Advanced Computer Graphics

Lecture-08 Introduction to OpenGL-6

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

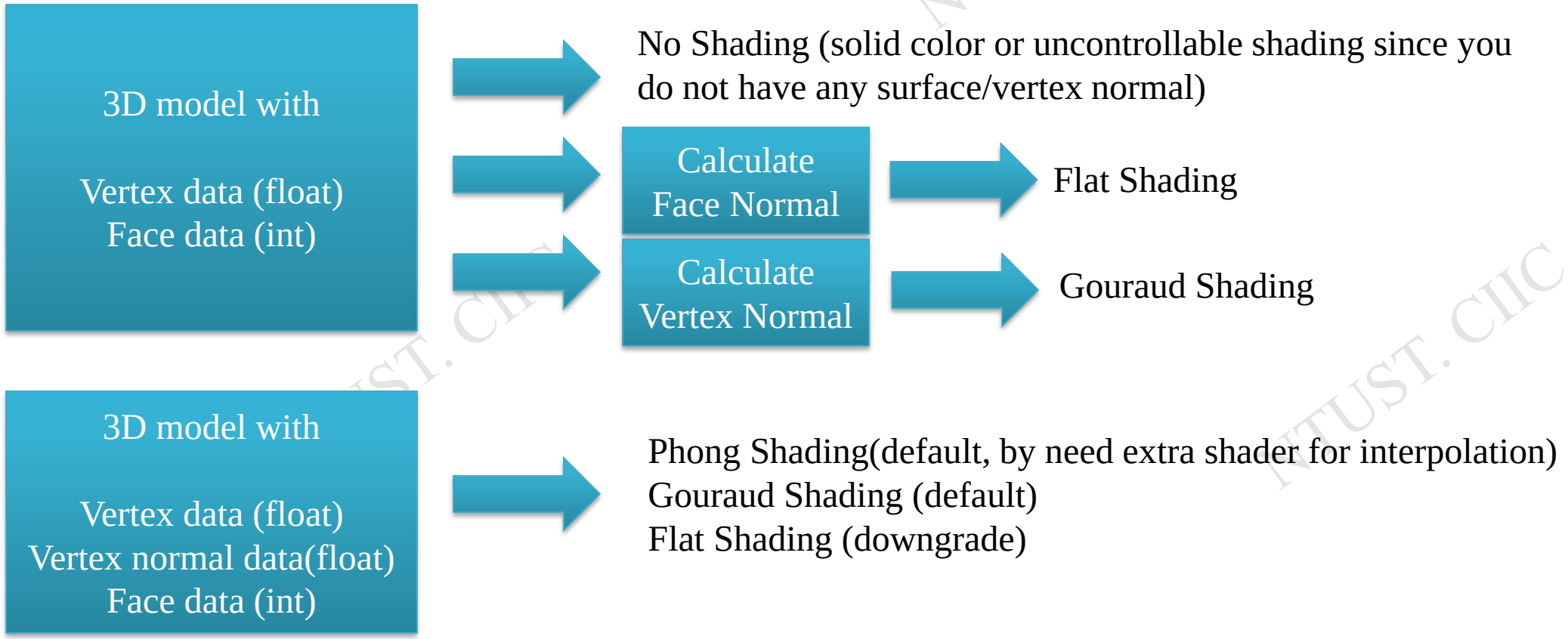
Outline

- Gouraud Shading / Flat Shading
- Backface Cull
- Depth Test
- Camera Control

- The necessary data: vertex's normal

Gouraud Shading vs Flat Shading

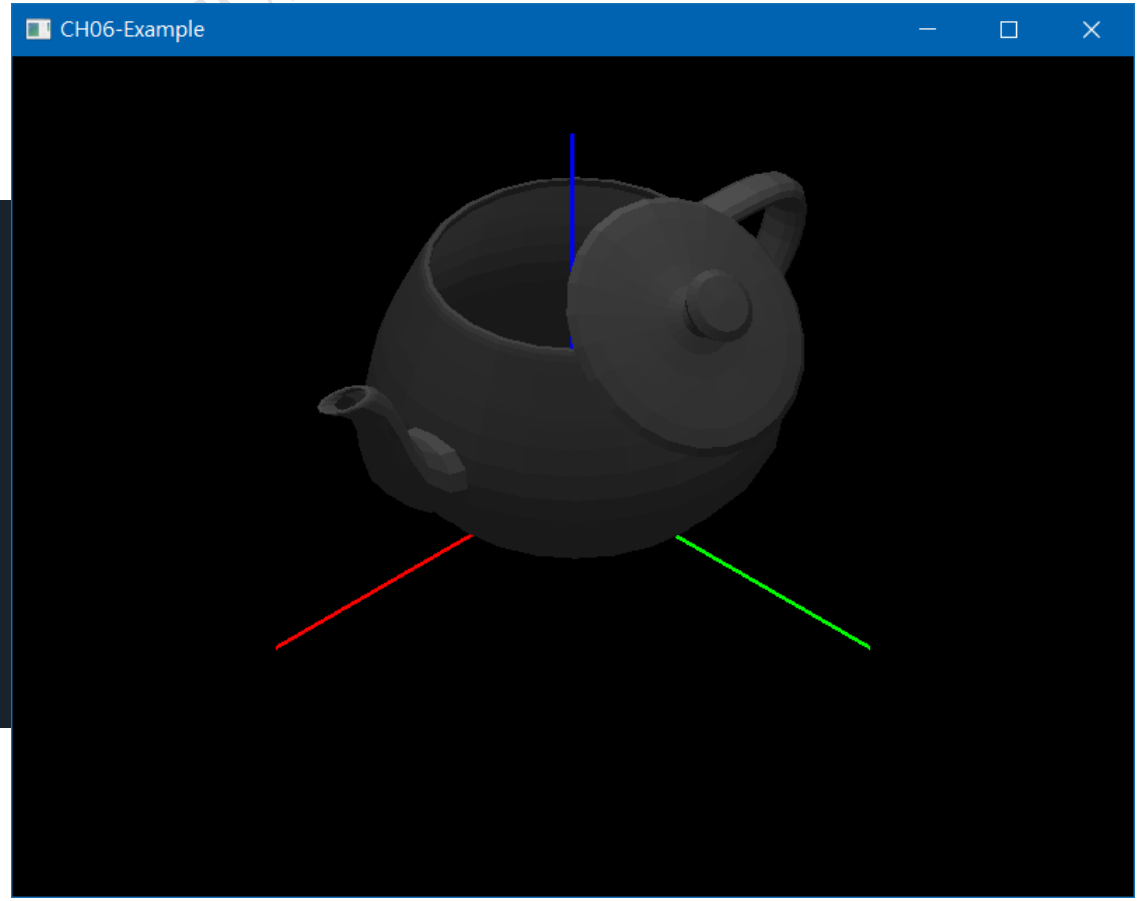
- For better visualization, you may need more data such vertex-normal.



Gouraud Shading vs Flat Shading

- Calculate Face Normal and assign it to each vertexes on a tringle (Flat shading)

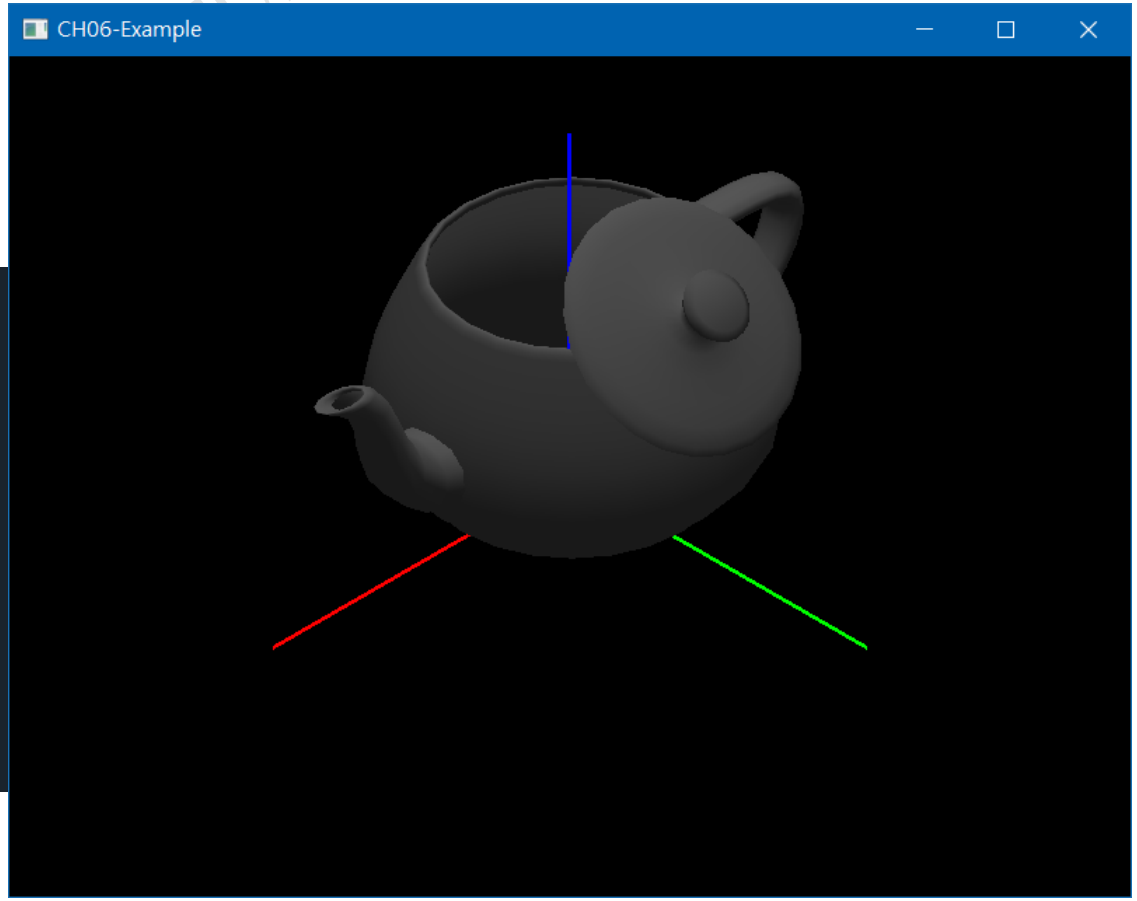
```
4633    def drawTeapot():
4634        glBegin(GL_TRIANGLES)
4635        for fID in teapotFace:
4636            v1 = np.subtract(teapotVertex[fID[1]] ,teapotVertex[fID[0]])
4637            v2 = np.subtract(teapotVertex[fID[2]] ,teapotVertex[fID[0]])
4638            nv = np.cross(v1,v2)
4639            nlen = np.linalg.norm(nv, ord=1)
4640            nv = nv / nlen
4641            glNormal3f(nv[0],nv[1],nv[2])
4642            glVertex3fv(teapotVertex[fID[0]])
4643            glVertex3fv(teapotVertex[fID[1]])
4644            glVertex3fv(teapotVertex[fID[2]])
4645        glEnd()
```



Gouraud Shading vs Flat Shading

- Assign a normal vector to each vertex

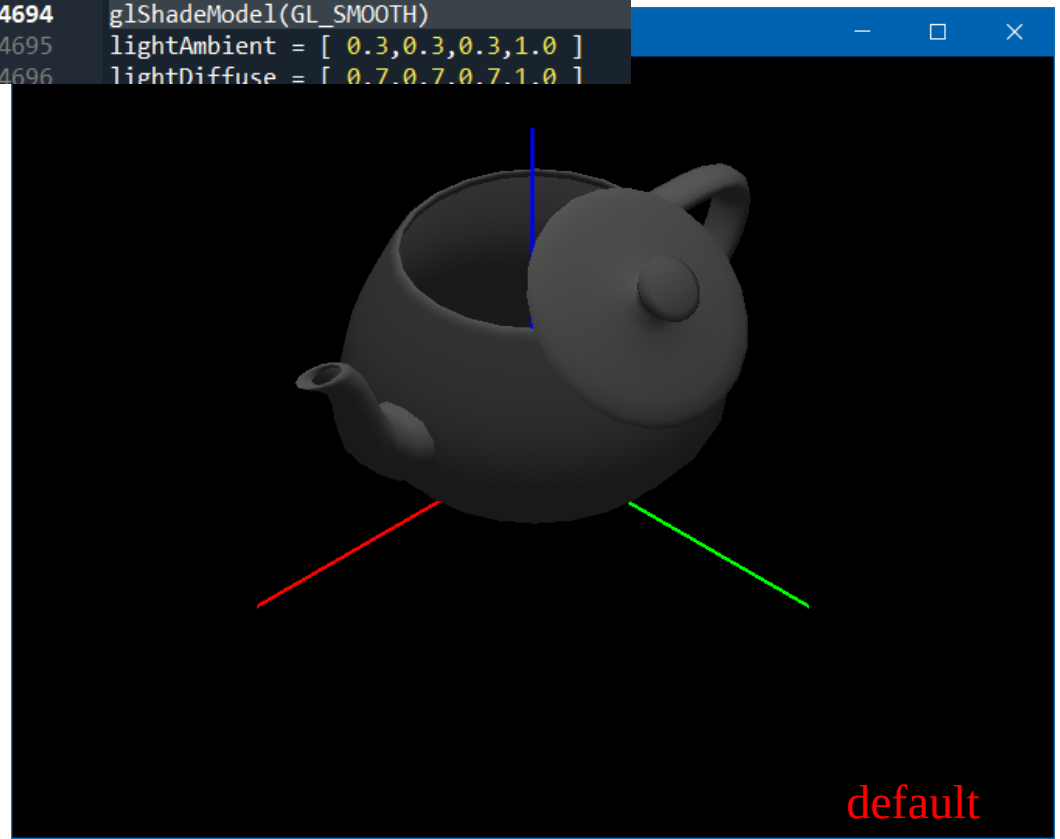
```
4633    def drawTeapot():
4634        glBegin(GL_TRIANGLES)
4635        for fID in teapotFace:
4636            glNormal3fv(teapotVertexNormal[fID[0]])
4637            glVertex3fv(teapotVertex[fID[0]])
4638            glNormal3fv(teapotVertexNormal[fID[1]])
4639            glVertex3fv(teapotVertex[fID[1]])
4640            glNormal3fv(teapotVertexNormal[fID[2]])
4641            glVertex3fv(teapotVertex[fID[2]])
4642        glEnd()
4643
```



