Advanced Computer Graphics

Lecture-08 Introduction to OpenGL-5

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

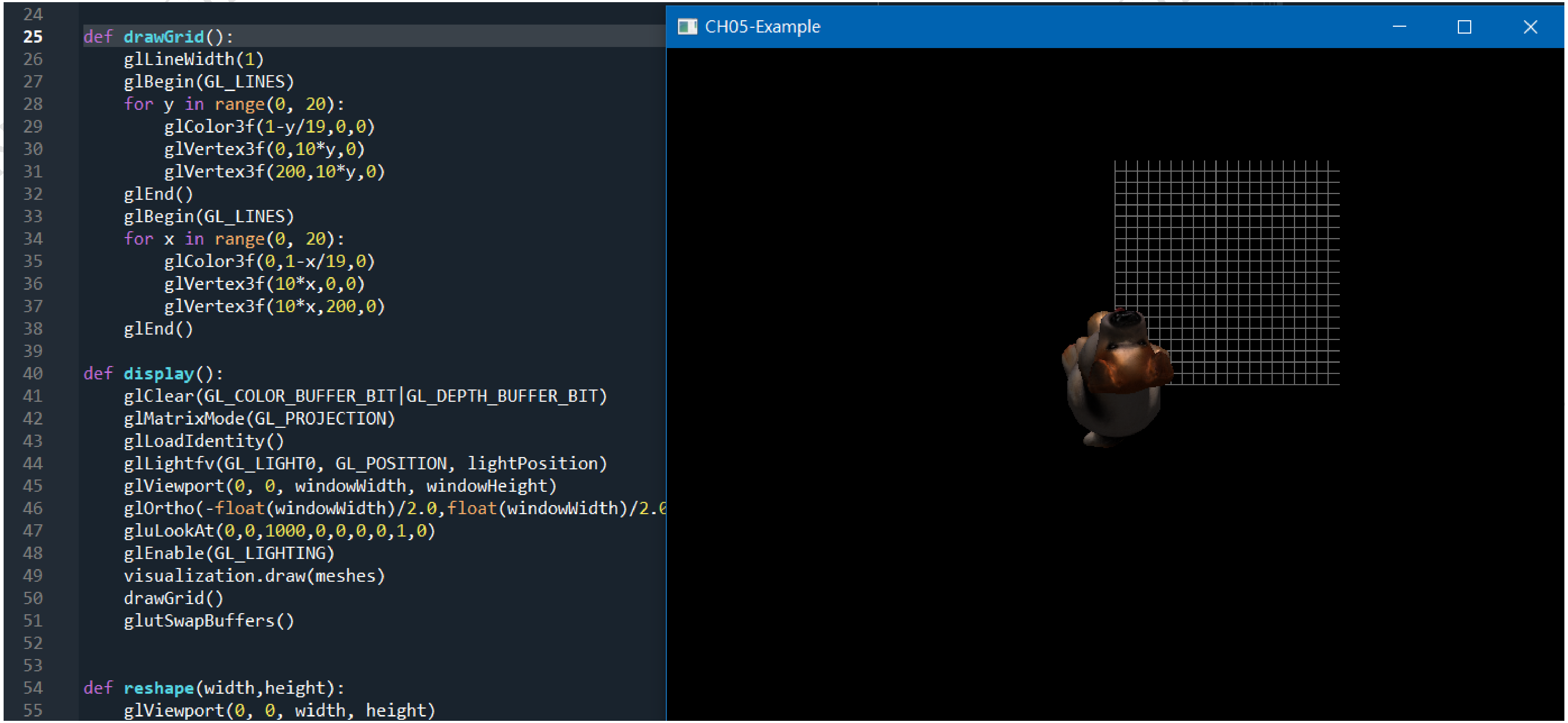
TAIWAN TECH National Taiwan University of Science and Technology

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

Camera/Viewing control

Difference between shading color and solid color

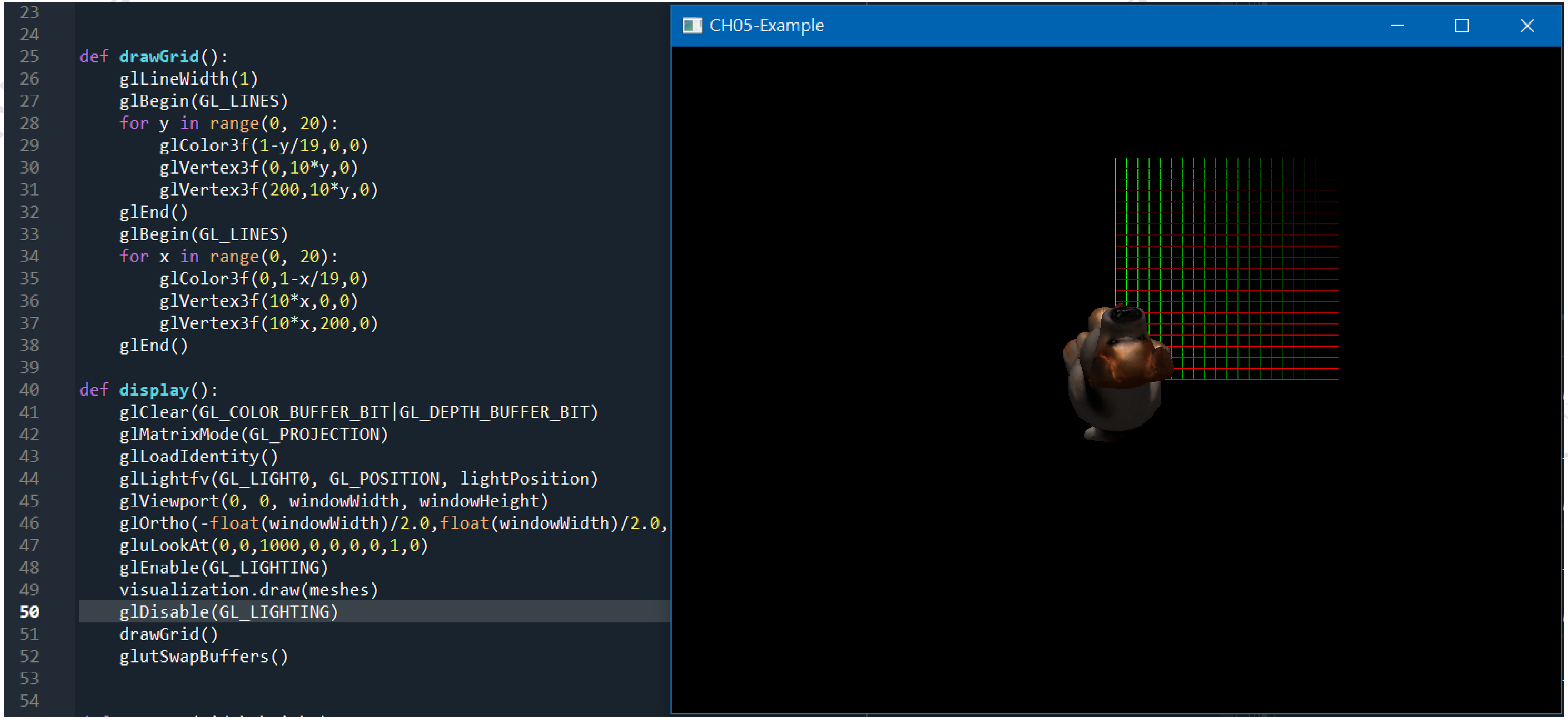
Note: deal with light carefully



```
def drawGrid():
    glLineWidth(1)
    glBegin(GL_LINES)
    for y in range(0, 20):
        glColor3f(1-y/19,0,0)
        glVertex3f(0,10*y,0)
        glVertex3f(200,10*y,0)
    glEnd()
    glBegin(GL_LINES)
    for x in range(0, 20):
        glColor3f(0,1-x/19,0)
        glVertex3f(10*x,0,0)
        glVertex3f(10*x,200,0)
    glEnd()

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


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

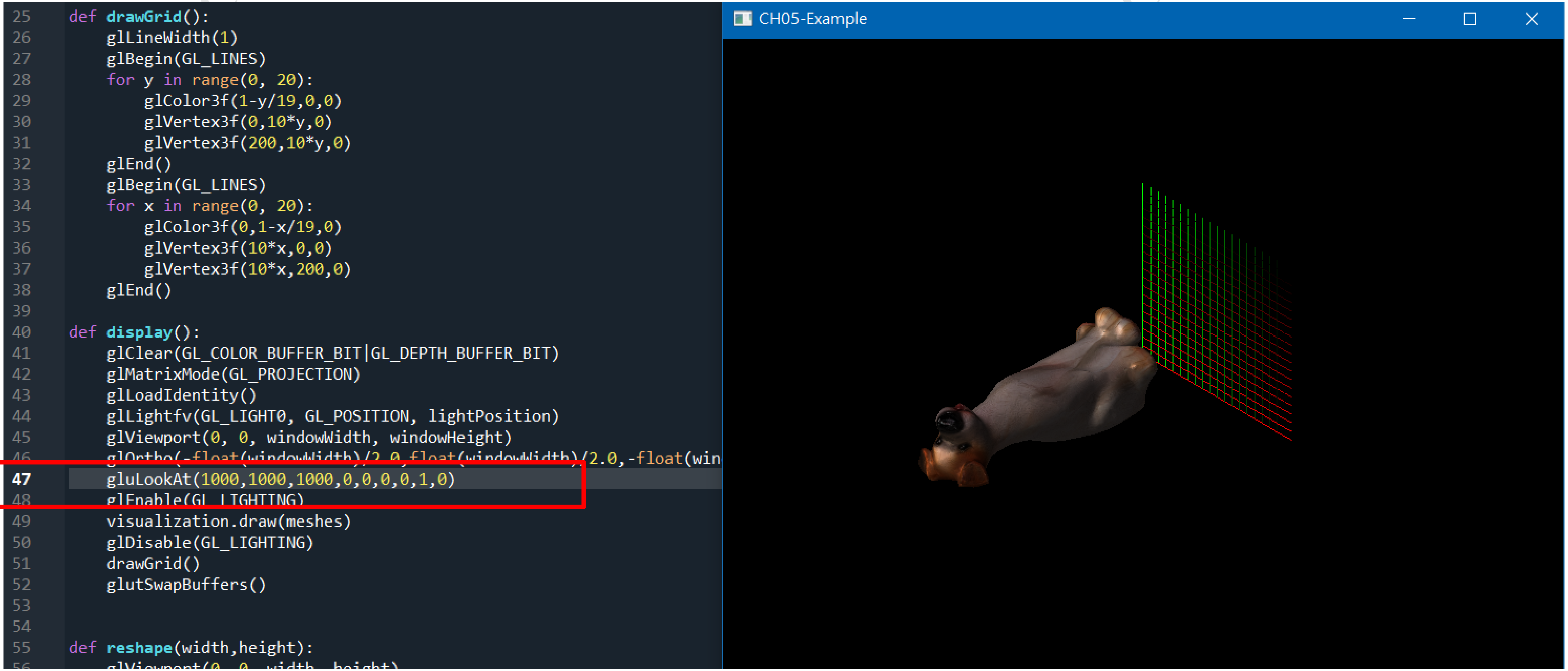
Difference between shading color and solid color

Note: deal with light carefully



