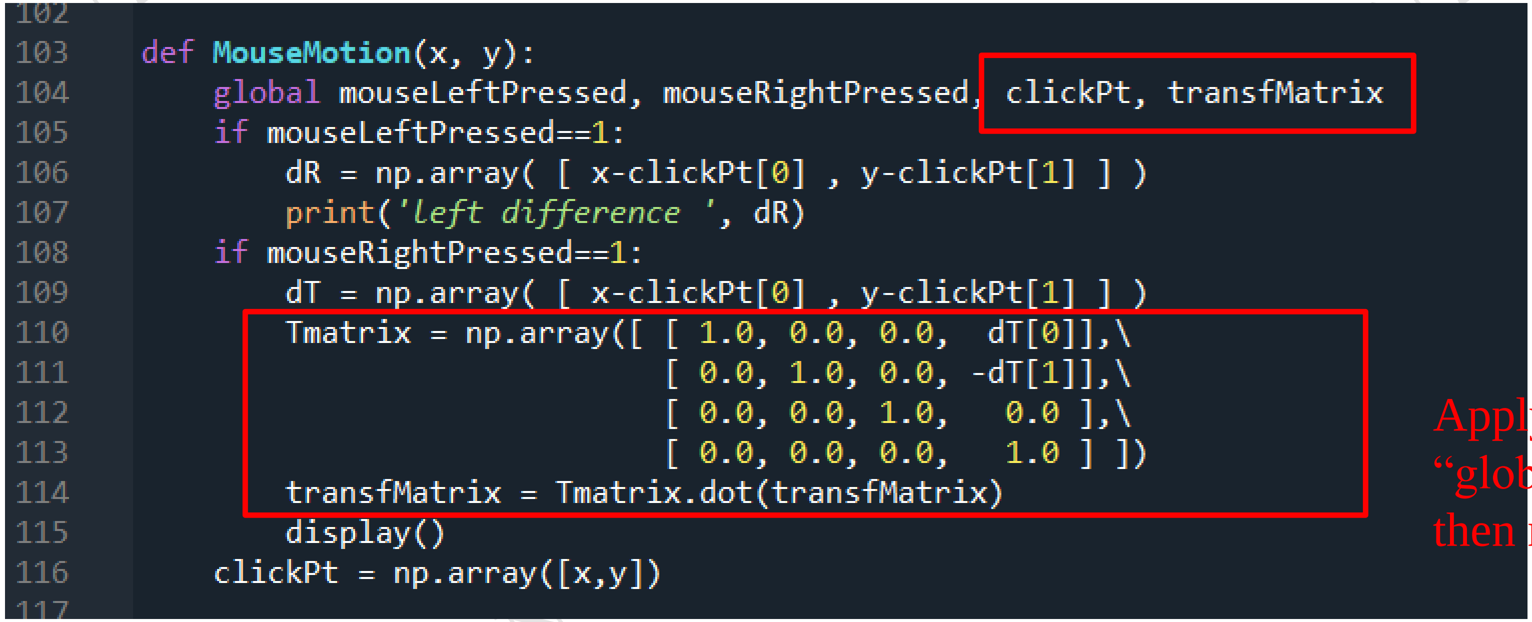
To store “transformation matrix” of the object



```python
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,-float(windowHeight)/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)
    glMatrixMode(GL_MODELVIEW)
    glPushMatrix()
    global transfMatrix
    transfMatrixT = np.transpose(transfMatrix)
    matmatList = [transfMatrixT[i][j] for i in range(4) for j in range(4)]
    glLoadMatrixf(matmatList)
    visualization.draw(meshes)
    glPopMatrix()
    glDisable(GL_LIGHTING)
    drawGrid()
    glutSwapBuffers()
```