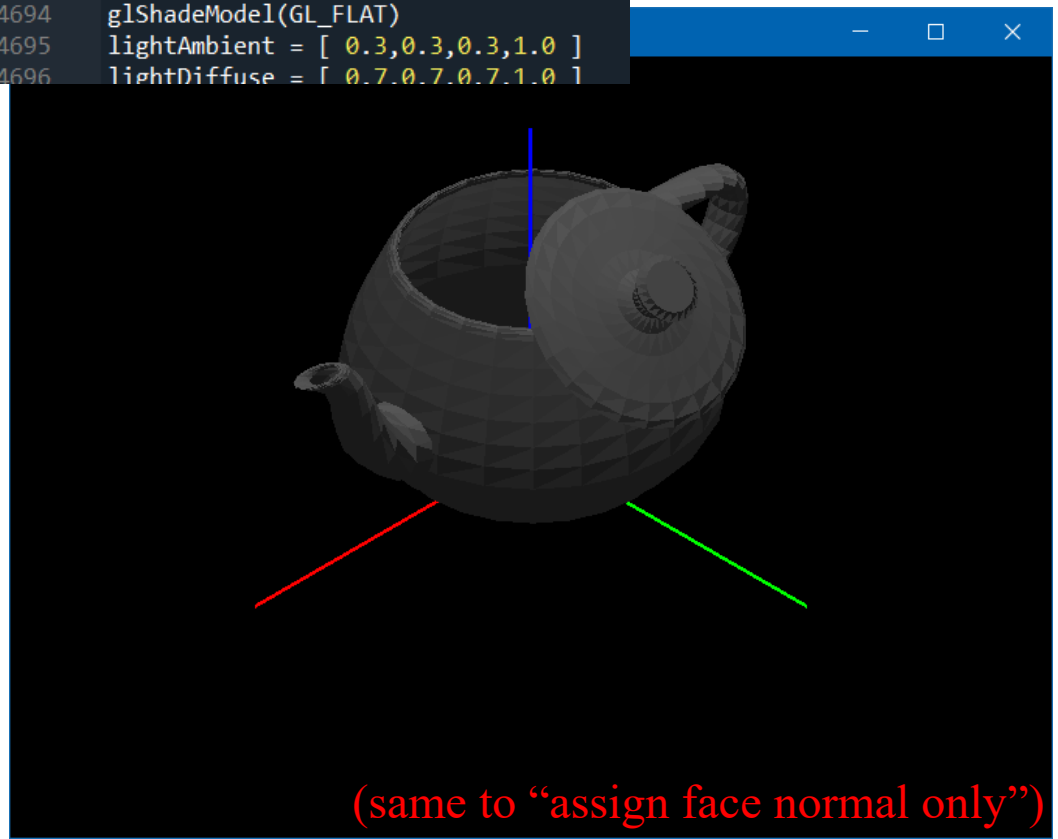
Gouraud Shading vs Flat Shading

```
4633  def drawTeapot():(assign normal vectors to vertexes)
4634      glBegin(GL_TRIANGLES)
4635      for fID in teapotFace:
4636          glNormal3fv(teapotVertexNormal[fID[0]])
4637          glVertex3fv(teapotVertex[fID[0]])
4638          glNormal3fv(teapotVertexNormal[fID[1]])
4639          glVertex3fv(teapotVertex[fID[1]])
4640          glNormal3fv(teapotVertexNormal[fID[2]])
4641          glVertex3fv(teapotVertex[fID[2]])
4642      glEnd()
```

```
4693  glEnable(GL_LIGHT0)
4694  glShadeModel(GL_SMOOTH)
4695  lightAmbient = [ 0.3,0.3,0.3,1.0 ]
4696  lightDiffuse = [ 0.7,0.7,0.7,1.0 ]
```

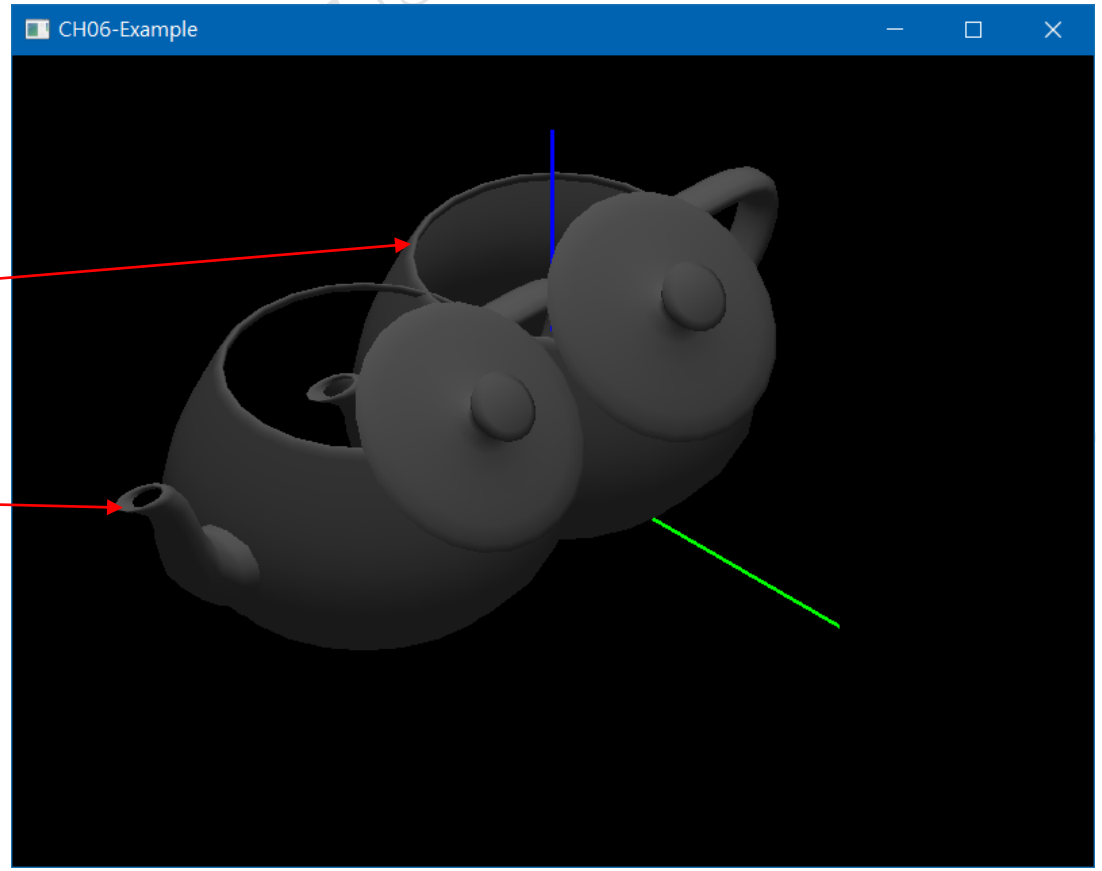



```
4693  glEnable(GL_LIGHT0)
4694  glShadeModel(GL_FLAT)
4695  lightAmbient = [ 0.3,0.3,0.3,1.0 ]
4696  lightDiffuse = [ 0.7,0.7,0.7,1.0 ]
```



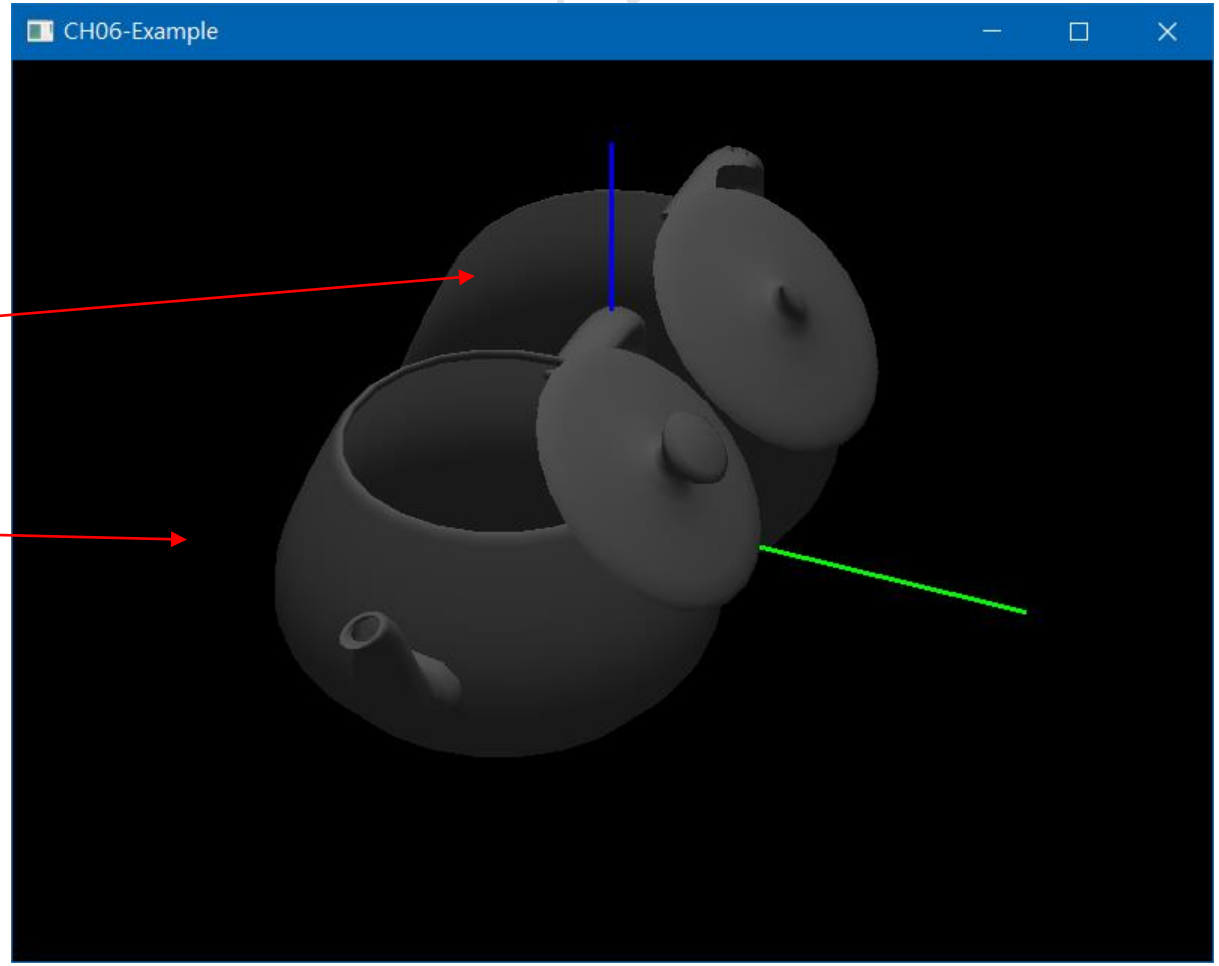
Enable / Disable Cull Face

```
4667        glEnable(GL_LIGHTING)
4668        glPushMatrix()
4669        glDisable(GL_CULL_FACE)
4670        drawTeapot()
4671        glPopMatrix()
4672        glPushMatrix()
4673        glEnable(GL_CULL_FACE)
4674        glTranslatef(200,0,0)
4675        drawTeapot()
4676        glPopMatrix()
4677        glDisable(GL_LIGHTING)
4678        drawCoordinate()
4679        glutSwapBuffers()
```