```
def drawGrid():
    glLineWidth(1)
    glBegin(GL_LINES)
    for y in range(0, 20):
        glColor3f(1-y/19,0,0)
        glVertex3f(0,10*y,0)
        glVertex3f(200,10*y,0)
    glEnd()
    glBegin(GL_LINES)
    for x in range(0, 20):
        glColor3f(0,1-x/19,0)
        glVertex3f(10*x,0,0)
        glVertex3f(10*x,200,0)
    glEnd()

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

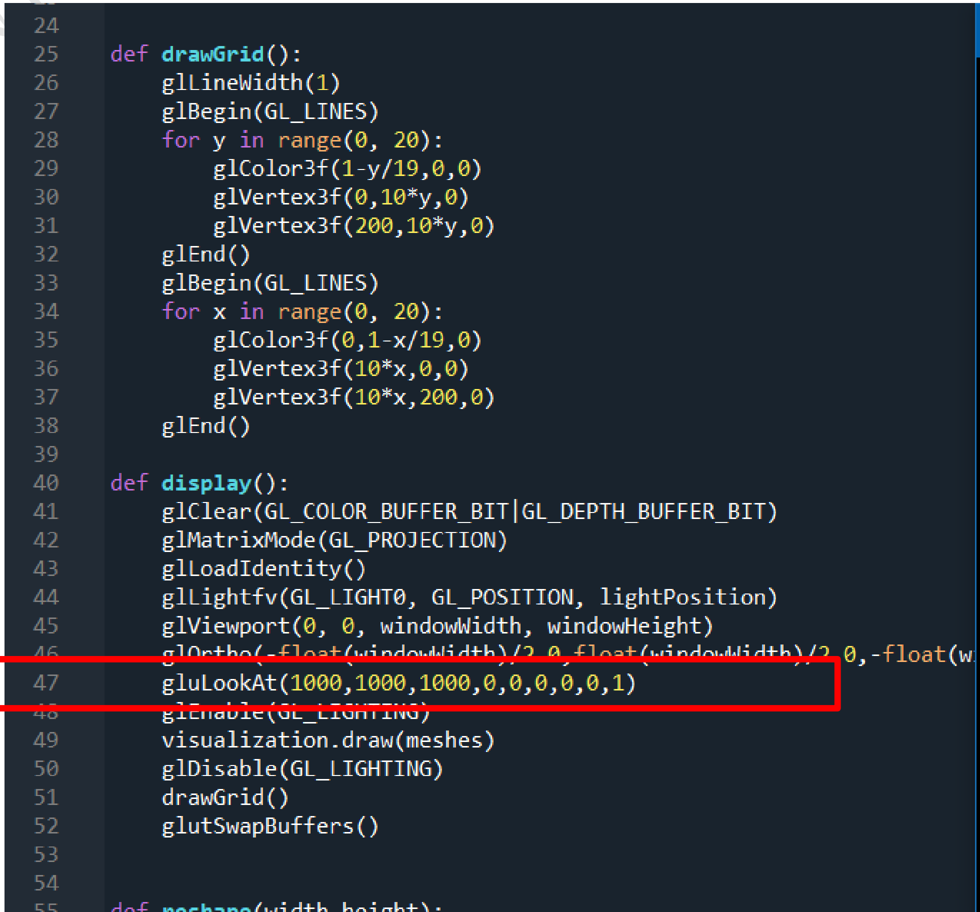
Control the position of camera



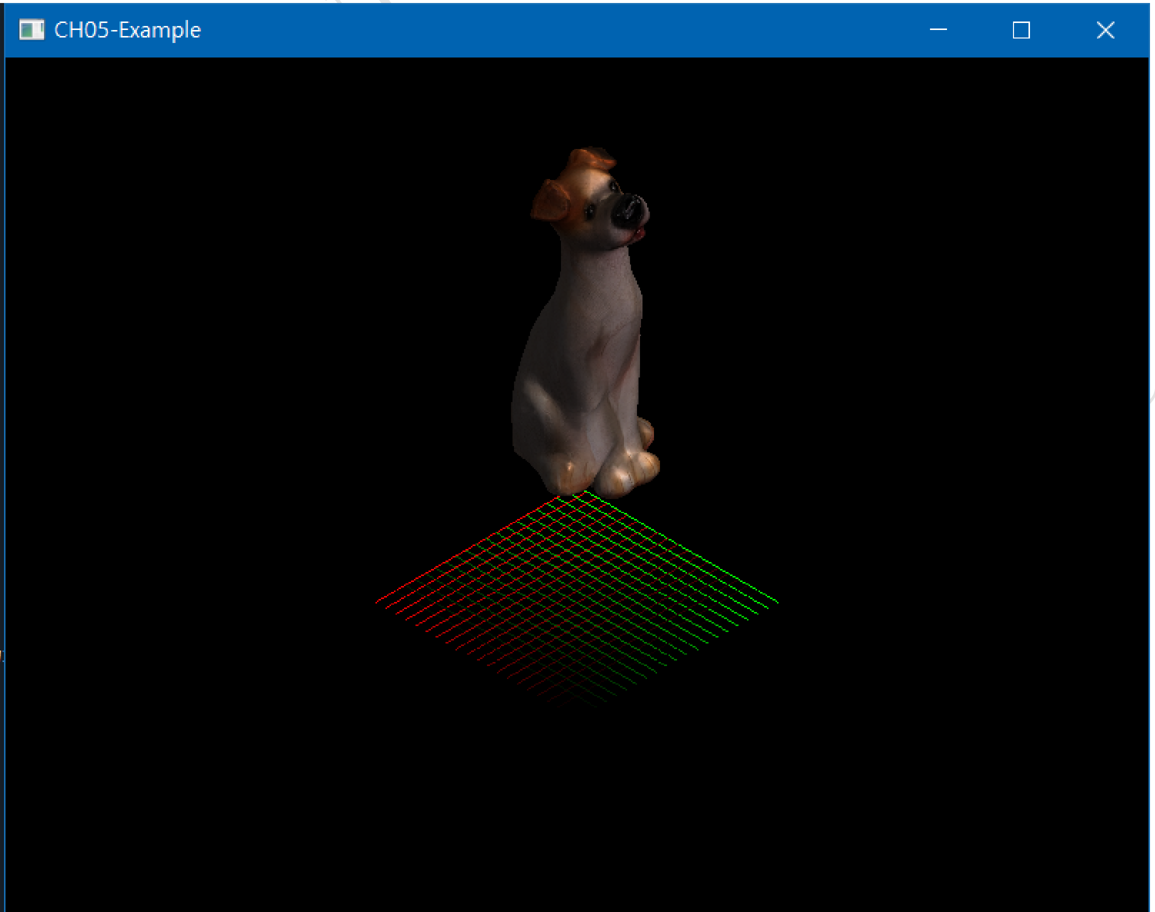
```
def drawGrid():
    glLineWidth(1)
    glBegin(GL_LINES)
    for y in range(0, 20):
        glColor3f(1-y/19,0,0)
        glVertex3f(0,10*y,0)
        glVertex3f(200,10*y,0)
    glEnd()
    glBegin(GL_LINES)
    for x in range(0, 20):
        glColor3f(0,1-x/19,0)
        glVertex3f(10*x,0,0)
        glVertex3f(10*x,200,0)
    glEnd()

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


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

Control the position of camera (Note: up vector)

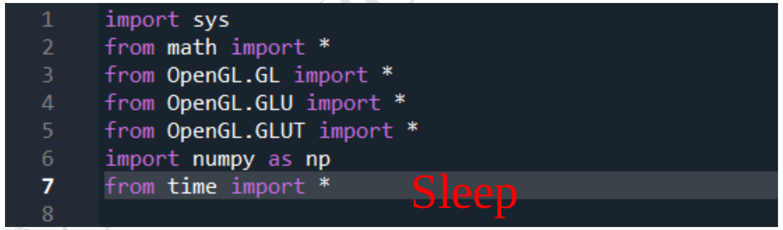
- Note: changing position of camera is always regarding to three 3D vectors!



