Advanced Computer Graphics

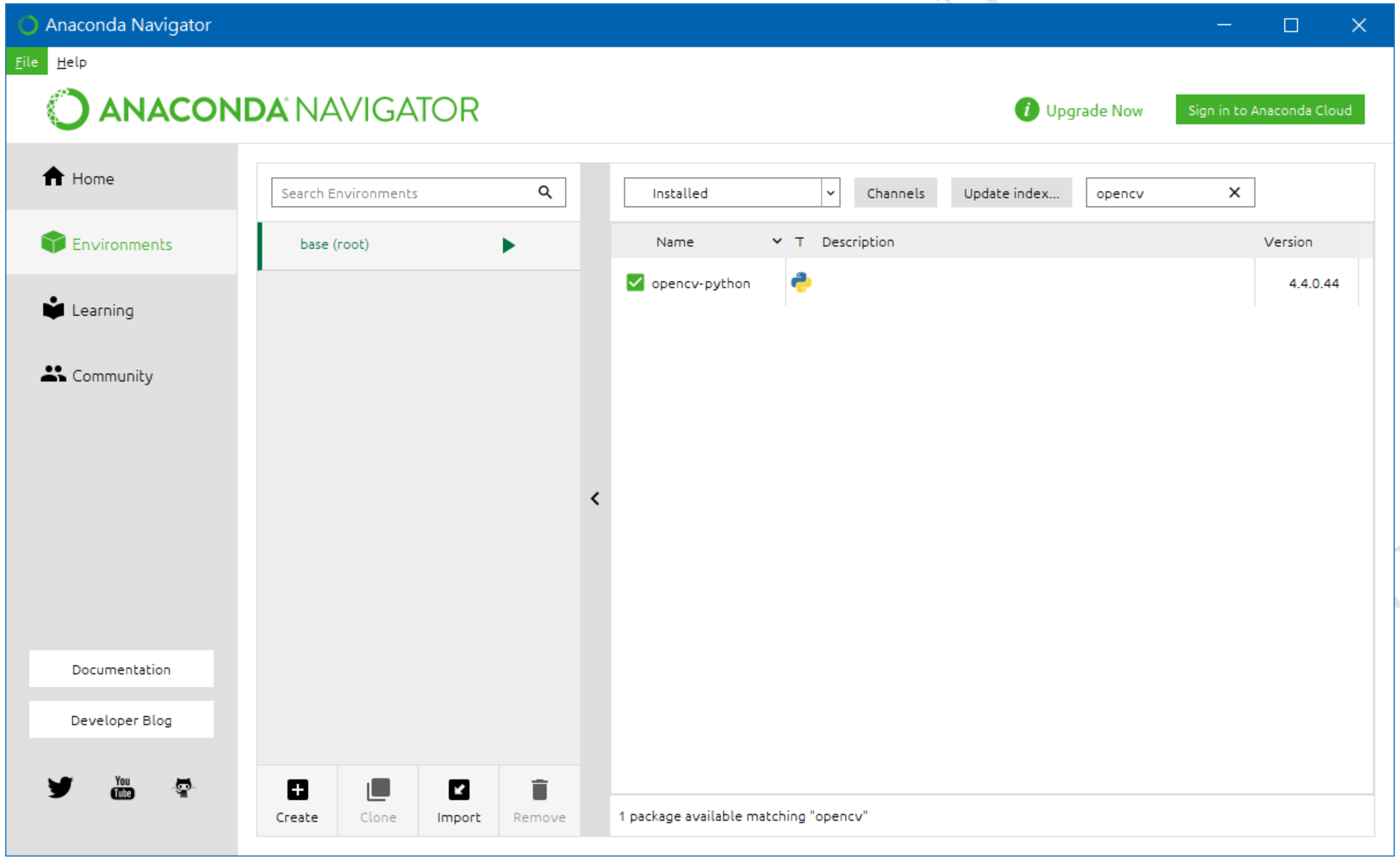
Lecture-08 Introduction to OpenGL-7

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

openGL
Z-Buffering
Color Buffer

Dump Buffer by openCV

- Make sure openCV is available in your program



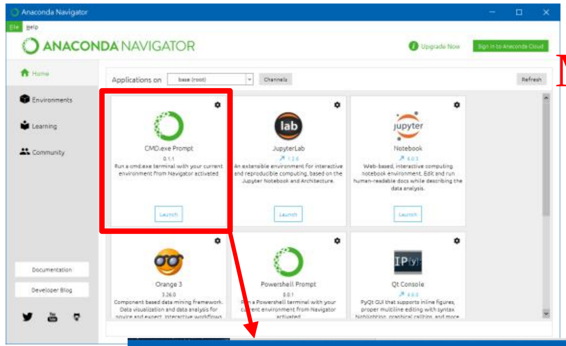
Name	T	Description	Version
✓ opencv-python			4.4.0.44