Enable/Disable Depth_Test

```
4669    angle = angle+1
4670    glRotatef(angle,0,0,1)
4671    glPushMatrix()
4672    glDisable(GL_DEPTH_TEST)
4673    drawTeapot()
4674    glPopMatrix()
4675    glPushMatrix()
4676    glEnable(GL_DEPTH_TEST)
4677    glTranslatef(200,0,0)
4678    drawTeapot()
4679    glPopMatrix()
4680    glDisable(GL_LIGHTING)
4681    drawCoordinate()
4682    glutSwapBuffers()
```

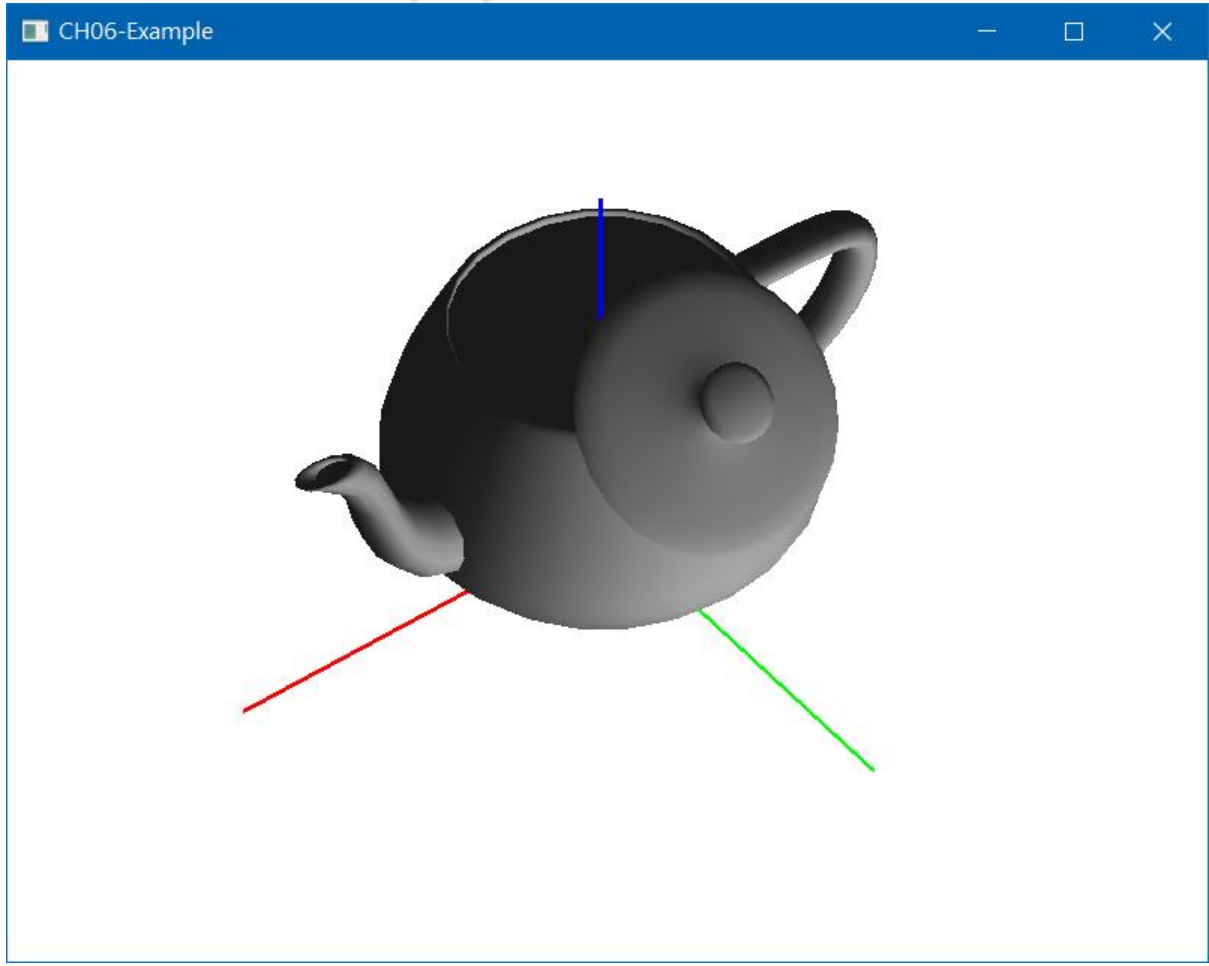


gluLookAt vs Coordinate Transformation

- gluLookAt

  Look At
  (300,400,500
  10,20,30
  0,0,1)

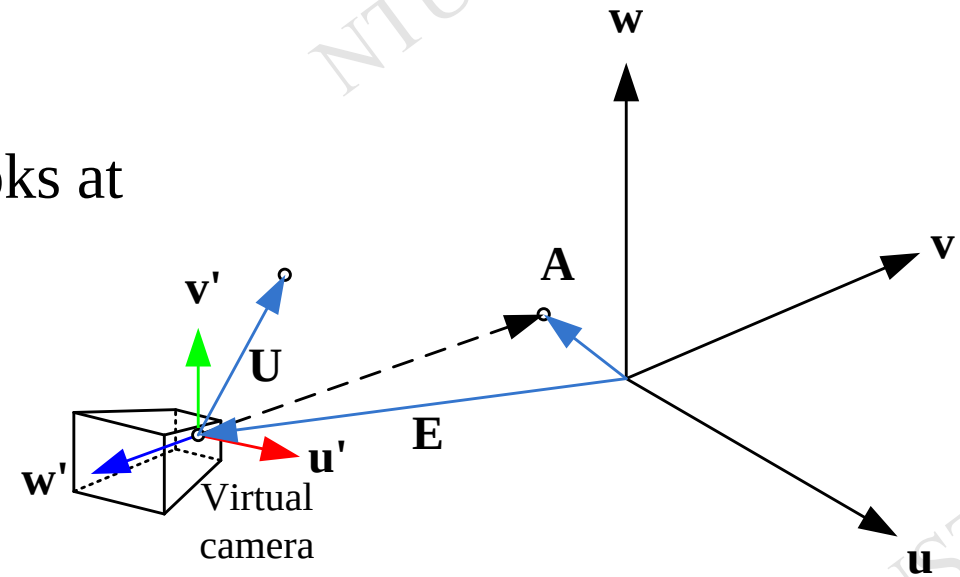
```
4660  def display():
4661      glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
4662      glMatrixMode(GL_PROJECTION)
4663      glLoadIdentity()
4664      glViewport(0, 0, windowWidth, windowHeight)
4665      glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.
      2.0,float(windowHeight)/2.0,-windowHeight*10.0,windowHeig
4666      gluLookAt(300,400,500,10,20,30,0,0,1)
4667      glEnable(GL_LIGHTING)
4668      glPushMatrix()
4669      drawTeapot()
4670      glPopMatrix()
4671      glDisable(GL_LIGHTING)
4672      drawCoordinate()
4673      glutSwapBuffers()
4674
```



gluLookAt vs Coordinate Transformation

- Coordinate Transformation

  Here,
  - E: eye (or camera) position
  - A: reference point, where eye looks at
  - U: direction of up vector



gluLookAt

The **gluLookAt** function defines a viewing transformation.

```
void gluLookAt(
  GLdouble eyex,
  GLdouble eyey,
  GLdouble eyez,
  GLdouble centerx,
  GLdouble centery,
  GLdouble centerz,
  GLdouble upx,
  GLdouble upy,
  GLdouble upz
);
```

Parameters

eyex, eyey, eyez
The position of the eye point.

centerx, centery, centerz
The position of the reference point.

upx, upy, upz
The direction of the up vector.

gluLookAt vs Coordinate Transformation

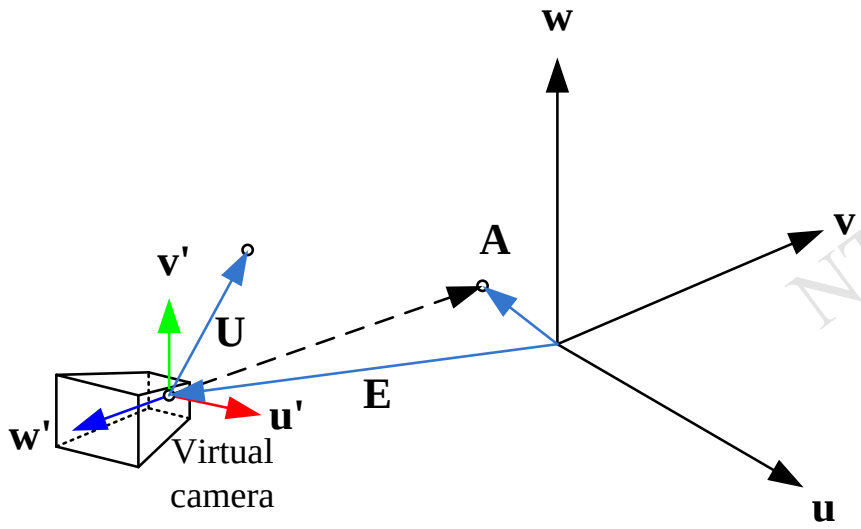
- To determine the camera transformation matrix
- Step-1: find **w'** from (**E-A**).
- Step-2: **u'** from cross product of **U** and **w'** (and convert into unit vector)
- Step-3: **v'** from cross product of **w'** and **u'**.

$$\begin{bmatrix} u_x' & v_x' & w_x' & E_x \\ u_y' & v_y' & w_y' & E_y \\ u_z' & v_z' & w_z' & E_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{w}' = Normalize(\mathbf{E} - \mathbf{A})$$

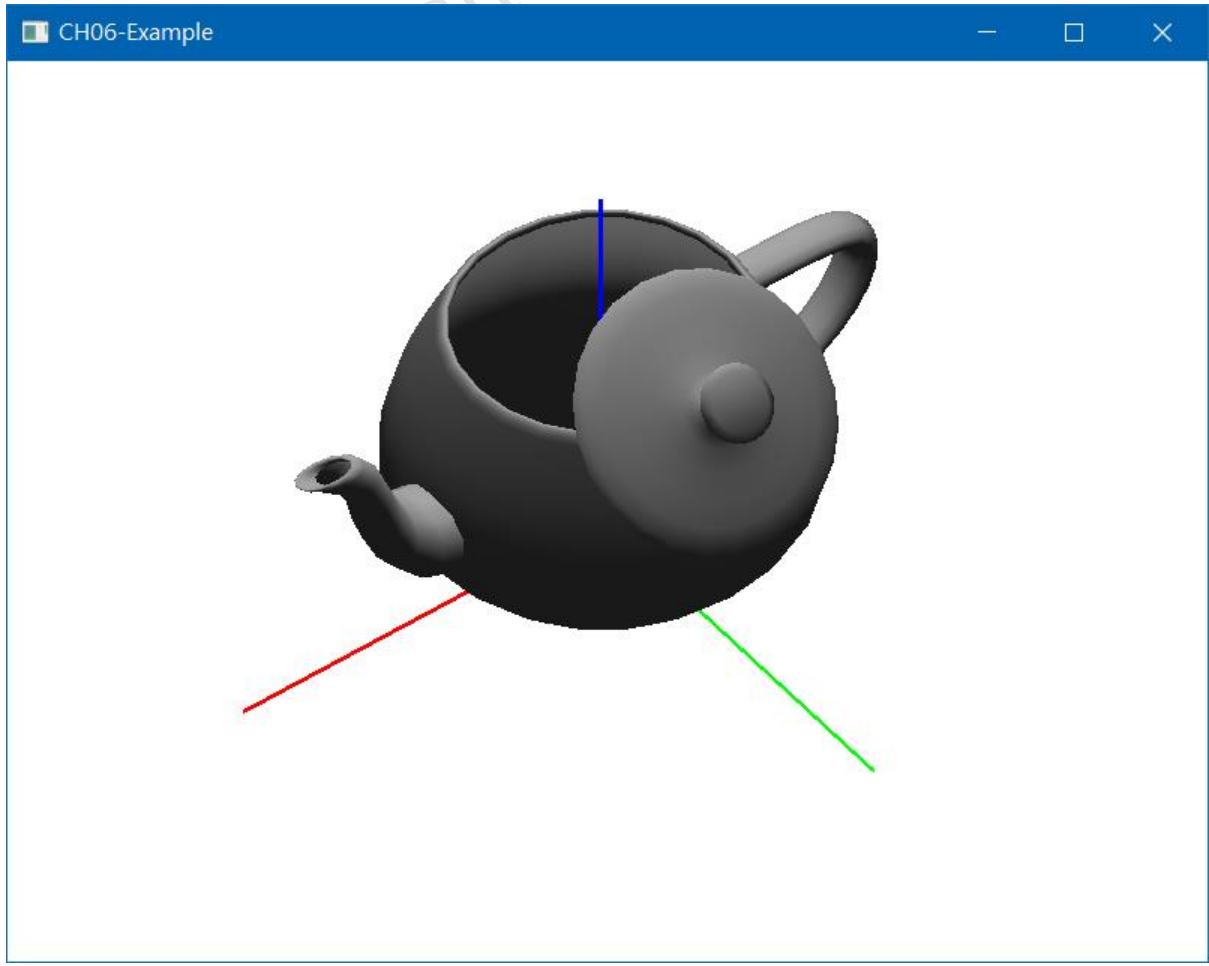
$$\mathbf{u}' = Normalize(\mathbf{U} \times \mathbf{w}')$$