```

def drawGrid():
    glLineWidth(1)
    glBegin(GL_LINES)
    for y in range(0, 20):
        glColor3f(1-y/19,0,0)
        glVertex3f(0,10*y,0)
        glVertex3f(200,10*y,0)
    glEnd()
    glBegin(GL_LINES)
    for x in range(0, 20):
        glColor3f(0,1-x/19,0)
        glVertex3f(10*x,0,0)
        glVertex3f(10*x,200,0)
    glEnd()

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


def reshape(width,height):
```

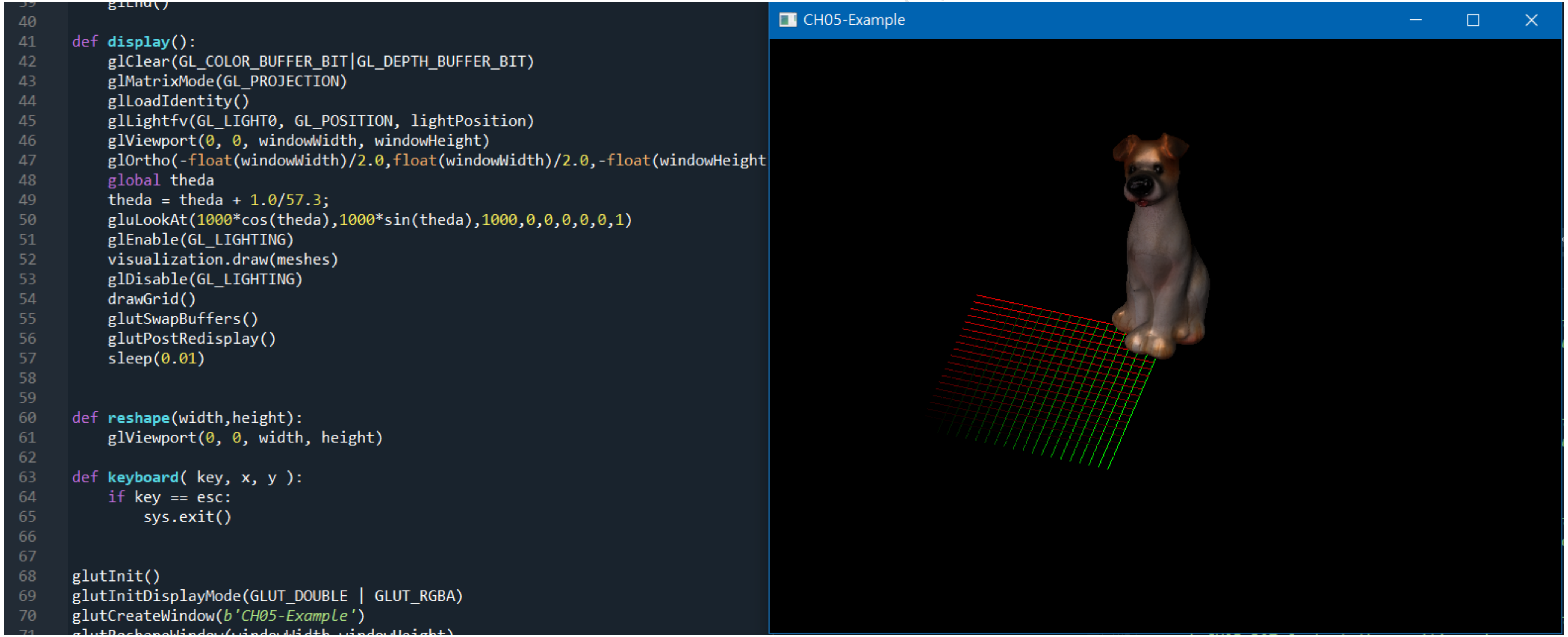


Control the position of camera: dynamic



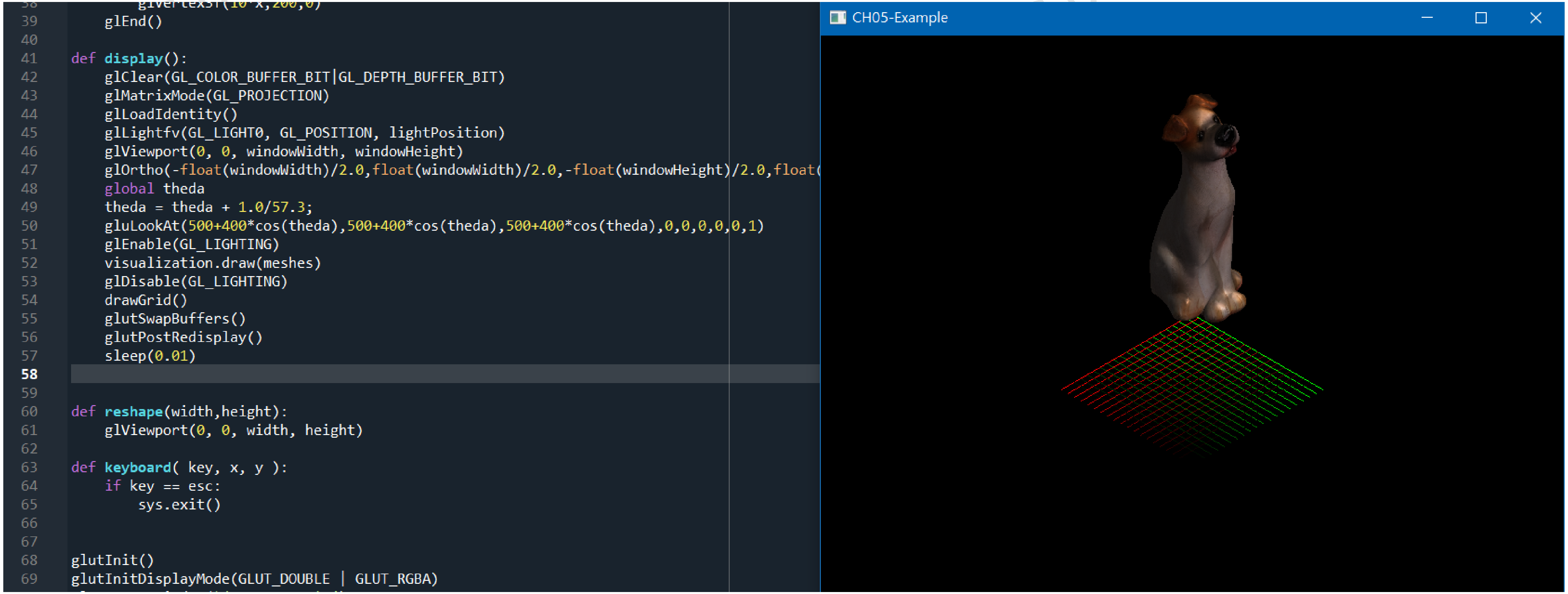
```python
import sys
from math import *
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *
import numpy as np
from time import *
```

Sleep



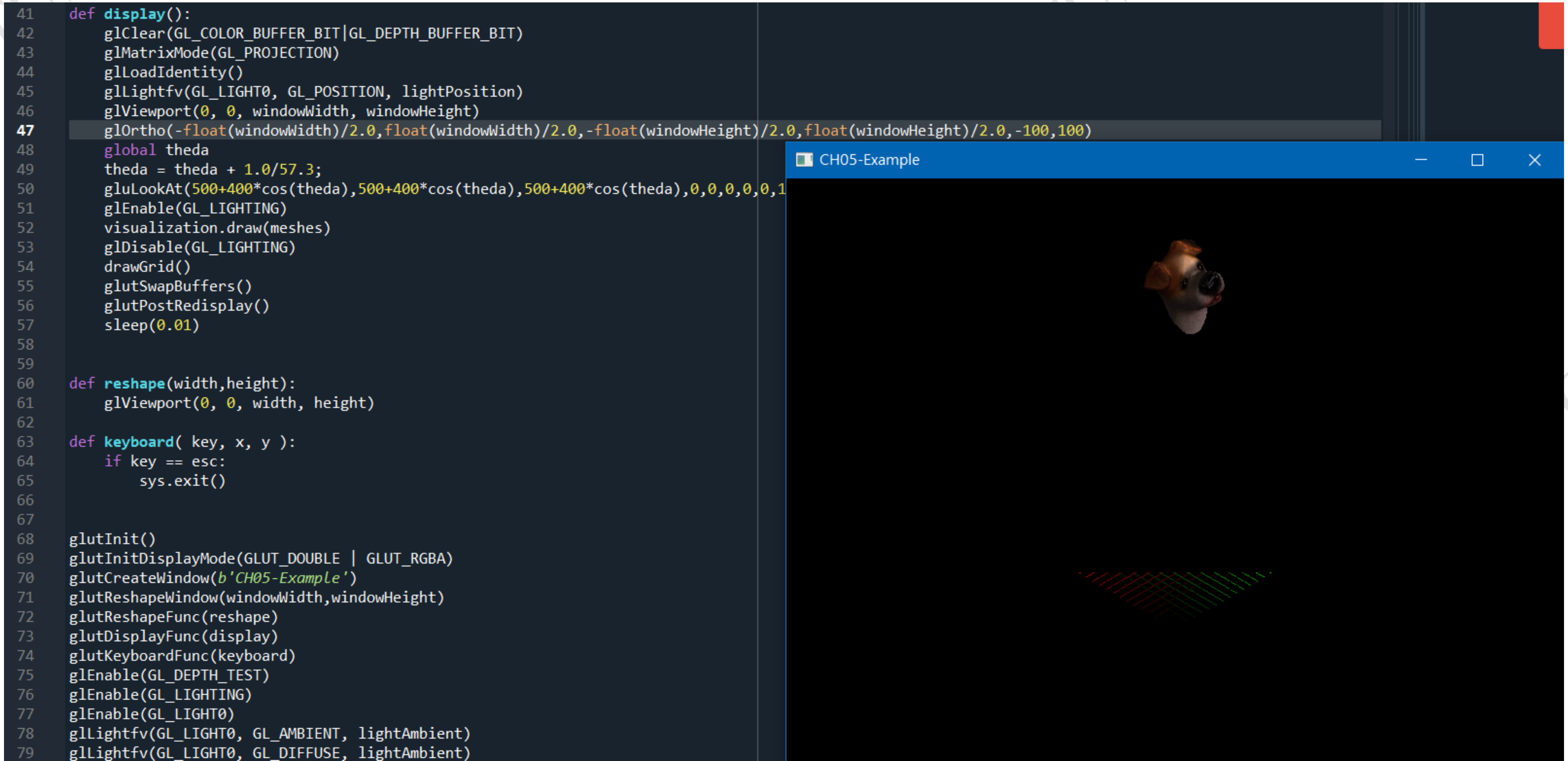
```python
    glEnd()