Apply this matrix on the object

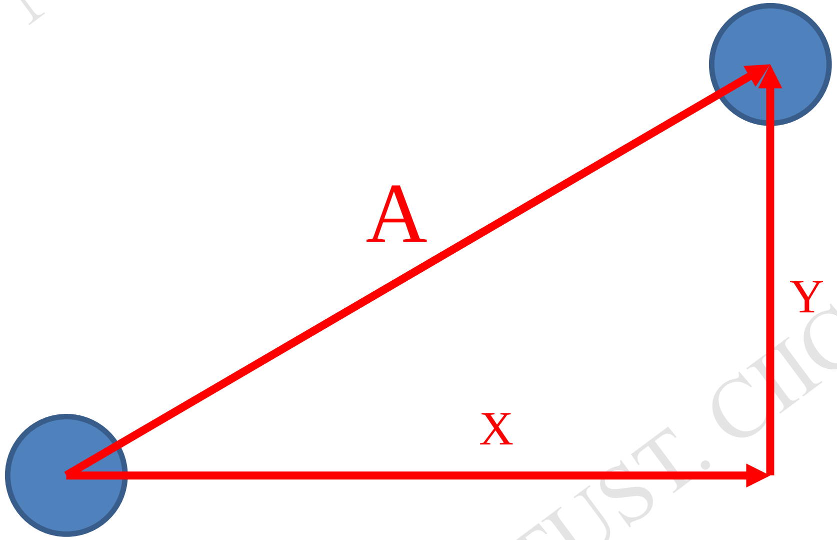
Mouse control: Right-button for Translation-2



```
def MouseMotion(x, y):
    global mouseLeftPressed, mouseRightPressed, clickPt, transfMatrix
    if mouseLeftPressed==1:
        dR = np.array( [ x-clickPt[0] , y-clickPt[1] ] )
        print('left difference ', dR)
    if mouseRightPressed==1:
        dT = np.array( [ x-clickPt[0] , y-clickPt[1] ] )
        Tmatrix = np.array([ [ 1.0, 0.0, 0.0,  dT[0]],\
                             [ 0.0, 1.0, 0.0, -dT[1]],\
                             [ 0.0, 0.0, 1.0,   0.0 ],\
                             [ 0.0, 0.0, 0.0,   1.0 ] ])
        transfMatrix = Tmatrix.dot(transfMatrix)
        display()
    clickPt = np.array([x,y])
```

Apply a translation matrix on “global transformation matrix”, then re-display immediately

Mouse control: Left-button for Rotation-1



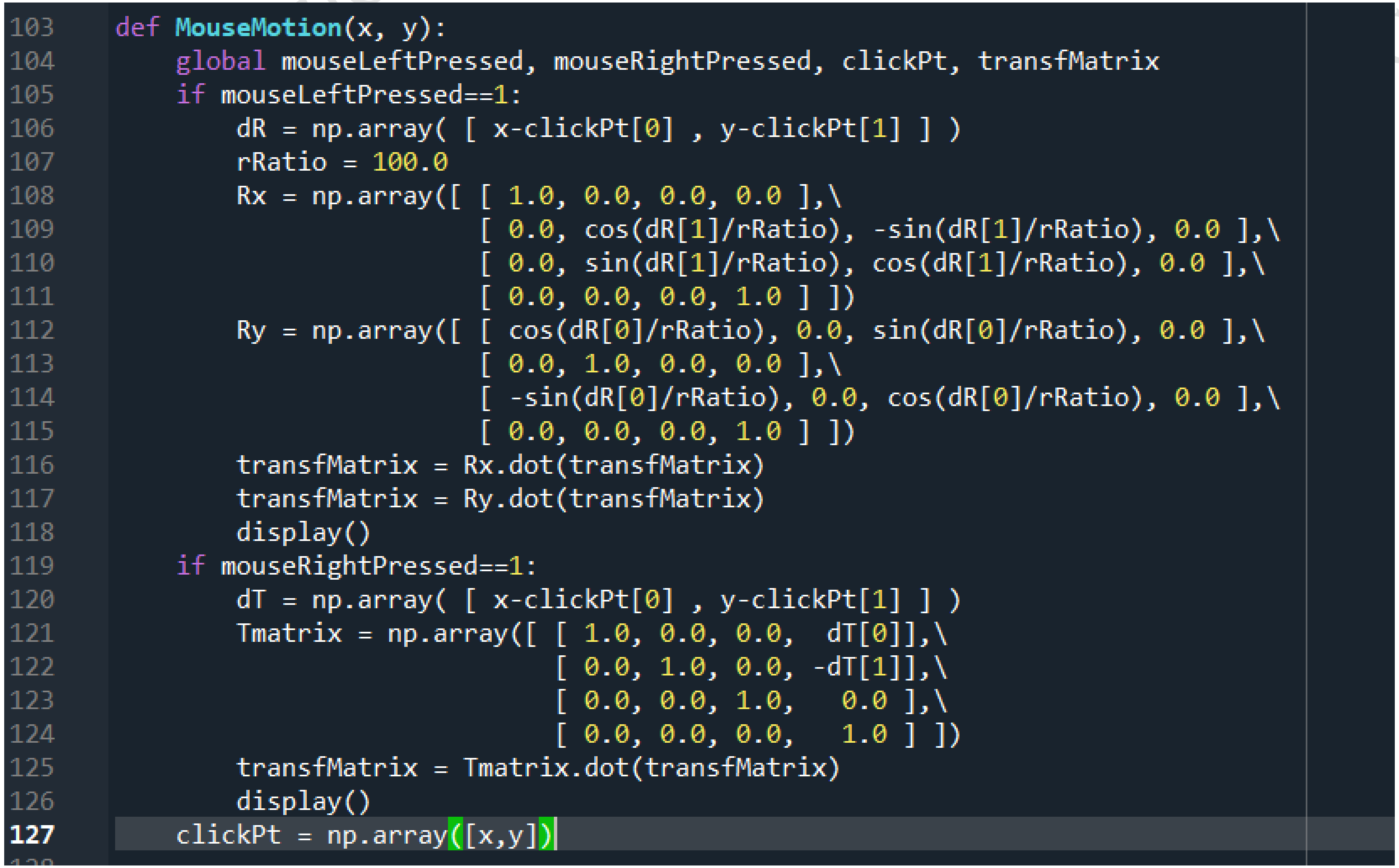
case 1: rotate according to A direction
→ Perfect solution but difficult to carry out

