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)

```
86
87  def MouseFunc(button, state, x, y):
88      print(button, state)
89
90  def MouseMotion(x, y):
91      print(x, y)
92
```

```
 99  glutReshapeFunc(reshape)
100  glutDisplayFunc(display)
101  glutKeyboardFunc(keyboard)
102  glutSpecialFunc(keyboardSpecial)
103  glutMouseFunc(MouseFunc)
104  glutMotionFunc(MouseMotion)
105  glEnable(GL_DEPTH_TEST)
```

Mouse control (know the definition)

```
 99  glutReshapeFunc(reshape)
100  glutDisplayFunc(display)
101  glutKeyboardFunc(keyboard)
102  glutSpecialFunc(keyboardSpecial)
103  glutMouseFunc(MouseFunc)
104  glutMotionFunc(MouseMotion)
105  glEnable(GL_DEPTH_TEST)
```

```
86
87  def MouseFunc(button, state, x, y):
88      print(button, state)
89
90  def MouseMotion(x, y):
91      print(x, y)
92
```

Mouse control (for translation and rotation)

```
87  def MouseFunc(button, state, x, y):
88      global mouseLeftPressed
89      global mouseRightPressed
90      if state == 1:
91          if button == 0:
92              mouseLeftPressed = 0
93          if button == 2:
94              mouseRightPressed = 0
95      else:
96          if button == 0:
97              mouseLeftPressed = 1
98          if button == 2:
99              mouseRightPressed = 1
100
101 def MouseMotion(x, y):
102     global mouseLeftPressed
103     global mouseRightPressed
104     if mouseLeftPressed==1:
105         print('Left ',x, y)
106     if mouseRightPressed==1:
107         print('Right ',x, y)
108
```

Use global variables to store the status of “Press”

Mouse control (for translation and rotation)
Store the difference vector after clicking and dragging

```
89  def MouseFunc(button, state, x, y):
90      global mouseLeftPressed, mouseRightPressed, clickPt
91      if state == 1:
92          if button == 0:
93              mouseLeftPressed = 0
94          if button == 2:
95              mouseRightPressed = 0
96      else:
97          if button == 0:
98              mouseLeftPressed = 1
99          if button == 2:
100             mouseRightPressed = 1
101         clickPt = np.array([x,y])
102
103
104 def MouseMotion(x, y):
105     global mouseLeftPressed, mouseRightPressed, clickPt
106     if mouseLeftPressed==1:
107       dR = np.array( [ x-clickPt[0] , y-clickPt[1] ] )
108       print('left difference ', dR)
109     if mouseRightPressed==1:
110       dT = np.array( [ x-clickPt[0] , y-clickPt[1] ] )
111       print('right difference', dT)
112     clickPt = np.array([x,y])
113
```

global variables

```
14  mouseLeftPressed = 0
15  mouseRightPressed = 0
16  clickPt = np.array([0,0])
17
```

Once you click mouse

calculation the difference (for drag motion)

Update mouse position

Mouse control: Right-button for Translation-1

```
14    mouseLeftPressed = 0
15    mouseRightPressed = 0
16    clickPt = np.array([0,0])
17    transfMatrix = np.eye(4,dtype=float)
```

To store “transformation matrix” of the object

```
43  def display():
44      glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
45      glMatrixMode(GL_PROJECTION)
46      glLoadIdentity()
47      glLightfv(GL_LIGHT0, GL_POSITION, lightPosition)
48      glViewport(0, 0, windowWidth, windowHeight)
49      glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.0,-float(windowHeight)/2.0,float(windowHeight)/2.0,-
    windowHeight*10.0,windowHeight*10.0)
50      gluLookAt(0,0,1000,0,0,0,0,1,0)
51      glEnable(GL_LIGHTING)
52      glMatrixMode(GL_MODELVIEW)
53      glPushMatrix()
54      global transfMatrix
55      transfMatrixT = np.transpose(transfMatrix)
56      matmatList = [transfMatrixT[i][j] for i in range(4) for j in range(4)]
57      glLoadMatrixf(matmatList)
58      visualization.draw(meshes)
59      glPopMatrix()
60      glDisable(GL_LIGHTING)
61      drawGrid()
62      glutSwapBuffers()
63
```