$$\mathbf{v}' = \mathbf{w}' \times \mathbf{u}'$$



gluLookAt vs Coordinate Transformation

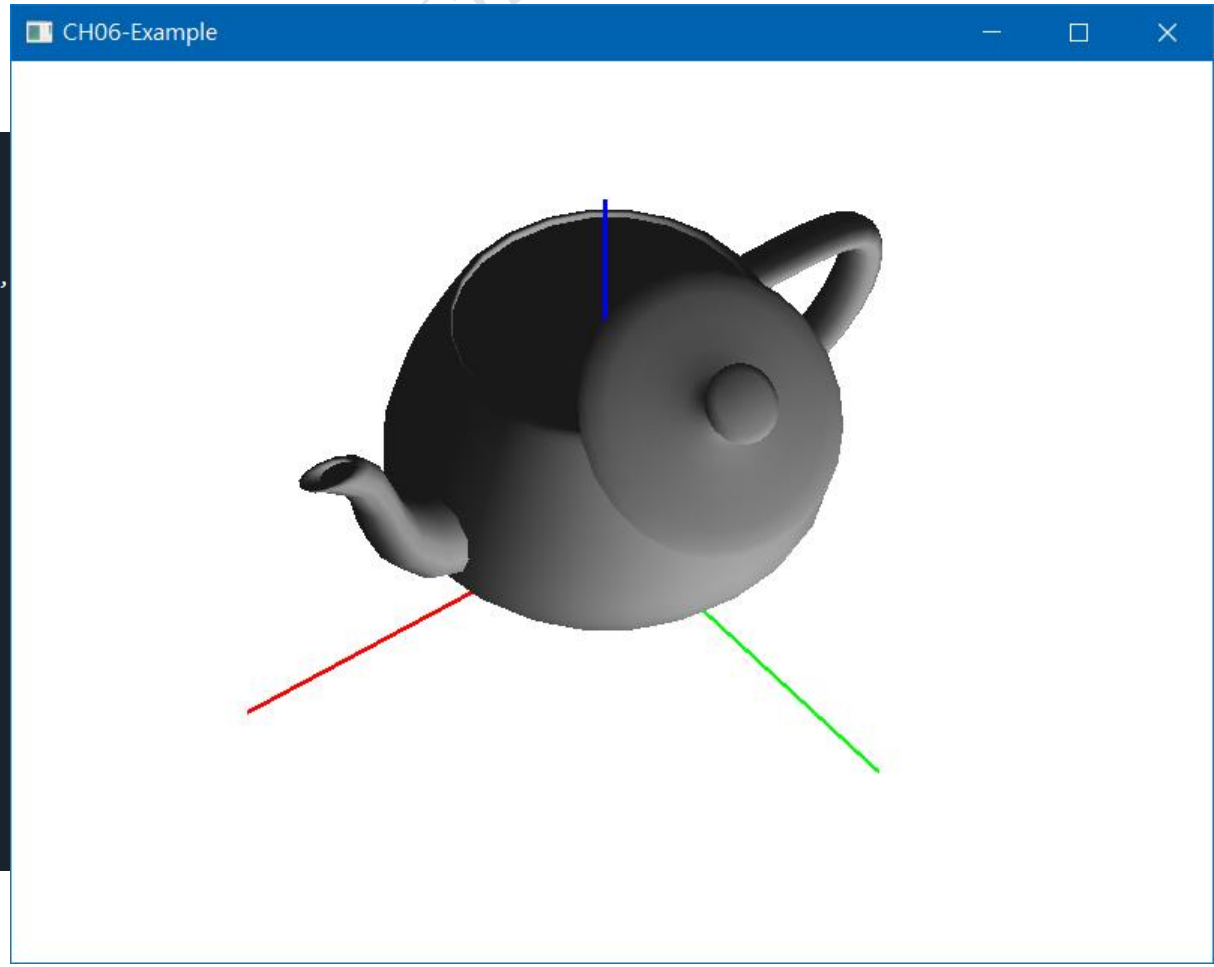
```
4665    glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.0,-float(windowHeight)/2.0,float(windowHeight)/
    windowHeight*10.0,windowHeight*10.0)
4666    glMatrixMode(GL_MODELVIEW)
4667    w = np.array([300,400,500])-np.array([10,20,30])
4668    w = w / np.linalg.norm(w)
4669    U = np.array([0,0,1])
4670    u = np.cross(U,w)
4671    u = u / np.linalg.norm(u)
4672    v = np.cross(w,u)
4673    M = [ [u[0], v[0], w[0], 300],[ u[1], v[1], w[1], 400], [u[2], v[2], w[2], 500], [0., 0., 0., 1.] ]
4674    Minv = np.linalg.inv(M)
4675    MinvT = np.transpose(Minv)
4676    matmatList = [MinvT[i][j] for i in range(4) for j in range(4)]
4677    glLoadMatrixf(matmatList)
4678    glEnable(GL_LIGHTING)
4679    glPushMatrix()
4680    drawTeapot()
4681    glPopMatrix()
4682    glDisable(GL_LIGHTING)
4683    drawCoordinate()
4684    glutSwapBuffers()
```



Note: light position is different

gluLookAt vs Coordinate Transformation

```
4658    def display():
4659        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
4660        glMatrixMode(GL_PROJECTION)
4661        glLoadIdentity()
4662        glViewport(0, 0, windowWidth, windowHeight)
4663        glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.0,-float(windowHeight)/2.0,float(windowHeight)/2.0,
    windowHeight*10.0,windowHeight*10.0)
4664        glMatrixMode(GL_MODELVIEW)
4665        w = np.array([300,400,500])-np.array([10,20,30])
4666        w = w / np.linalg.norm(w)
4667        U = np.array([0,0,1])
4668        u = np.cross(U,w)
4669        u = u / np.linalg.norm(u)
4670        v = np.cross(w,u)
4671        M = [ [u[0], v[0], w[0], 300],[ u[1], v[1], w[1], 400], [u[2], v[2], w[2], 500], [0., 0., 0., 1.] ]
4672        Minv = np.linalg.inv(M)
4673        MinvT = np.transpose(Minv)
4674        matmatList = [MinvT[i][j] for i in range(4) for j in range(4)]
4675        glLoadMatrixf(matmatList)
4676        lightPosition = [ 0.0,1000.0,0.0,1.0 ]
4677        glLightfv(GL_LIGHT0, GL_POSITION, lightPosition)
4678        glEnable(GL_LIGHTING)
4679        glPushMatrix()
4680        drawTeapot()
4681        glPopMatrix()
4682        glDisable(GL_LIGHTING)
4683        drawCoordinate()
4684        glutSwapBuffers()
4685
```



Note: light position

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

$$\mathbf{p}' = \begin{bmatrix} u_x' & v_x' & w_x' & E_x \\ u_y' & v_y' & w_y' & E_y \\ u_z' & v_z' & w_z' & E_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \mathbf{p}$$

$$\mathbf{w}' = Normalize(\mathbf{E} - \mathbf{A})$$

$$\mathbf{u}' = Normalize(\mathbf{U} \times \mathbf{w}')$$

$$\mathbf{v}' = \mathbf{w}' \times \mathbf{u}'$$

Statement for Matrix operation

w‘ = normalize(300-10,400-20,500-30)
=(0.4325950, 0.5668486, 0.7011022)

u’=(0,0,1)X(0.4325950, 0.5668486, 0.7011022)
=…= (-0.7949512, 0.6066733, 0) →normalized

v’= (-0.42534, -0.5573420, 0.713061)

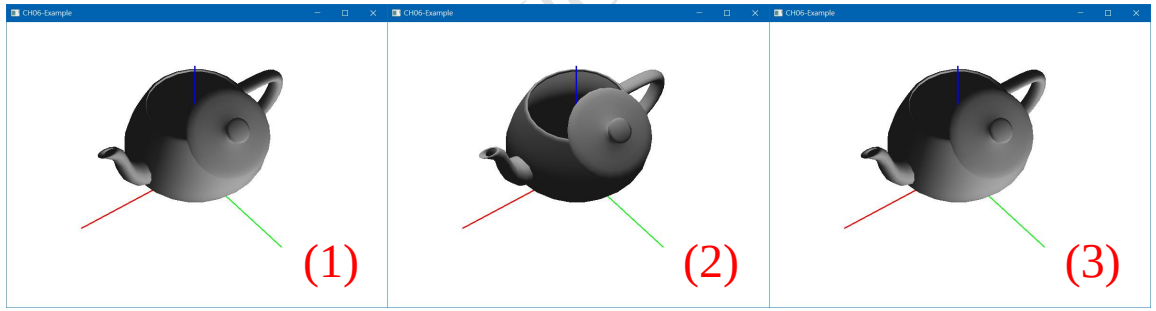
- 0.7949512 - 0.4253400 0.4325950 300.
0.6066733 - 0.5573420 0.5668486 400.
0. 0.7130608 0.7011022 500.
0. 0. 0. 1.