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
    global theda
    theda = theda + 1.0/57.3;
    gluLookAt(1000*cos(theda),1000*sin(theda),1000,0,0,0,0,0,1)
    glEnable(GL_LIGHTING)
    visualization.draw(meshes)
    glDisable(GL_LIGHTING)
    drawGrid()
    glutSwapBuffers()
    glutPostRedisplay()
    sleep(0.01)


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

def keyboard( key, x, y ):
    if key == esc:
        sys.exit()


glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA)
glutCreateWindow(b'CH05-Example')
glutReshapeWindow(windowWidth,windowHeight)
```

Control the position of camera (distance changes)



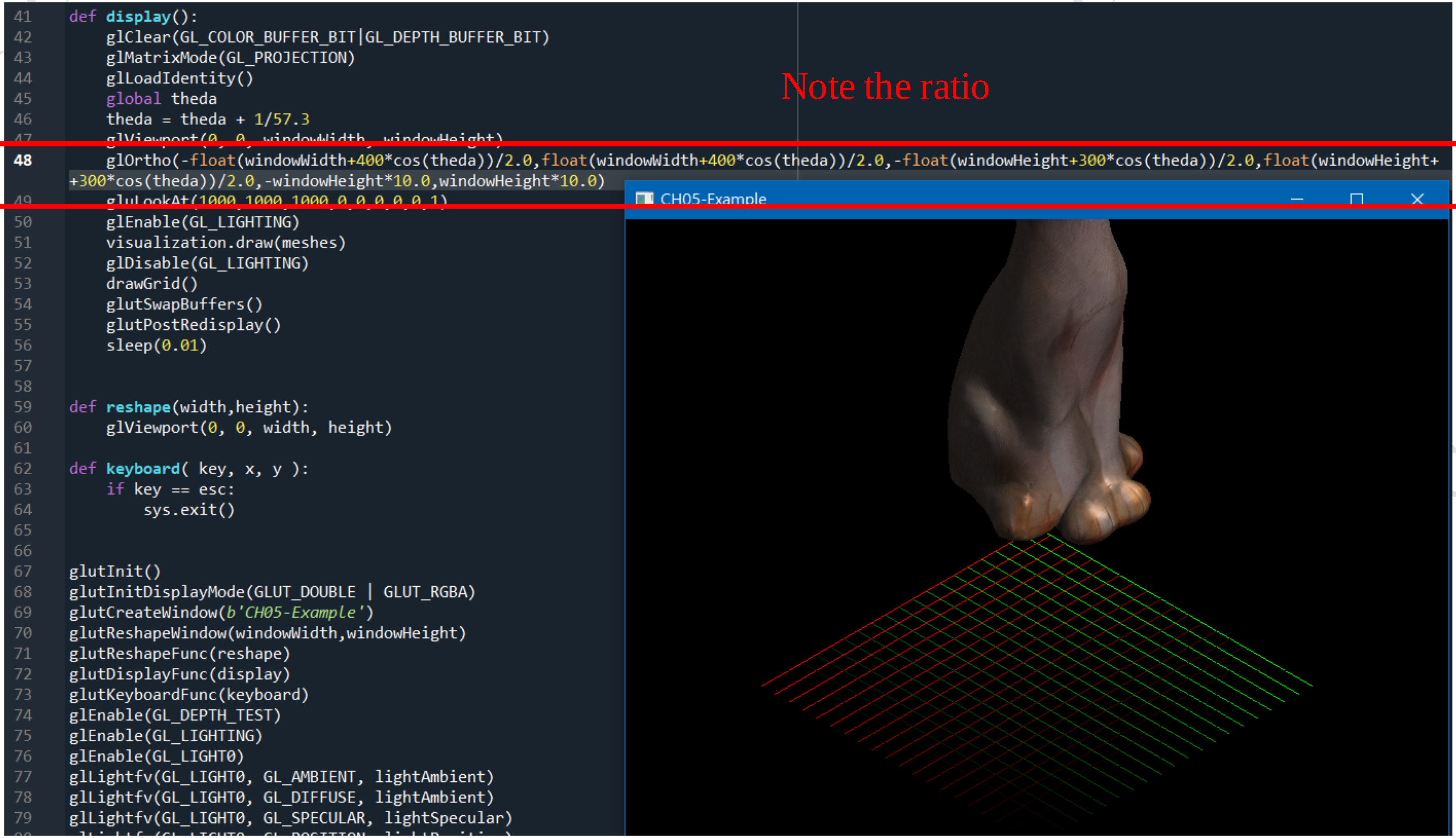
```
        glVertex3f(10*x,200,0)
    glEnd()

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(
    global theda
    theda = theda + 1.0/57.3;
    gluLookAt(500+400*cos(theda),500+400*cos(theda),500+400*cos(theda),0,0,0,0,0,1)
    glEnable(GL_LIGHTING)
    visualization.draw(meshes)
    glDisable(GL_LIGHTING)
    drawGrid()
    glutSwapBuffers()
    glutPostRedisplay()
    sleep(0.01)


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

def keyboard( key, x, y ):
    if key == esc:
        sys.exit()


glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA)
```

Control the position of camera distance changes-small view volume



```
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,-100,100)
    global theda
    theda = theda + 1.0/57.3;
    gluLookAt(500+400*cos(theda),500+400*cos(theda),500+400*cos(theda),0,0,0,0,0,1
    glEnable(GL_LIGHTING)
    visualization.draw(meshes)
    glDisable(GL_LIGHTING)
    drawGrid()
    glutSwapBuffers()
    glutPostRedisplay()
    sleep(0.01)


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

def keyboard( key, x, y ):
    if key == esc:
        sys.exit()


glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA)
glutCreateWindow(b'CH05-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)
```

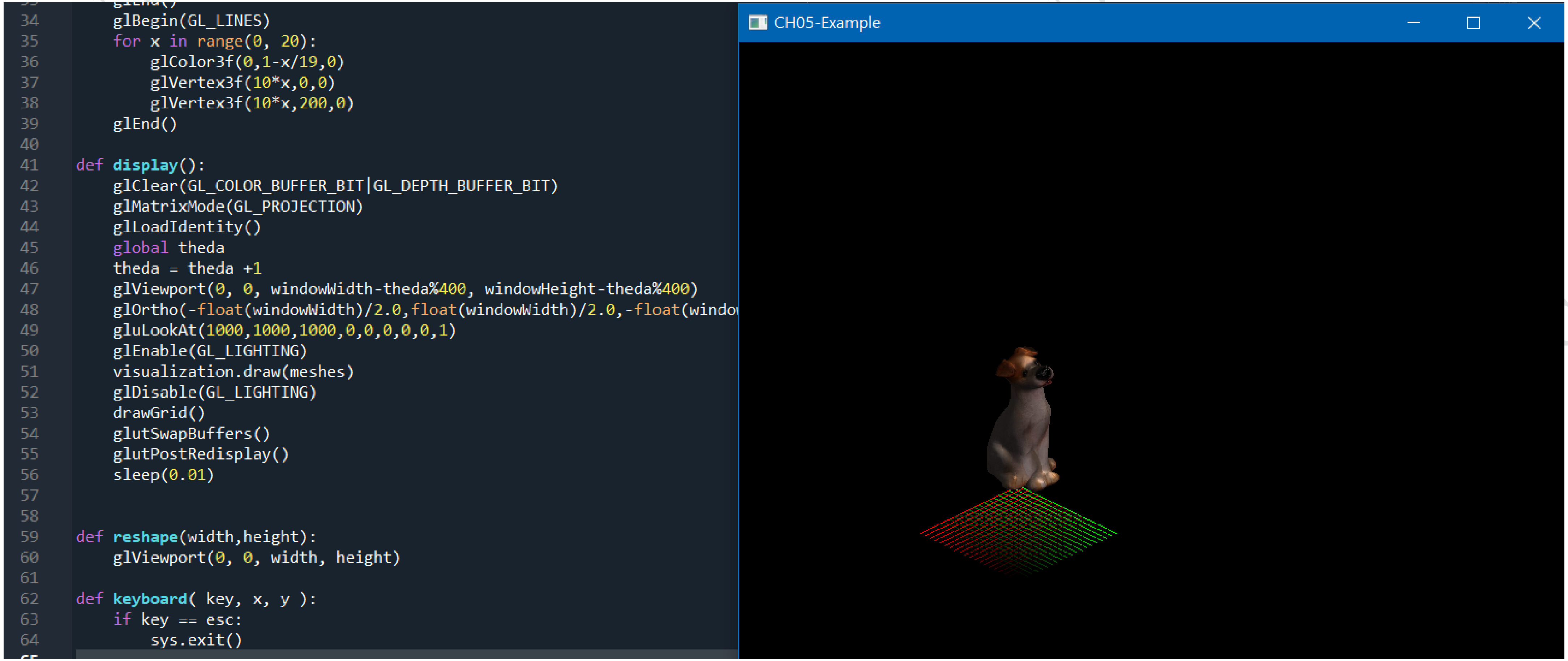
Change viewing size - by viewing volume (Orthographic projection)