case 2: rotate for X then Y

case 3: rotate for Y then Y

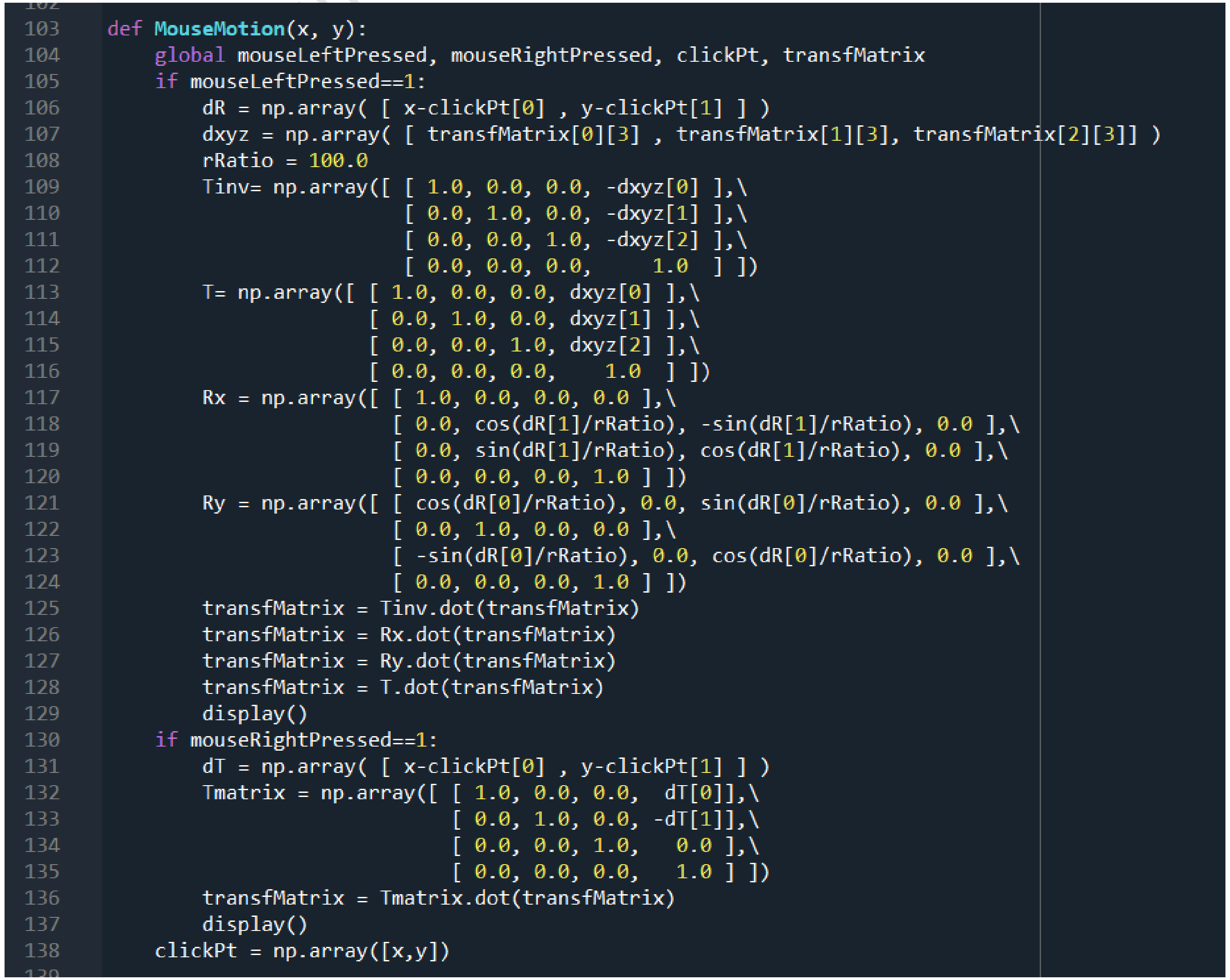
will be a bit different from A

Mouse control: Left-button for Rotation-2



```
def MouseMotion(x, y):
    global mouseLeftPressed, mouseRightPressed, clickPt, transfMatrix
    if mouseLeftPressed==1:
        dR = np.array( [ x-clickPt[0] , y-clickPt[1] ] )
        rRatio = 100.0
        Rx = np.array([ [ 1.0, 0.0, 0.0, 0.0 ],\
                        [ 0.0, cos(dR[1]/rRatio), -sin(dR[1]/rRatio), 0.0 ],\
                        [ 0.0, sin(dR[1]/rRatio), cos(dR[1]/rRatio), 0.0 ],\
                        [ 0.0, 0.0, 0.0, 1.0 ] ])
        Ry = np.array([ [ cos(dR[0]/rRatio), 0.0, sin(dR[0]/rRatio), 0.0 ],\
                        [ 0.0, 1.0, 0.0, 0.0 ],\
                        [ -sin(dR[0]/rRatio), 0.0, cos(dR[0]/rRatio), 0.0 ],\
                        [ 0.0, 0.0, 0.0, 1.0 ] ])
        transfMatrix = Rx.dot(transfMatrix)
        transfMatrix = Ry.dot(transfMatrix)
        display()
    if mouseRightPressed==1:
        dT = np.array( [ x-clickPt[0] , y-clickPt[1] ] )
        Tmatrix = np.array([ [ 1.0, 0.0, 0.0,  dT[0]],\
                             [ 0.0, 1.0, 0.0, -dT[1]],\
                             [ 0.0, 0.0, 1.0,   0.0 ],\
                             [ 0.0, 0.0, 0.0,   1.0 ] ])
        transfMatrix = Tmatrix.dot(transfMatrix)
        display()
    clickPt = np.array([x,y])
```

Mouse control: Rotation and Translation (consider pivot)



```python
def MouseMotion(x, y):
    global mouseLeftPressed, mouseRightPressed, clickPt, transfMatrix
    if mouseLeftPressed==1:
        dR = np.array( [ x-clickPt[0] , y-clickPt[1] ] )
        dxyz = np.array( [ transfMatrix[0][3] , transfMatrix[1][3], transfMatrix[2][3]] )