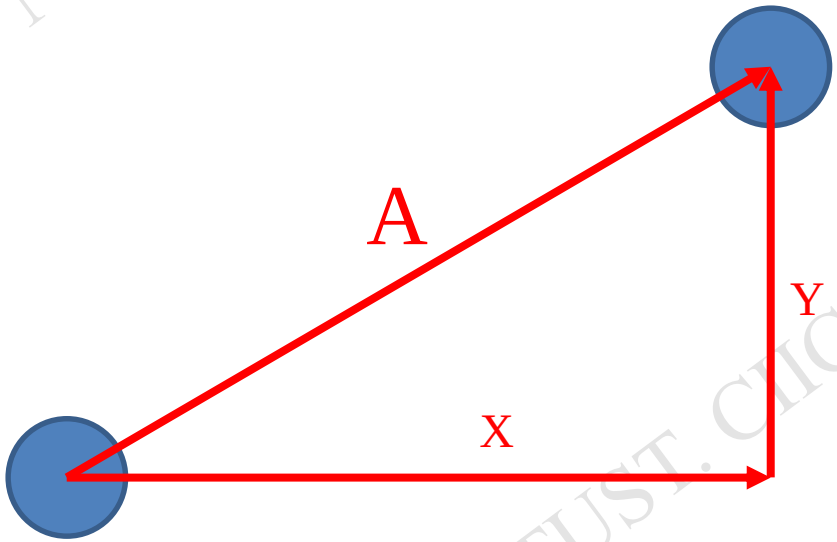
Apply this matrix on the object

Mouse control: Right-button for Translation-2

```
102
103 def MouseMotion(x, y):
104     global mouseLeftPressed, mouseRightPressed, clickPt, transfMatrix
105     if mouseLeftPressed==1:
106         dR = np.array( [ x-clickPt[0] , y-clickPt[1] ] )
107         print('left difference ', dR)
108     if mouseRightPressed==1:
109         dT = np.array( [ x-clickPt[0] , y-clickPt[1] ] )
110         Tmatrix = np.array([ [ 1.0, 0.0, 0.0,  dT[0]],\
111                              [ 0.0, 1.0, 0.0, -dT[1]],\
112                              [ 0.0, 0.0, 1.0,   0.0 ],\
113                              [ 0.0, 0.0, 0.0,   1.0 ] ])
114         transfMatrix = Tmatrix.dot(transfMatrix)
115         display()
116     clickPt = np.array([x,y])
117
```

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

```
103  def MouseMotion(x, y):
104      global mouseLeftPressed, mouseRightPressed, clickPt, transfMatrix
105      if mouseLeftPressed==1:
106          dR = np.array( [ x-clickPt[0] , y-clickPt[1] ] )
107          rRatio = 100.0
108          Rx = np.array([ [ 1.0, 0.0, 0.0, 0.0 ],\
109                          [ 0.0, cos(dR[1]/rRatio), -sin(dR[1]/rRatio), 0.0 ],\
110                          [ 0.0, sin(dR[1]/rRatio), cos(dR[1]/rRatio), 0.0 ],\
111                          [ 0.0, 0.0, 0.0, 1.0 ] ])
112          Ry = np.array([ [ cos(dR[0]/rRatio), 0.0, sin(dR[0]/rRatio), 0.0 ],\
113                          [ 0.0, 1.0, 0.0, 0.0 ],\
114                          [ -sin(dR[0]/rRatio), 0.0, cos(dR[0]/rRatio), 0.0 ],\
115                          [ 0.0, 0.0, 0.0, 1.0 ] ])
116          transfMatrix = Rx.dot(transfMatrix)
117          transfMatrix = Ry.dot(transfMatrix)
118          display()
119      if mouseRightPressed==1:
120          dT = np.array( [ x-clickPt[0] , y-clickPt[1] ] )
121          Tmatrix = np.array([ [ 1.0, 0.0, 0.0,  dT[0]],\
122                               [ 0.0, 1.0, 0.0, -dT[1]],\
123                               [ 0.0, 0.0, 1.0,   0.0 ],\
124                               [ 0.0, 0.0, 0.0,   1.0 ] ])
125          transfMatrix = Tmatrix.dot(transfMatrix)
126          display()
127      clickPt = np.array([x,y])
```

Mouse control: Rotation and Translation (consider pivot)

```
103  def MouseMotion(x, y):
104      global mouseLeftPressed, mouseRightPressed, clickPt, transfMatrix
105      if mouseLeftPressed==1:
106          dR = np.array( [ x-clickPt[0] , y-clickPt[1] ] )
107          dxyz = np.array( [ transfMatrix[0][3] , transfMatrix[1][3], transfMatrix[2][3]] )
108          rRatio = 100.0
109          Tinv= np.array([ [ 1.0, 0.0, 0.0, -dxyz[0] ],\
110                           [ 0.0, 1.0, 0.0, -dxyz[1] ],\
111                           [ 0.0, 0.0, 1.0, -dxyz[2] ],\
112                           [ 0.0, 0.0, 0.0,     1.0  ] ])
113          T= np.array([ [ 1.0, 0.0, 0.0, dxyz[0] ],\
114                        [ 0.0, 1.0, 0.0, dxyz[1] ],\
115                        [ 0.0, 0.0, 1.0, dxyz[2] ],\
116                        [ 0.0, 0.0, 0.0,    1.0  ] ])
117          Rx = np.array([ [ 1.0, 0.0, 0.0, 0.0 ],\
118                          [ 0.0, cos(dR[1]/rRatio), -sin(dR[1]/rRatio), 0.0 ],\
119                          [ 0.0, sin(dR[1]/rRatio), cos(dR[1]/rRatio), 0.0 ],\
120                          [ 0.0, 0.0, 0.0, 1.0 ] ])
121          Ry = np.array([ [ cos(dR[0]/rRatio), 0.0, sin(dR[0]/rRatio), 0.0 ],\
122                          [ 0.0, 1.0, 0.0, 0.0 ],\
123                          [ -sin(dR[0]/rRatio), 0.0, cos(dR[0]/rRatio), 0.0 ],\
124                          [ 0.0, 0.0, 0.0, 1.0 ] ])
125          transfMatrix = Tinv.dot(transfMatrix)
126          transfMatrix = Rx.dot(transfMatrix)
127          transfMatrix = Ry.dot(transfMatrix)
128          transfMatrix = T.dot(transfMatrix)
129          display()
130      if mouseRightPressed==1:
131          dT = np.array( [ x-clickPt[0] , y-clickPt[1] ] )
132          Tmatrix = np.array([ [ 1.0, 0.0, 0.0,  dT[0]],\
133                               [ 0.0, 1.0, 0.0, -dT[1]],\
134                               [ 0.0, 0.0, 1.0,   0.0 ],\
135                               [ 0.0, 0.0, 0.0,   1.0 ] ])
136          transfMatrix = Tmatrix.dot(transfMatrix)
137          display()
138      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