```
def display():
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    global theda
    theda = theda + 1/57.3
    glViewport(0, 0, windowWidth, windowHeight)
    glOrtho(-float(windowWidth+400*cos(theda))/2.0,float(windowWidth+400*cos(theda))/2.0,-float(windowHeight+300*cos(theda))/2.0,float(windowHeight+300*cos(theda))/2.0,-windowHeight*10.0,windowHeight*10.0)
    gluLookAt(1000,1000,1000,0,0,0,0,0,1)
    glEnable(GL_LIGHTING)
    visualization.draw(meshes)
    glDisable(GL_LIGHTING)
    drawGrid()
    glutSwapBuffers()
    glutPostRedisplay()
    sleep(0.01)


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

def keyboard( key, x, y ):
    if key == esc:
        sys.exit()


glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA)
glutCreateWindow(b'CH05-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)
```

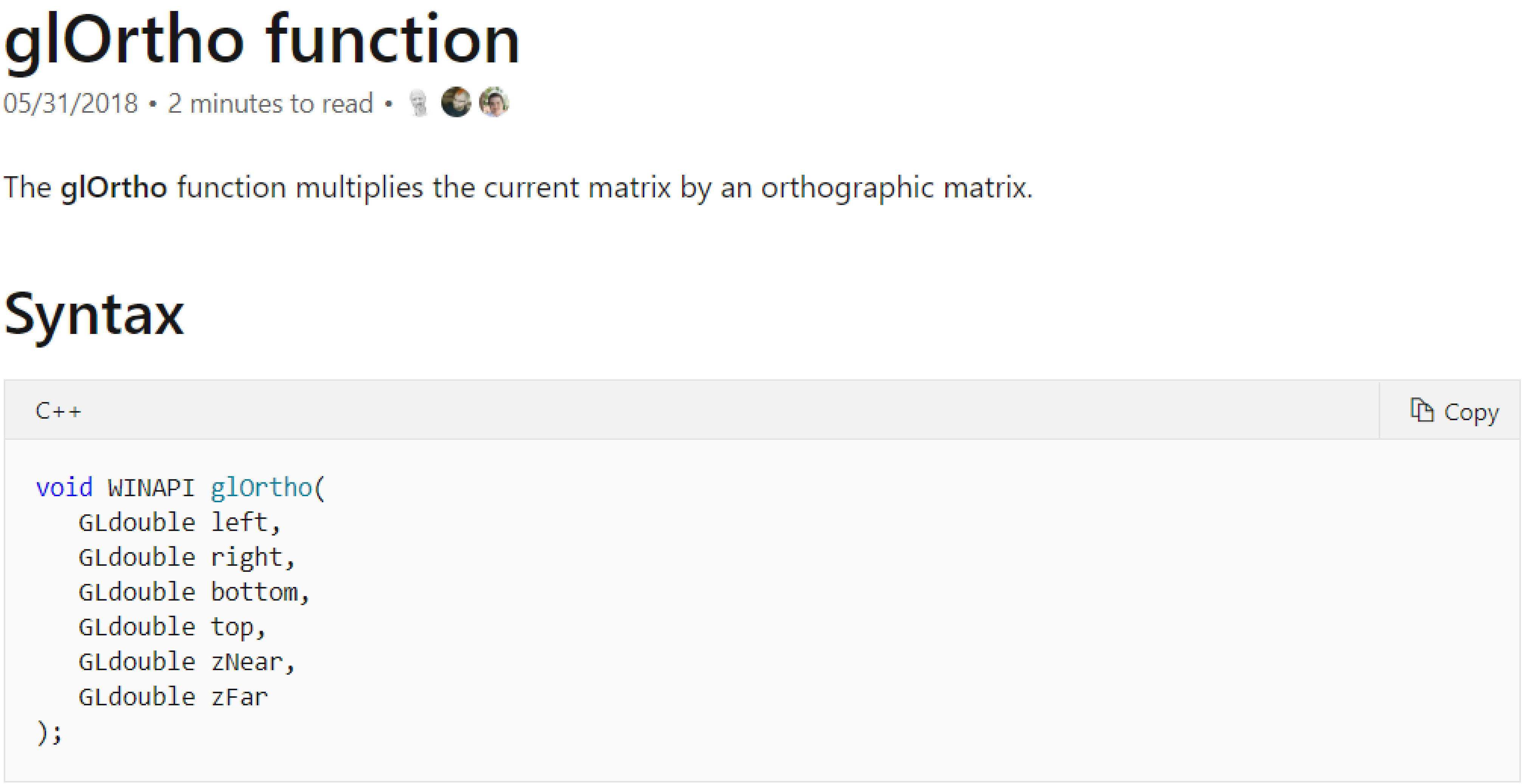
Note the ratio

Change viewing size - by viewport (Not recommended)



```
    glEnd()
    glBegin(GL_LINES)
    for x in range(0, 20):
        glColor3f(0,1-x/19,0)
        glVertex3f(10*x,0,0)
        glVertex3f(10*x,200,0)
    glEnd()

def display():
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    global theda
    theda = theda +1
    glViewport(0, 0, windowWidth-theda%400, windowHeight-theda%400)
    glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.0,-float(windo
    gluLookAt(1000,1000,1000,0,0,0,0,0,1)
    glEnable(GL_LIGHTING)
    visualization.draw(meshes)
    glDisable(GL_LIGHTING)
    drawGrid()
    glutSwapBuffers()
    glutPostRedisplay()
    sleep(0.01)


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

def keyboard( key, x, y ):
    if key == esc:
        sys.exit()
```

Control the position of camera orthographic mode



glOrtho function

05/31/2018 • 2 minutes to read •

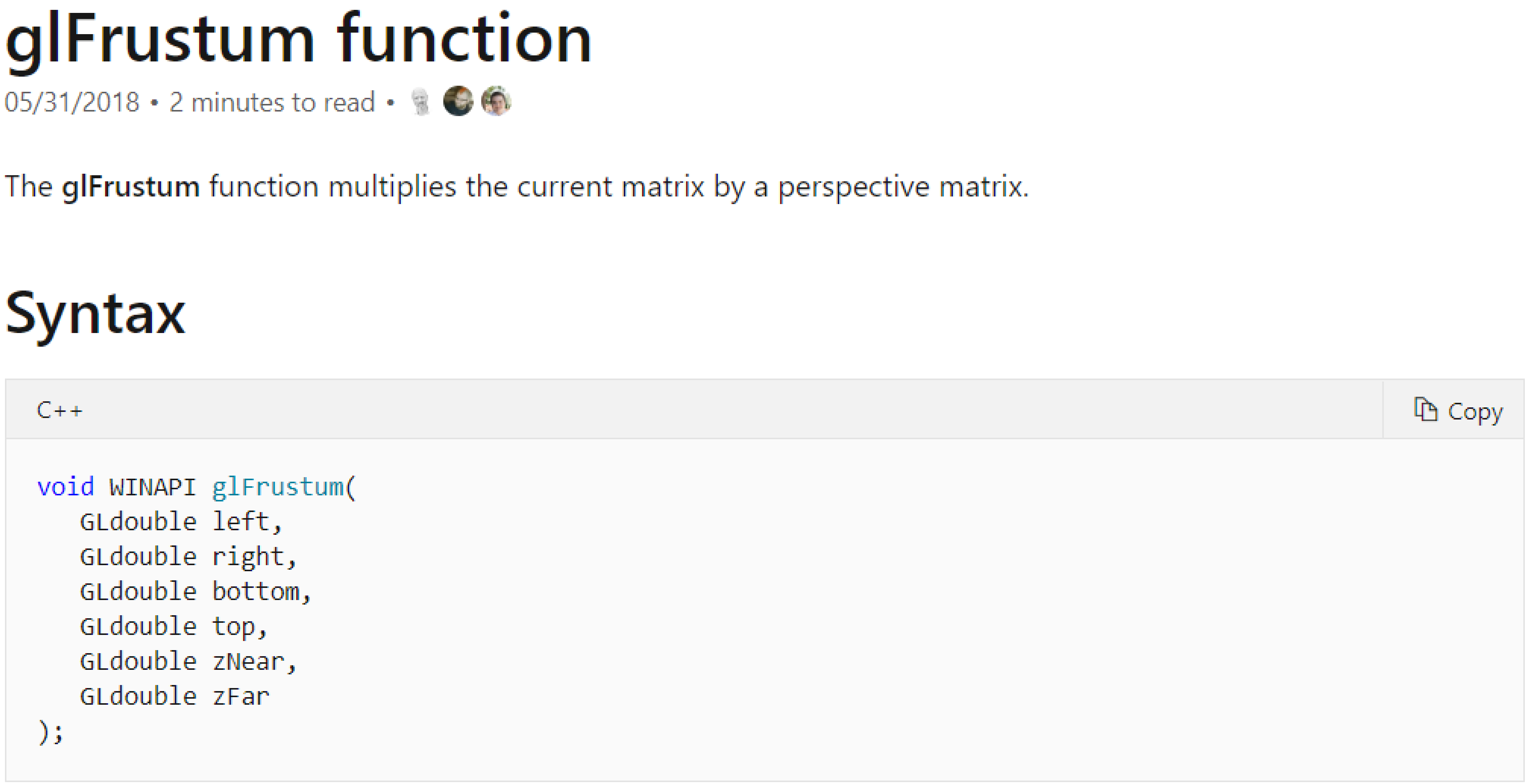
The **glOrtho** function multiplies the current matrix by an orthographic matrix.

Syntax