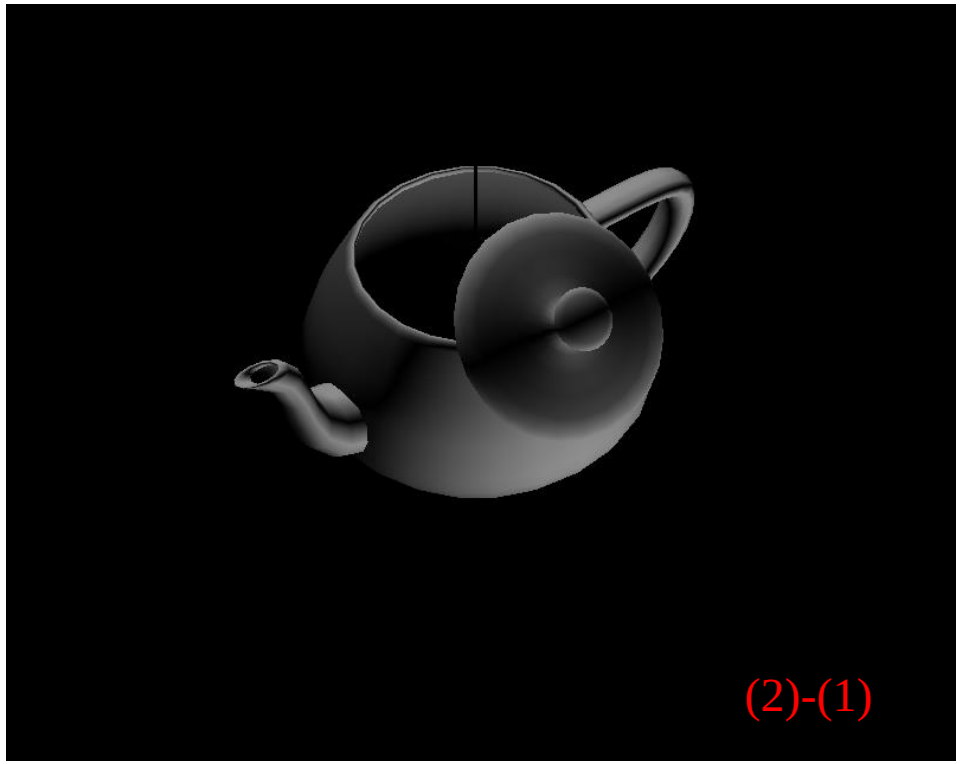
inverse

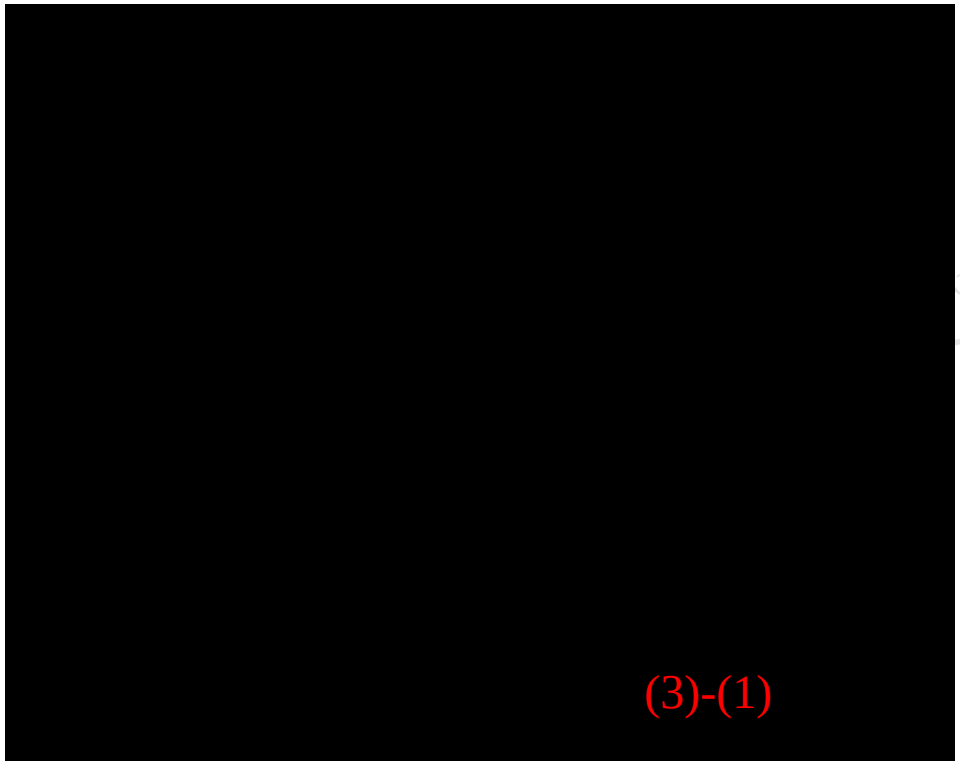
- 0.7949513 0.6066734 - 6.275D-09 - 4.1839541
- 0.4253400 - 0.5573421 0.7130608 - 5.9915833
0.4325950 0.5668486 0.7011022 - 707.06902
0. 0. 0. 1.

gluLookAt vs Coordinate Transformation

- Difference between images

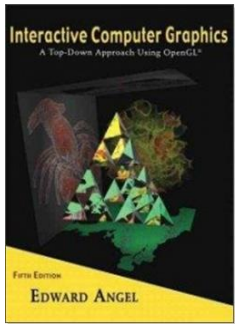




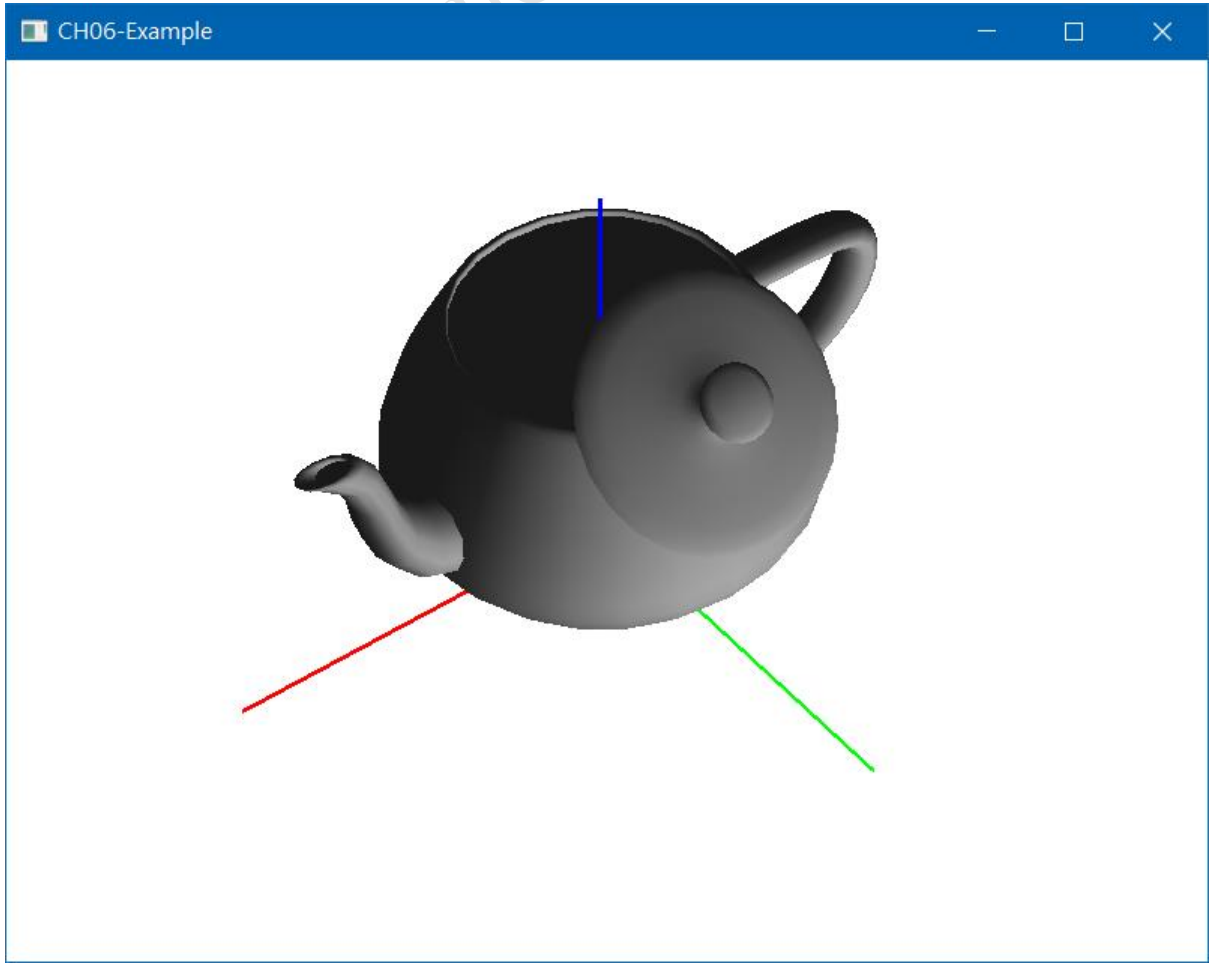


glOrtho vs Projection Matrix

$$\mathbf{P} = \mathbf{ST} = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{left+right}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & -\frac{2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

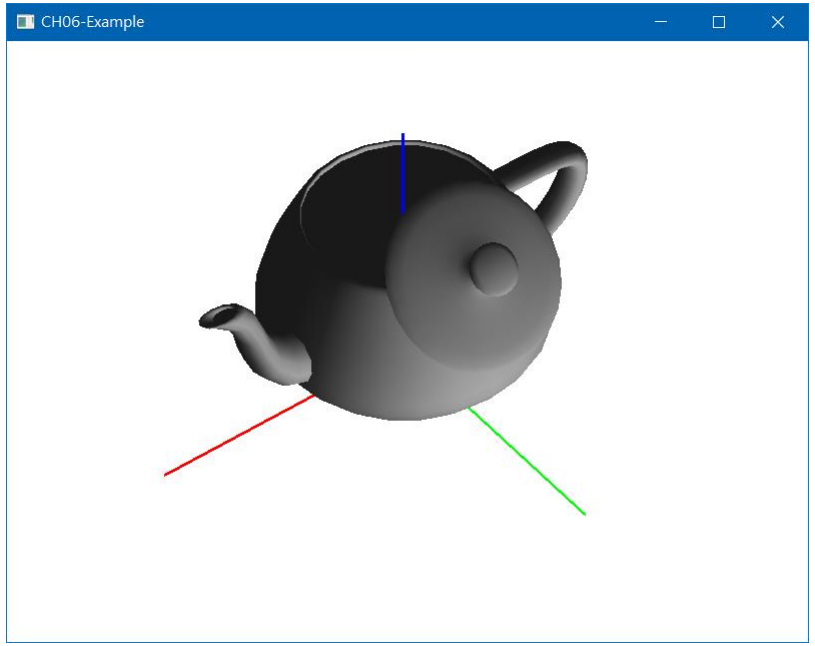



```
4659  def display():
4660      glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
4661      glMatrixMode(GL_PROJECTION)
4662      glLoadIdentity()
4663      glViewport(0, 0, windowWidth, windowHeight)
4664      #glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.0,-float(windowHeight)/2.0,float(windowHeight)/2.0,-
      windowHeight*10.0,windowHeight*10.0)
4665
4666      Mortho = [ [ 2./windowWidth, 0.0, 0.0, -0.0/windowWidth],[ 0.0, 2./windowHeight, 0.0, -0.0/windowHeight],
      [ 0.0, 0.0, -2.0/(20*windowHeight), -0.0/(20*windowHeight)], [0., 0., 0., 1.] ]
4667      MorthoT = np.transpose(Mortho)
4668      matmatList = [Mortho[i][j] for i in range(4) for j in range(4)]
4669      glLoadMatrixf(matmatList)
4670
4671      gluLookAt(300,400,500,10,20,30,0,0,1)
4672      glEnable(GL_LIGHTING)
4673      glPushMatrix()
4674      drawTeapot()
4675      glPopMatrix()
4676      glDisable(GL_LIGHTING)
4677      drawCoordinate()
4678      glutSwapBuffers()
4679
```

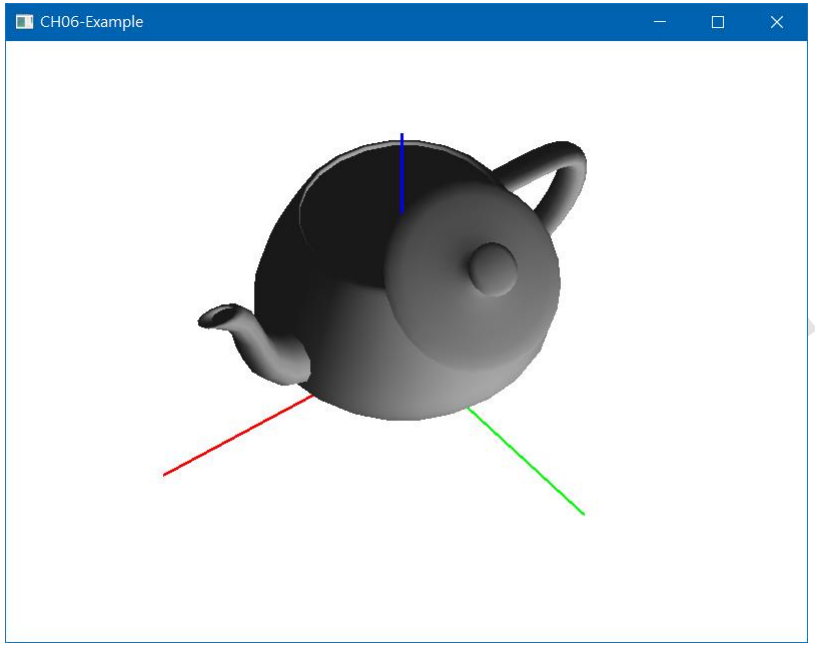


glOrtho vs Projection Matrix

```
4659 def display():
4660     glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
4661     glMatrixMode(GL_PROJECTION)
4662     glLoadIdentity()
4663     glViewport(0, 0, windowWidth, windowHeight)
4664     glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.0,-float(windowHeight)/2.0,float(windowHeight)/2.0,-
windowHeight*10.0,windowHeight*10.0)
4665     '''
4666     Mortho = [ [ 2./windowWidth, 0.0, 0.0, -0.0/windowWidth],[ 0.0, 2./windowHeight, 0.0, -0.0/windowHeight],
[ 0.0, 0.0, -2.0/(20*windowHeight), -0.0/(20*windowHeight)], [0., 0., 0., 1.] ]
4667     MorthoT = np.transpose(Mortho)
4668     matmatlist = [Mortho[i][j] for i in range(4) for j in range(4)]
4669     glLoadMatrixf(matmatlist)
4670     '''
4671
4672     gluLookAt(300,400,500,10,20,30,0,0,1)
4673     glEnable(GL_LIGHTING)
4674     glPushMatrix()
4675     drawTeapot()
4676     glPopMatrix()
4677     glDisable(GL_LIGHTING)
4678     drawCoordinate()
4679     glutSwapBuffers()
```