        rRatio = 100.0
        Tinv= np.array([ [ 1.0, 0.0, 0.0, -dxyz[0] ],\
                         [ 0.0, 1.0, 0.0, -dxyz[1] ],\
                         [ 0.0, 0.0, 1.0, -dxyz[2] ],\
                         [ 0.0, 0.0, 0.0,     1.0  ] ])
        T= np.array([ [ 1.0, 0.0, 0.0, dxyz[0] ],\
                      [ 0.0, 1.0, 0.0, dxyz[1] ],\
                      [ 0.0, 0.0, 1.0, dxyz[2] ],\
                      [ 0.0, 0.0, 0.0,    1.0  ] ])
        Rx = np.array([ [ 1.0, 0.0, 0.0, 0.0 ],\
                        [ 0.0, cos(dR[1]/rRatio), -sin(dR[1]/rRatio), 0.0 ],\
                        [ 0.0, sin(dR[1]/rRatio), cos(dR[1]/rRatio), 0.0 ],\
                        [ 0.0, 0.0, 0.0, 1.0 ] ])
        Ry = np.array([ [ cos(dR[0]/rRatio), 0.0, sin(dR[0]/rRatio), 0.0 ],\
                        [ 0.0, 1.0, 0.0, 0.0 ],\
                        [ -sin(dR[0]/rRatio), 0.0, cos(dR[0]/rRatio), 0.0 ],\
                        [ 0.0, 0.0, 0.0, 1.0 ] ])
        transfMatrix = Tinv.dot(transfMatrix)
        transfMatrix = Rx.dot(transfMatrix)
        transfMatrix = Ry.dot(transfMatrix)
        transfMatrix = T.dot(transfMatrix)
        display()
    if mouseRightPressed==1:
        dT = np.array( [ x-clickPt[0] , y-clickPt[1] ] )
        Tmatrix = np.array([ [ 1.0, 0.0, 0.0,  dT[0]],\
                             [ 0.0, 1.0, 0.0, -dT[1]],\
                             [ 0.0, 0.0, 1.0,   0.0 ],\
                             [ 0.0, 0.0, 0.0,   1.0 ] ])
        transfMatrix = Tmatrix.dot(transfMatrix)
        display()
    clickPt = np.array([x,y])
```

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

Find us on: facebook®

This photo is licensed under CC BY-ND