```cpp
void WINAPI glOrtho(
   GLdouble left,
   GLdouble right,
   GLdouble bottom,
   GLdouble top,
   GLdouble zNear,
   GLdouble zFar
);
```

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

Control the position of camera perspective mode



glFrustum function

05/31/2018 • 2 minutes to read •

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

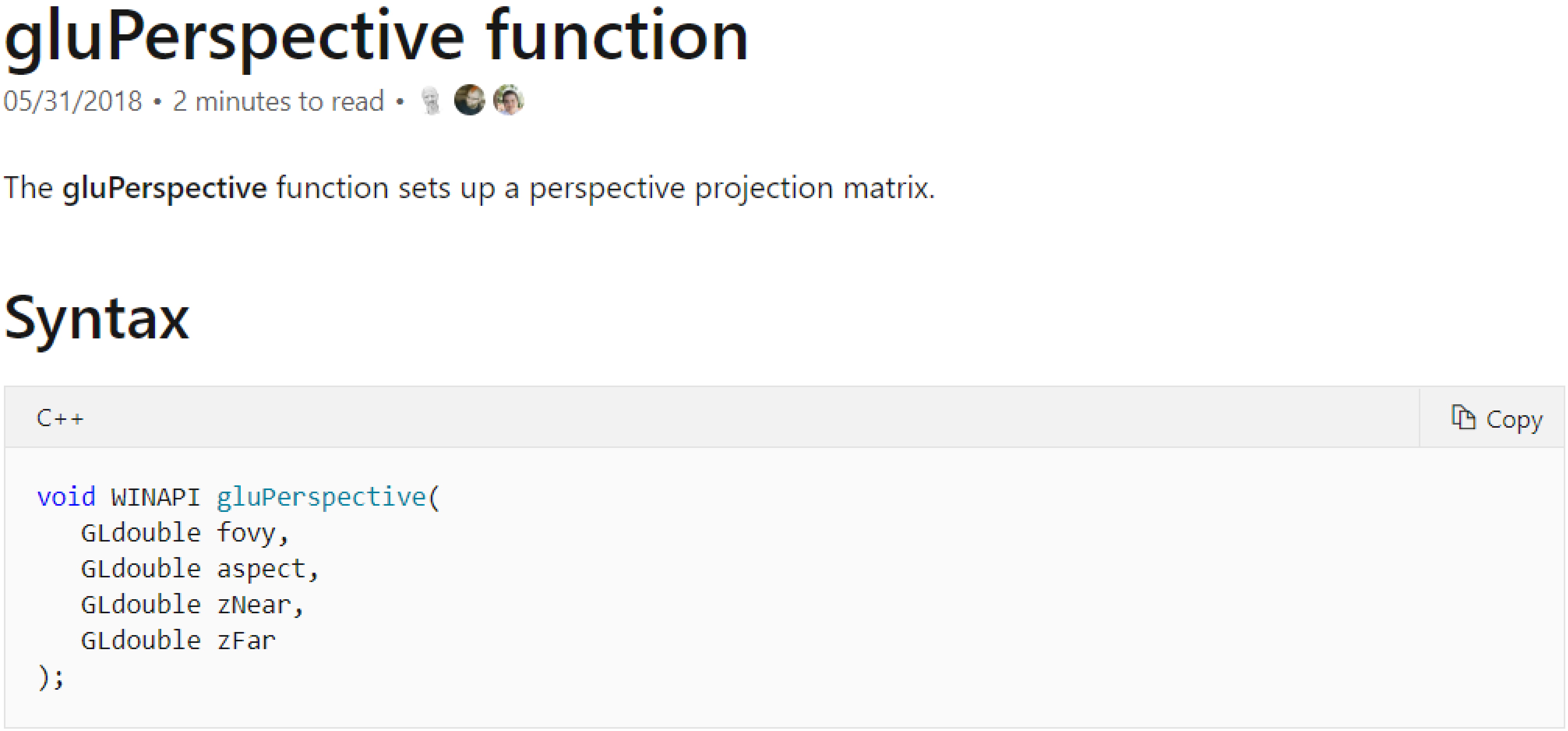
Syntax

C++ Copy

```
void WINAPI glFrustum(
   GLdouble left,
   GLdouble right,
   GLdouble bottom,
   GLdouble top,
   GLdouble zNear,
   GLdouble zFar
);
```

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

Control the position of camera perspective mode



gluPerspective function

05/31/2018 • 2 minutes to read •

The **gluPerspective** function sets up a perspective projection matrix.

Syntax

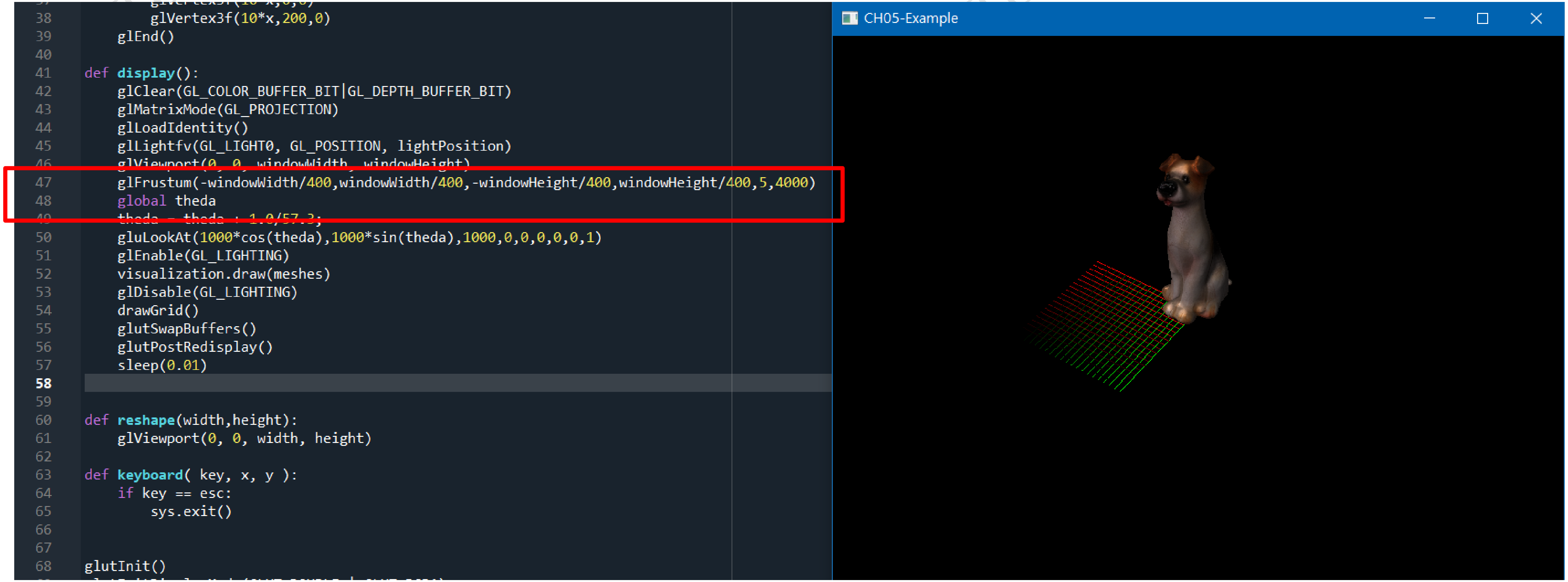
C++ Copy