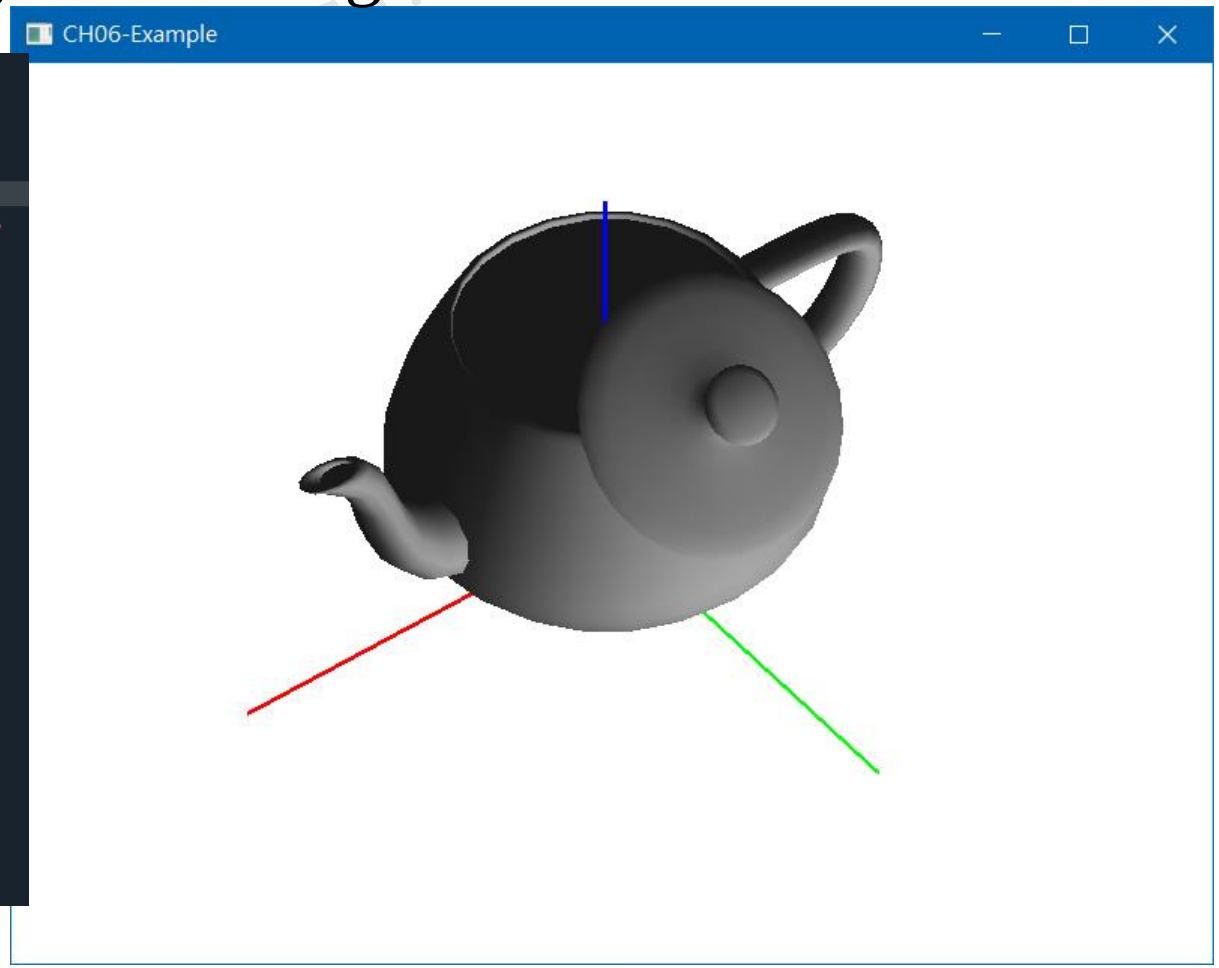

```
4659 def display():
4660     glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
4661     glMatrixMode(GL_PROJECTION)
4662     glLoadIdentity()
4663     glViewport(0, 0, windowWidth, windowHeight)
4664     #glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.0,-float(windowHeight)/2.0,float(windowHeight)/2.0,-
windowHeight*10.0,windowHeight*10.0)
4665
4666     Mortho = [ [ 2./windowWidth, 0.0, 0.0, -0.0/windowWidth],[ 0.0, 2./windowHeight, 0.0, -0.0/windowHeight],
[ 0.0, 0.0, -2.0/(20*windowHeight), -0.0/(20*windowHeight)], [0., 0., 0., 1.] ]
4667     MorthoT = np.transpose(Mortho)
4668     matmatList = [Mortho[i][j] for i in range(4) for j in range(4)]
4669     glLoadMatrixf(matmatList)
4670
4671
4672     gluLookAt(300,400,500,10,20,30,0,0,1)
4673     glEnable(GL_LIGHTING)
4674     glPushMatrix()
4675     drawTeapot()
4676     glPopMatrix()
4677     glDisable(GL_LIGHTING)
4678     drawCoordinate()
4679     glutSwapBuffers()
```



glOrtho vs Projection Matrix (more complex form)

- All using glLoadMatrixf instead of glOrtho and glLookAt

```
4659  def display():
4660      glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
4661      glMatrixMode(GL_PROJECTION)
4662      glLoadIdentity()
4663      glViewport(0, 0, windowWidth, windowHeight)
4664
4665      Mortho = [ [ 2./windowWidth, 0.0, 0.0, -0.0/windowWidth],[ 0.0, 2./windowHeight, 0.0, -0.0/windowHeight],
      [ 0.0, 0.0, -2.0/(20*windowHeight), -0.0/(20*windowHeight)], [0., 0., 0., 1.] ]
4666      MorthoT = np.transpose(Mortho)
4667      matmatList = [Mortho[i][j] for i in range(4) for j in range(4)]
4668      glLoadMatrixf(matmatList)
4669
4670      glMatrixMode(GL_MODELVIEW)
4671      w = np.array([300,400,500])-np.array([10,20,30])
4672      w = w / np.linalg.norm(w)
4673      U = np.array([0,0,1])
4674      u = np.cross(U,w)
4675      u = u / np.linalg.norm(u)
4676      v = np.cross(w,u)
4677      M = [ [u[0], v[0], w[0], 300],[ u[1], v[1], w[1], 400], [u[2], v[2], w[2], 500], [0., 0., 0., 1.] ]
4678      Minv = np.linalg.inv(M)
4679      MinvT = np.transpose(Minv)
4680      matmatList = [MinvT[i][j] for i in range(4) for j in range(4)]
4681      glLoadMatrixf(matmatList)
4682      lightPosition = [ 0.0,1000.0,0.0,1.0 ]
4683      glLightfv(GL_LIGHT0, GL_POSITION, lightPosition)
4684      glEnable(GL_LIGHTING)
4685      glPushMatrix()
4686      drawTeapot()
4687      glPopMatrix()
4688      glDisable(GL_LIGHTING)
4689      drawCoordinate()
4690      glutSwapBuffers()
4691
```

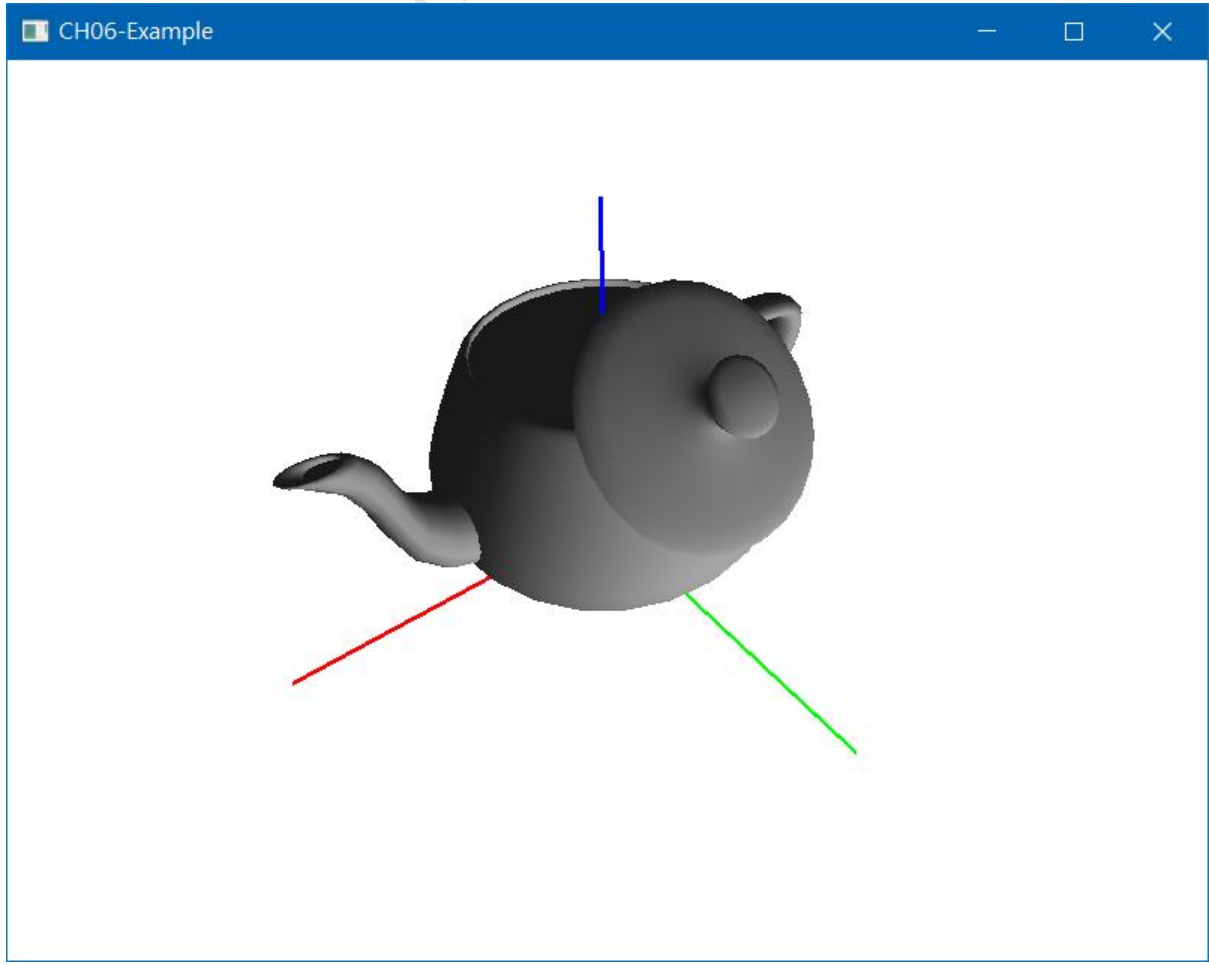


glFrustum vs Projection Matrix

- by glFrustum

glFrustum(-800/1000.0, 800/1000.0,
- 600/1000.0, 600/1000.0, 1.0, 5000)

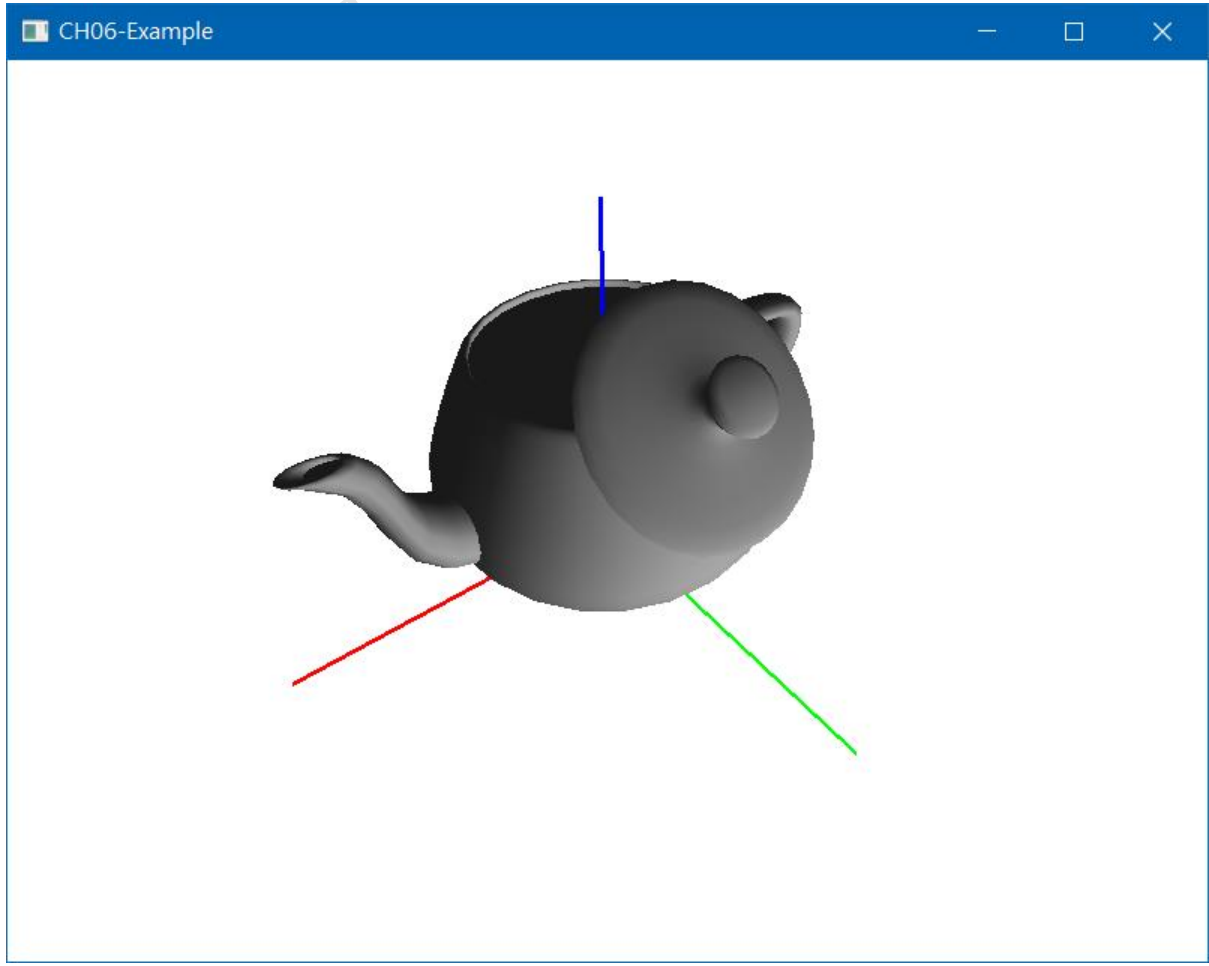
```
4659    def display():
4660        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
4661        glMatrixMode(GL_PROJECTION)
4662        glLoadIdentity()
4663        glViewport(0, 0, windowWidth, windowHeight)
4664        glFrustum(-800/1000.0, 800/1000.0,- 600/1000.0, 600/1000.0, 1.0, 5000);
4665
4666        gluLookAt(300,400,500,10,20,30,0,0,1)
4667        glEnable(GL_LIGHTING)
4668        glPushMatrix()
4669        drawTeapot()
4670        glPopMatrix()
4671        glDisable(GL_LIGHTING)
4672        drawCoordinate()
4673        glutSwapBuffers()
4674
```