```
void WINAPI gluPerspective(
   GLdouble fovy,
   GLdouble aspect,
   GLdouble zNear,
   GLdouble zFar
);
```

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

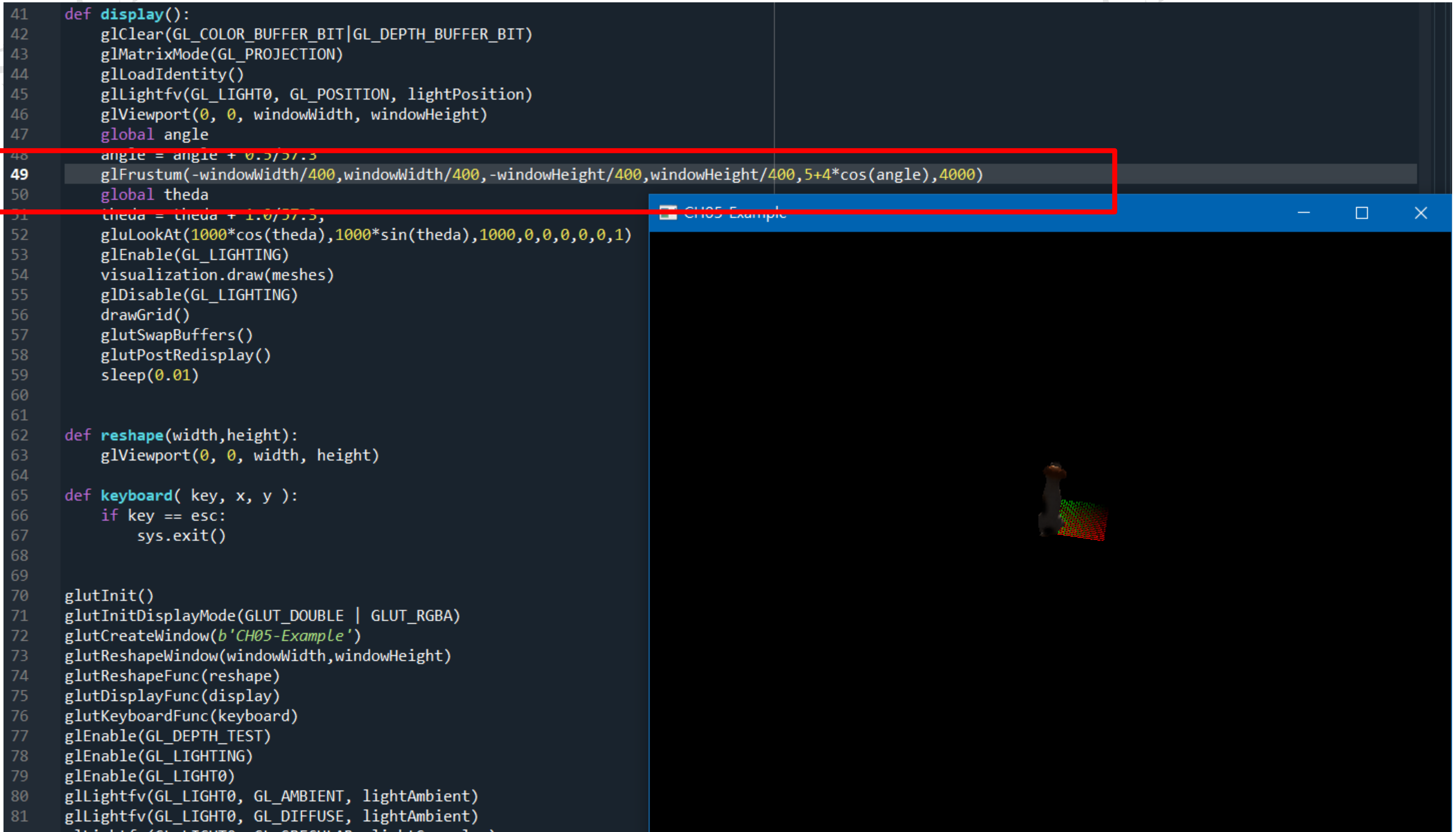
Control the position of camera: perspective mode

- glFrustum (assign parameters carefully)



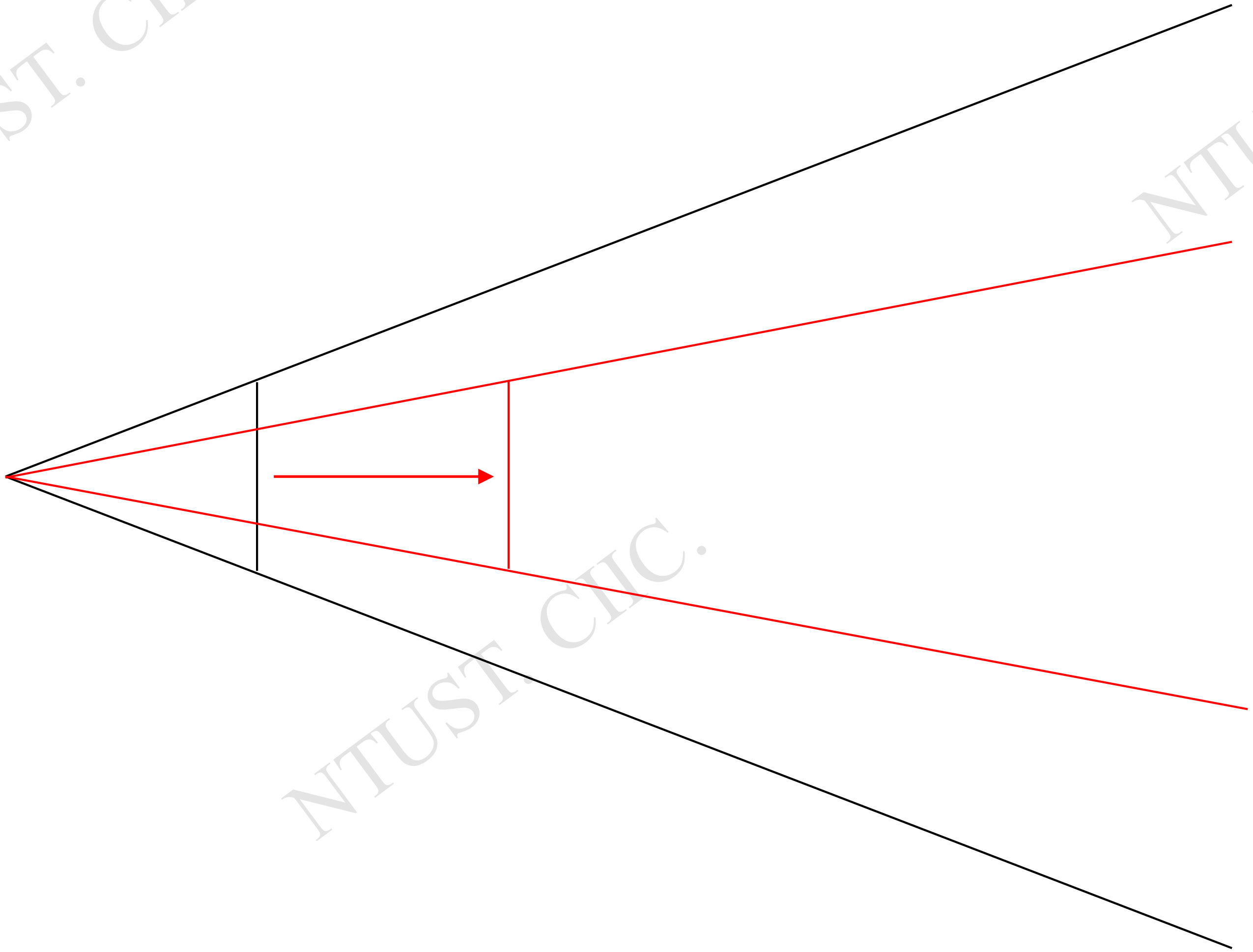
```
        glVertex3f(10*x,0,0)
        glVertex3f(10*x,200,0)
    glEnd()

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)
    glFrustum(-windowWidth/400,windowWidth/400,-windowHeight/400,windowHeight/400,5,4000)
    global theda
    theda = theda + 1.0/57.3;
    gluLookAt(1000*cos(theda),1000*sin(theda),1000,0,0,0,0,0,1)
    glEnable(GL_LIGHTING)
    visualization.draw(meshes)
    glDisable(GL_LIGHTING)
    drawGrid()
    glutSwapBuffers()
    glutPostRedisplay()
    sleep(0.01)



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

def keyboard( key, x, y ):
    if key == esc:
        sys.exit()


glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA)
```

Control the position of camera: zoom by plane distance

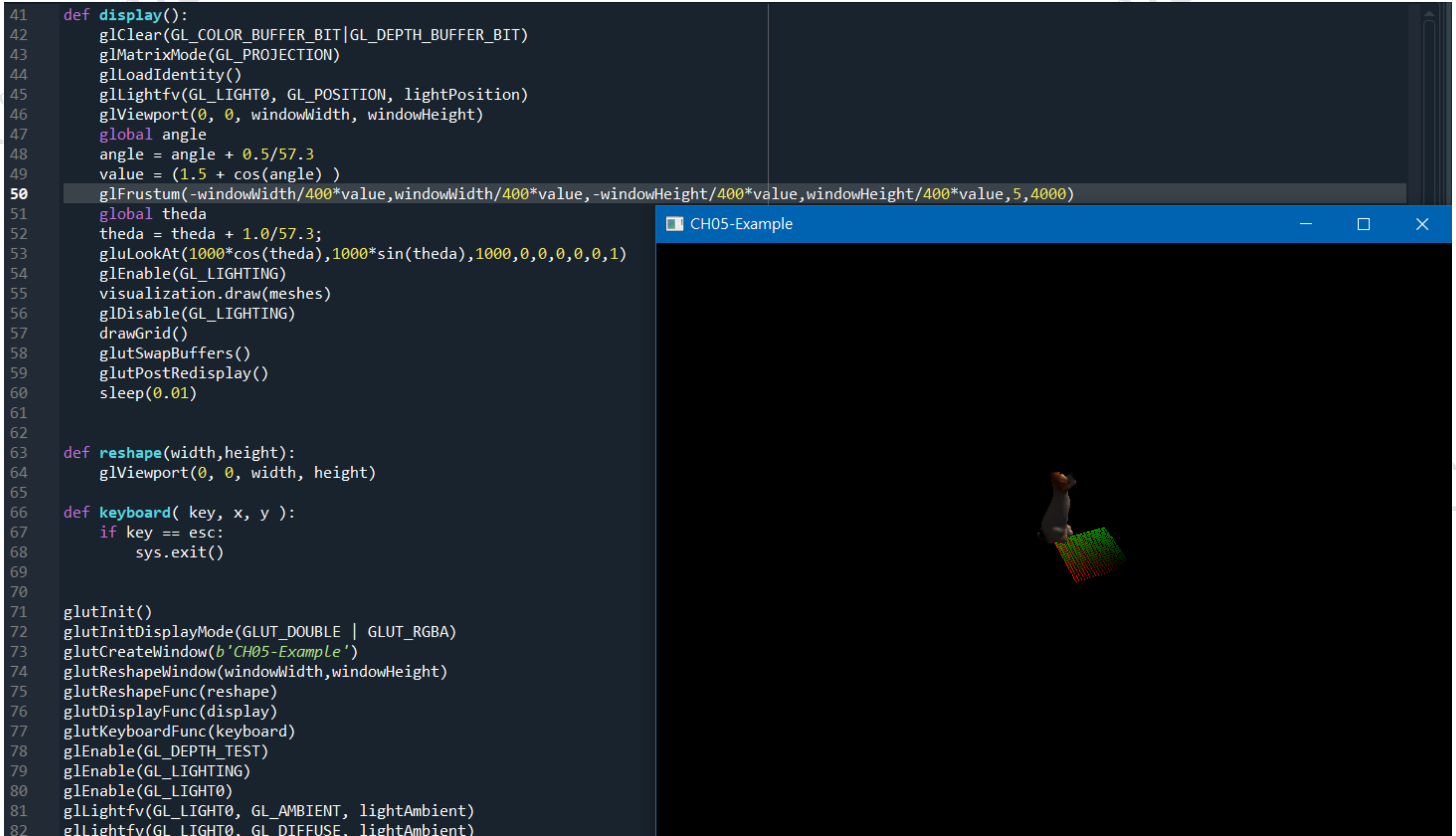


```
def display():
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glLightfv(GL_LIGHT0, GL_POSITION, lightPosition)
    glViewport(0, 0, windowWidth, windowHeight)
    global angle
    angle = angle + 0.5/57.3
    glFrustum(-windowWidth/400,windowWidth/400,-windowHeight/400,windowHeight/400,5+4*cos(angle),4000)
    global theda
    theda = theda + 1.0/57.3;
    gluLookAt(1000*cos(theda),1000*sin(theda),1000,0,0,0,0,0,1)
    glEnable(GL_LIGHTING)
    visualization.draw(meshes)
    glDisable(GL_LIGHTING)
    drawGrid()
    glutSwapBuffers()
    glutPostRedisplay()
    sleep(0.01)


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

def keyboard( key, x, y ):
    if key == esc:
        sys.exit()


glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA)
glutCreateWindow(b'CH05-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)
```

Control the position of camera: zoom by plane distance

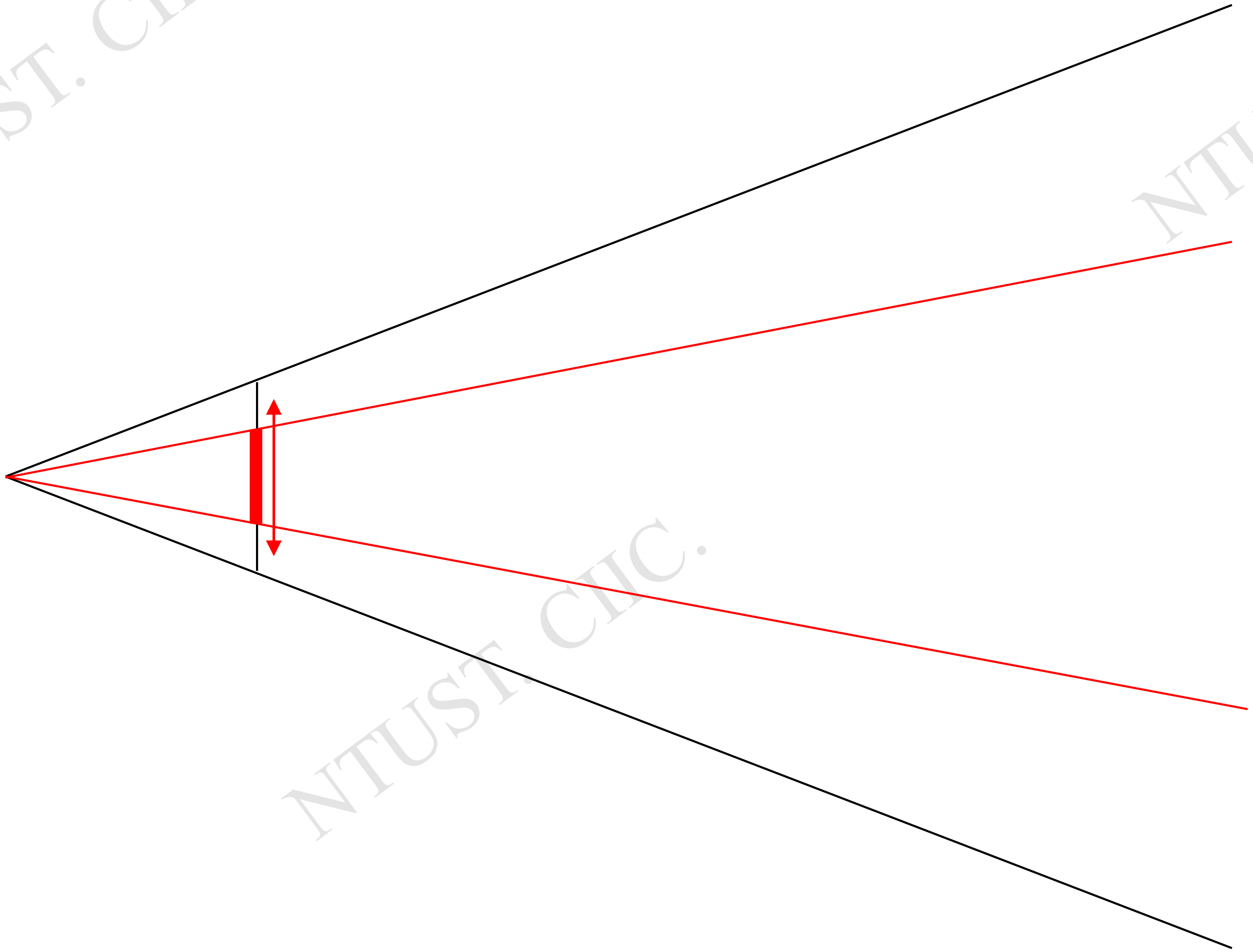


Control the position of camera: zoom by plane size

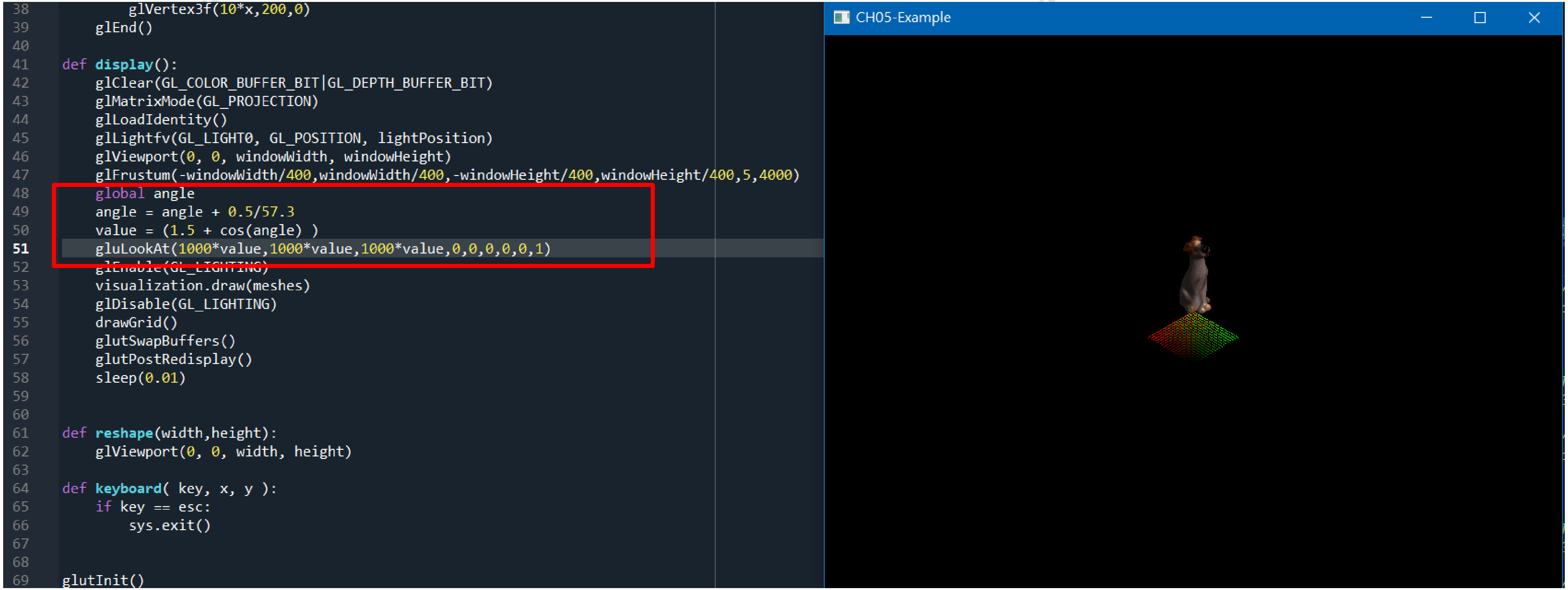


```
def display():
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glLightfv(GL_LIGHT0, GL_POSITION, lightPosition)
    glViewport(0, 0, windowWidth, windowHeight)
    global angle
    angle = angle + 0.5/57.3
    value = (1.5 + cos(angle) )
    glFrustum(-windowWidth/400*value,windowWidth/400*value,-windowHeight/400*value,windowHeight/400*value,5,4000)
    global theda
    theda = theda + 1.0/57.3;
    gluLookAt(1000*cos(theda),1000*sin(theda),1000,0,0,0,0,0,1)
    glEnable(GL_LIGHTING)
    visualization.draw(meshes)
    glDisable(GL_LIGHTING)
    drawGrid()
    glutSwapBuffers()
    glutPostRedisplay()
    sleep(0.01)


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

def keyboard( key, x, y ):
    if key == esc:
        sys.exit()


glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA)
glutCreateWindow(b'CH05-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)
```

Control the position of camera: zoom by plane size



Control the position of camera: zoom by position



```
        glVertex3f(10*x,200,0)
    glEnd()

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)
    glFrustum(-windowWidth/400,windowWidth/400,-windowHeight/400,windowHeight/400,5,4000)
    global angle
    angle = angle + 0.5/57.3
    value = (1.5 + cos(angle) )
    gluLookAt(1000*value,1000*value,1000*value,0,0,0,0,0,1)
    glEnable(GL_LIGHTING)
    visualization.draw(meshes)
    glDisable(GL_LIGHTING)
    drawGrid()
    glutSwapBuffers()
    glutPostRedisplay()
    sleep(0.01)


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

def keyboard( key, x, y ):
    if key == esc:
        sys.exit()


glutInit()
```

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

Find us on: facebook