glFrustum vs Projection Matrix

- by Projection matrix

```
4659  def display():
4660      glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
4661      glMatrixMode(GL_PROJECTION)
4662      glLoadIdentity()
4663      glViewport(0, 0, windowWidth, windowHeight)
4664
4665      Mfrustum = [ [ 2./(0.8+0.8), 0.0, 0.0, 0.0],[ 0.0, 2./(0.6+0.6), 0.0, 0.0],
      [ 0.0, 0.0, -(5000+1.0)/(5000-1.0), -2.0*1.0*5000./(5000-1.0)], [0., 0., -1.0,
      0.0] ]
4666      MfrustumT = np.transpose(Mfrustum)
4667      matmatList = [MfrustumT[i][j] for i in range(4) for j in range(4)]
4668      glLoadMatrixf(matmatList)
4669
4670      gluLookAt(300,400,500,10,20,30,0,0,1)
4671      glEnable(GL_LIGHTING)
4672      glPushMatrix()
4673      drawTeapot()
4674      glPopMatrix()
4675      glDisable(GL_LIGHTING)
4676      drawCoordinate()
4677      glutSwapBuffers()
4678
```



glFrustum

The **glFrustum** function multiplies the current matrix by a perspective matrix.

```
void glFrustum(
  GLdouble left,
  GLdouble right,
  GLdouble bottom,
  GLdouble top,
  GLdouble znear,
  GLdouble zfar
);
```

Parameters

left, right
The coordinates for the left and right vertical clipping planes.

bottom, top
The coordinates for the bottom and top horizontal clipping planes.

znear, zfar
The distances to the near and far depth clipping planes. Both distances must be positive.

Remarks

The **glFrustum** function describes a perspective matrix that produces a perspective projection. The (*left*, *bottom*, *znear*) and (*right*, *top*, *znear*) parameters specify the points on the near clipping plane that are mapped to the lower-left and upper-right corners of the window, respectively, assuming that the eye is located at (0, 0, 0). The *zfar* parameter specifies the location of the far clipping plane. Both *znear* and *zfar* must be positive. The corresponding matrix is:

$$\begin{pmatrix} \frac{2\,near}{right-left} & 0 & A & 0 \\ 0 & \frac{2\,near}{top-bottom} & B & 0 \\ 0 & 0 & C & D \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

$$A = \frac{right+left}{right-left}$$

$$B = \frac{top+bottom}{top-bottom}$$

$$C = -\frac{far+near}{far-near}$$

$$D = -\frac{2\,far\,near}{far-near}$$

The **glFrustum** function multiplies the current matrix by this matrix, with the result replacing the current matrix. That is, if M is the current matrix and F is the frustum perspective matrix, then **glFrustum** replaces M with M • F.

glFrustum(-800/1000.0, 800/1000.0, -600/1000.0, 600/1000.0, 1.0, 5000)

near : 1
Far : 5000
left: -0.8
right: 0.8
top: 0.6
bottom: -0.6

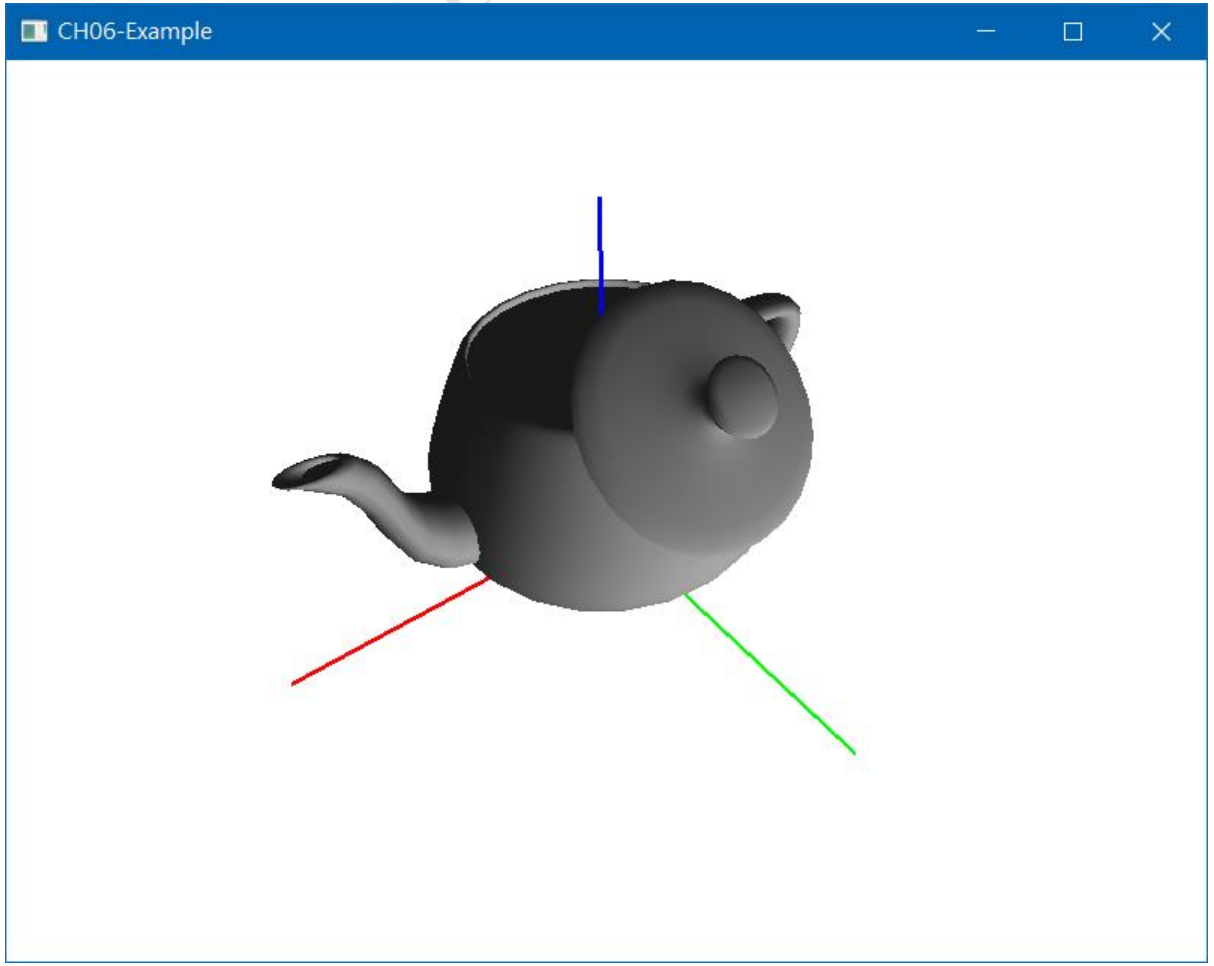
Projection Matrix：
F=[2*1/(0.8+0.8) 0 (0.8-0.8)/(0.8+0.8) 0; 0 2*1/(0.6+0.6) (0.6-0.6)/(0.6+0.6) 0;0 0 -(5000+1)/(5000-1) -2*1*5000/(5000-1); 0 0 -1 0]

F =

1.25	0.	0.	0.
0.	1.67	0.	0.
0.	0.	- 1.0004	- 2.0004
0.	0.	- 1.	0.

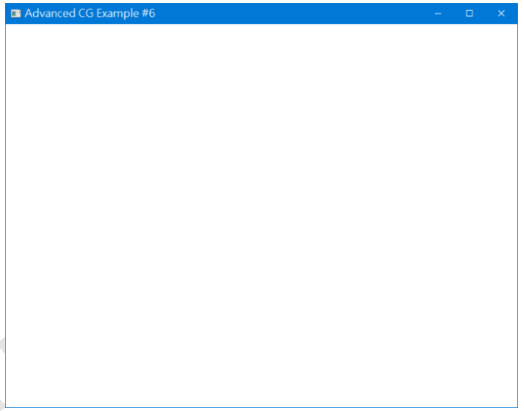
glFrustum vs Projection Matrix

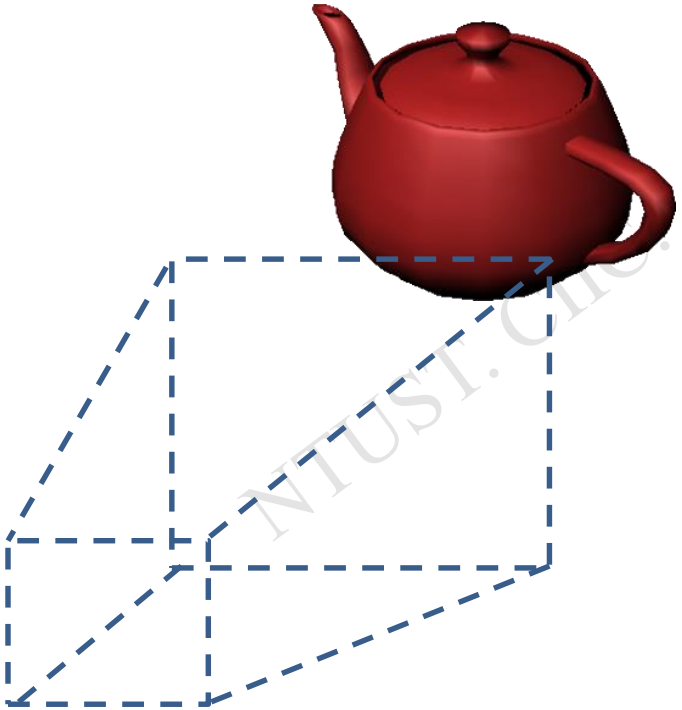
```
4659    def display():
4660        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
4661        glMatrixMode(GL_PROJECTION)
4662        glLoadIdentity()
4663        glViewport(0, 0, windowWidth, windowHeight)
4664
4665        Mfrustum = [ [ 2./(0.8+0.8), 0.0, 0.0, 0.0],[ 0.0, 2./(0.6+0.6), 0.0, 0.0],
        [ 0.0, 0.0, -(5000+1.0)/(5000-1.0), -2.0*1.0*5000./(5000-1.0)], [0., 0., -1.,
        0.0] ]
4666        MfrustumT = np.transpose(Mfrustum)
4667        matmatList = [MfrustumT[i][j] for i in range(4) for j in range(4)]
4668        glLoadMatrixf(matmatList)
4669
4670        glMatrixMode(GL_MODELVIEW)
4671        w = np.array([300,400,500])-np.array([10,20,30])
4672        w = w / np.linalg.norm(w)
4673        U = np.array([0,0,1])
4674        u = np.cross(U,w)
4675        u = u / np.linalg.norm(u)
4676        v = np.cross(w,u)
4677        M = [ [u[0], v[0], w[0], 300],[ u[1], v[1], w[1], 400], [u[2], v[2], w[2],
        500], [0., 0., 0., 1.] ]
4678        Minv = np.linalg.inv(M)
4679        MinvT = np.transpose(Minv)
4680        matmatList = [MinvT[i][j] for i in range(4) for j in range(4)]
4681        glLoadMatrixf(matmatList)
4682        lightPosition = [ 0.0,1000.0,0.0,1.0 ]
4683        glLightfv(GL_LIGHT0, GL_POSITION, lightPosition)
4684
4685        glEnable(GL_LIGHTING)
4686        glPushMatrix()
4687        drawTeapot()
4688        glPopMatrix()
4689        glDisable(GL_LIGHTING)
4690        drawCoordinate()
4691        glutSwapBuffers()
4692
```

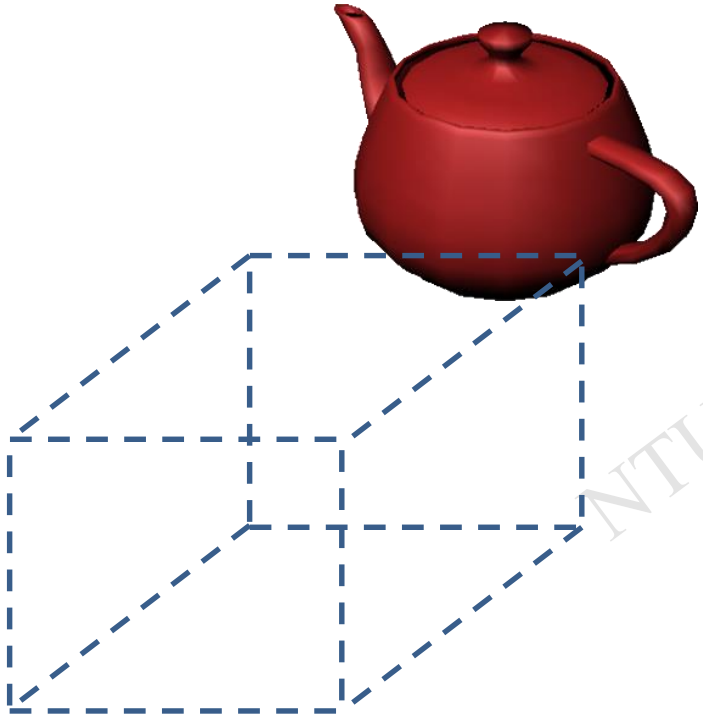


Other issue: See nothing

- Object is not inside the viewing volume



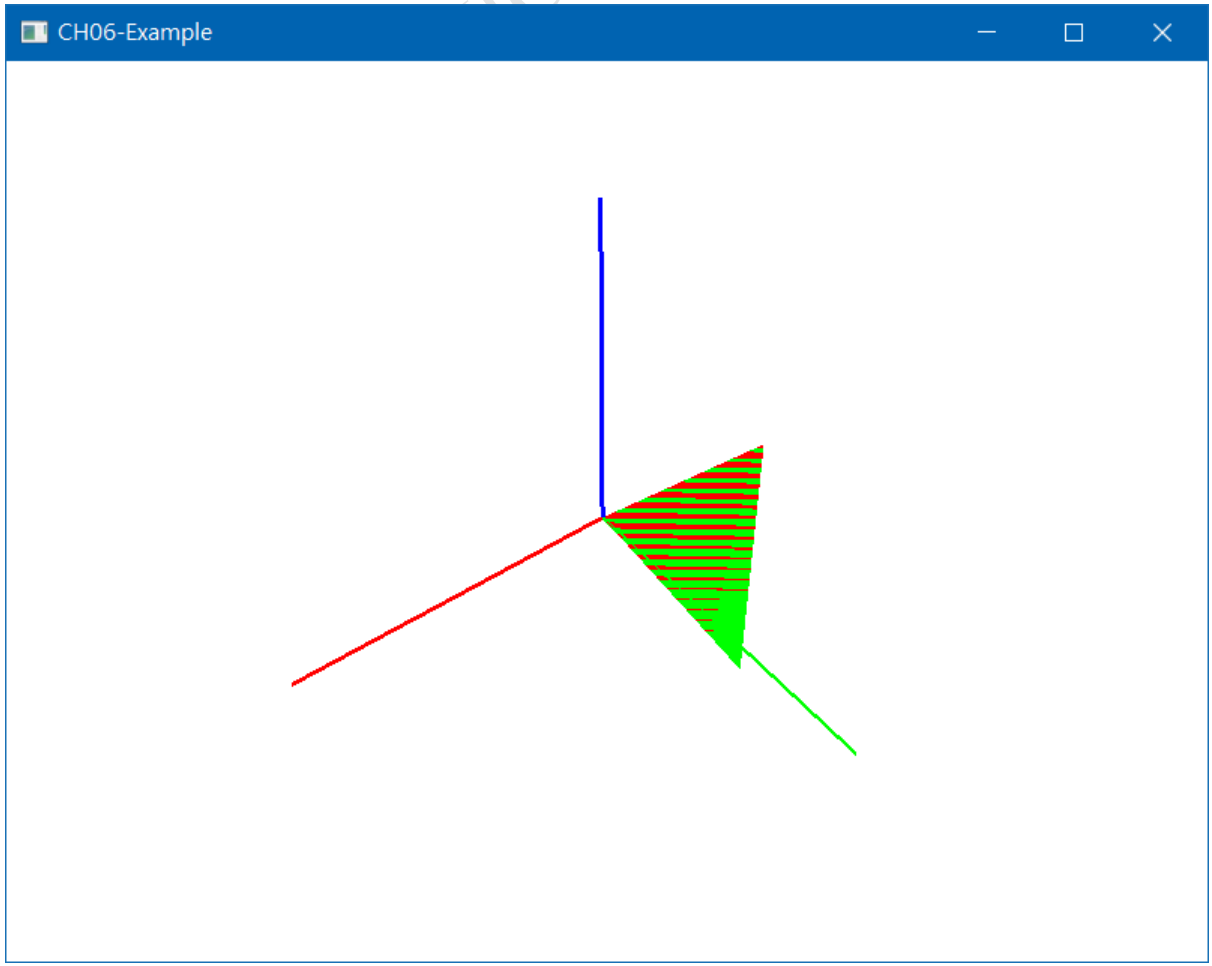




Other issue: Poor depth resolution

```
14  def drawTriangles():
15      glBegin(GL_TRIANGLES)
16      glColor3f(1,0,0)
17      glNormal3f(0.,0.,1.)
18      glVertex3f(0.,0.,0.)
19      glVertex3f(200.,0.,0.)
20      glVertex3f(0.,200.,0.)
21      glColor3f(0,1,0)
22      glNormal3f(0.,0.,1.)
23      glVertex3f(0.,0.,0.01)
24      glVertex3f(200.,0.,0.01)
25      glVertex3f(0.,200.,0.01)
26      glEnd()
```

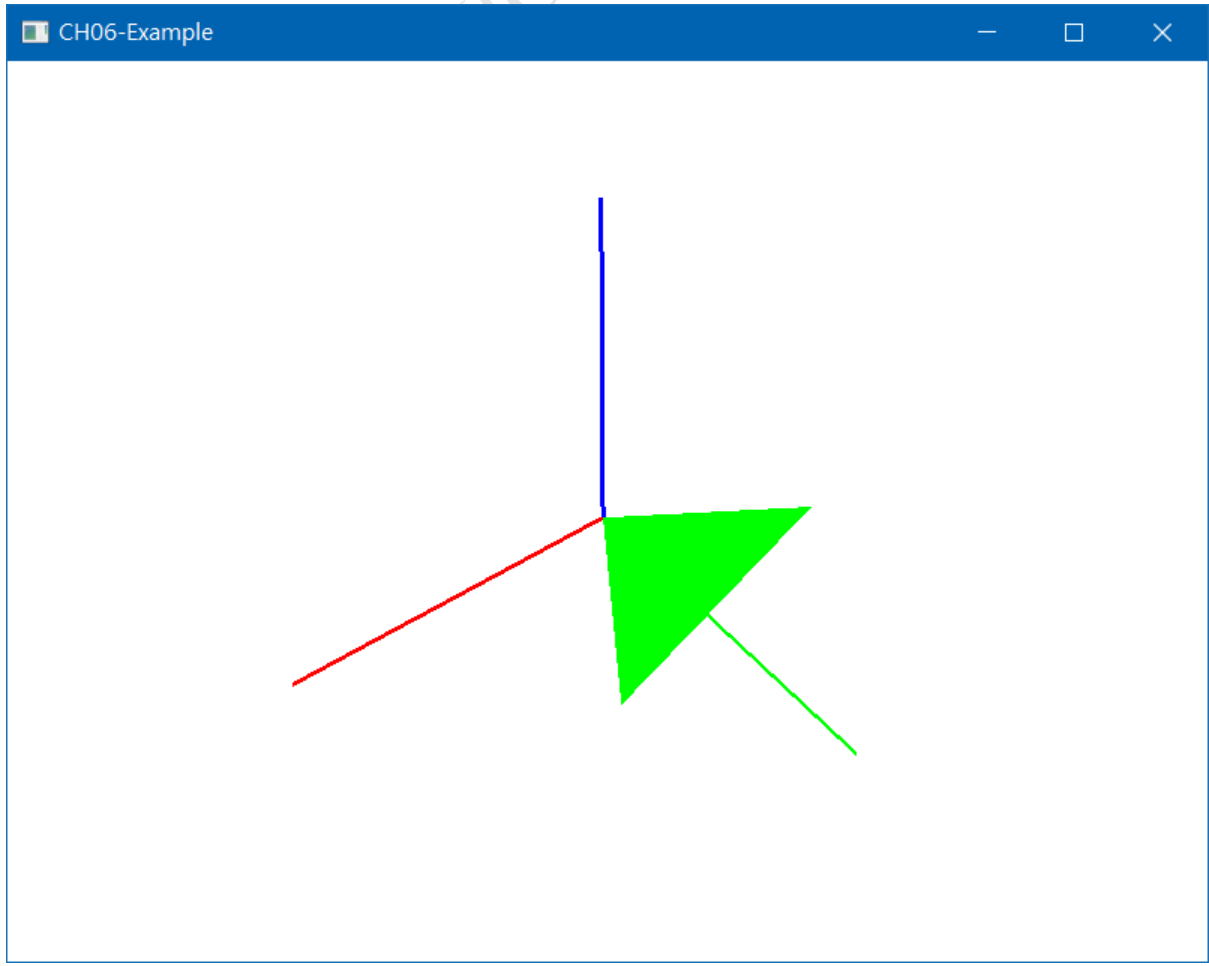
```
43  def display():
44      glClear(GL_COLOR_BUFFER_BIT|
    GL_DEPTH_BUFFER_BIT)
45      glMatrixMode(GL_PROJECTION)
46      glLoadIdentity()
47      glViewport(0, 0, windowWidth, windowHeight)
48      vscale = 1
49      glFrustum(-800/1000.0*vscale,
    800/1000.0*vscale,- 600/1000.0*vscale,
    600/1000.0*vscale, 1.0*vscale, 5000)
50
51      gluLookAt(300,400,500,10,20,30,0,0,1)
52      global angle
53      angle = angle+0.01
54      glPushMatrix()
55      glRotatef(angle,0,0,1)
56      drawTriangles()
57      glPopMatrix()
58      drawCoordinate()
59      glutSwapBuffers()
60      glutPostRedisplay()
```



Other issue: Poor depth resolution

```
14  def drawTriangles():
15      glBegin(GL_TRIANGLES)
16      glColor3f(1,0,0)
17      glNormal3f(0.,0.,1.)
18      glVertex3f(0.,0.,0.)
19      glVertex3f(200.,0.,0.)
20      glVertex3f(0.,200.,0.)
21      glColor3f(0,1,0)
22      glNormal3f(0.,0.,1.)
23      glVertex3f(0.,0.,0.01)
24      glVertex3f(200.,0.,0.01)
25      glVertex3f(0.,200.,0.01)
26      glEnd()
```

```
43  def display():
44      glClear(GL_COLOR_BUFFER_BIT|
    GL_DEPTH_BUFFER_BIT)
45      glMatrixMode(GL_PROJECTION)
46      glLoadIdentity()
47      glViewport(0, 0, windowWidth, windowHeight)
48      vscale = 10
49      glFrustum(-800/1000.0*vscale,
    800/1000.0*vscale,- 600/1000.0*vscale,
    600/1000.0*vscale, 1.0*vscale, 5000)
50
51      gluLookAt(300,400,500,10,20,30,0,0,1)
52      global angle
53      angle = angle+0.01
54      glPushMatrix()
55      glRotatef(angle,0,0,1)
56      drawTriangles()
57      glPopMatrix()
58      drawCoordinate()
59      glutSwapBuffers()
60      glutPostRedisplay()
61
```



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

Find us on:
facebook