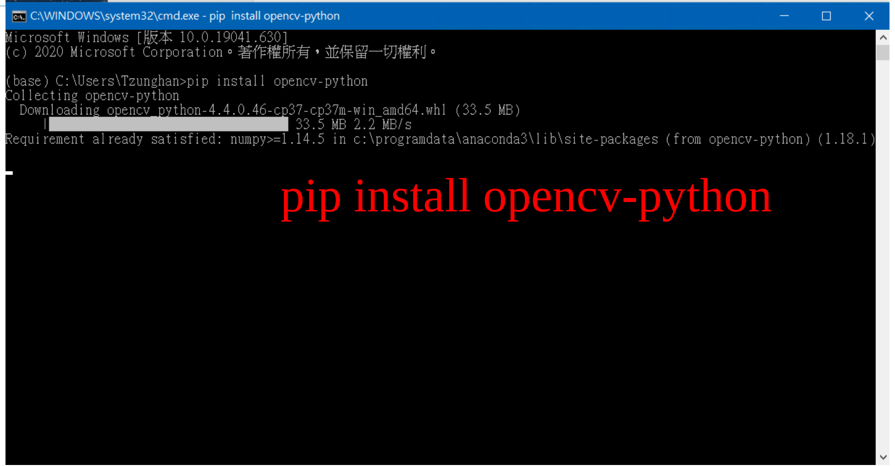
1 package available matching "opencv"

Dump Buffer by openCV

- Two ways to install openCV
  - In command console: type “pip install opencv-python” (for latest version)
  - In anaconda search openCV

Method 1 (latest version)



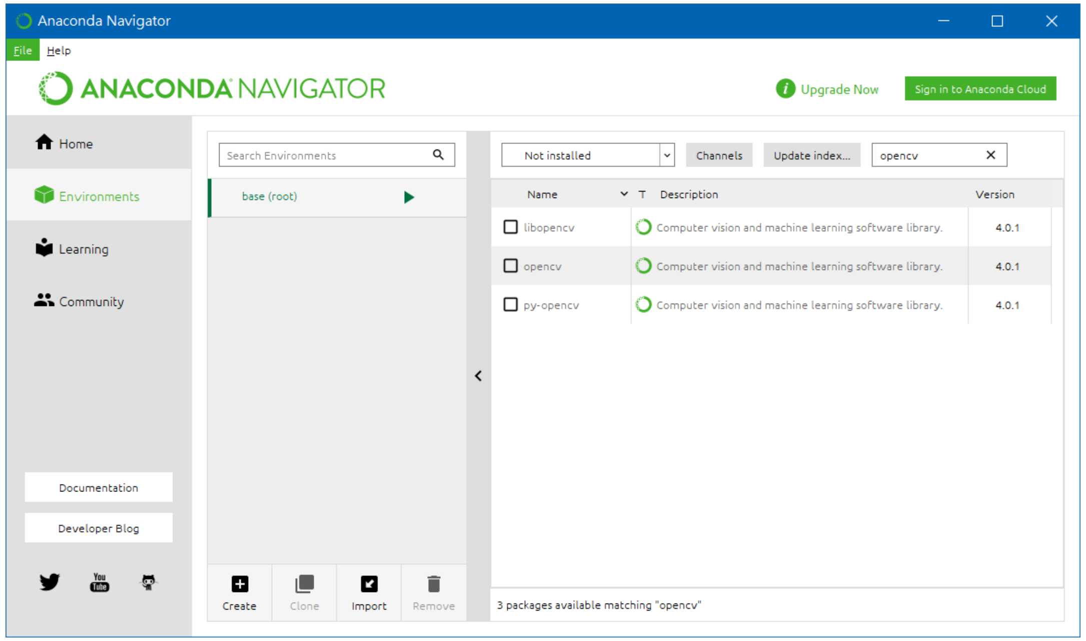


C:\WINDOWS\system32\cmd.exe - pip install opencv-python

```
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>pip install opencv-python
Collecting opencv-python
  Downloading opencv_python-4.4.0.46-cp37-cp37m-win_amd64.whl (33.5 MB)
     |████████████████████████████████| 33.5 MB 2.2 MB/s
Requirement already satisfied: numpy>=1.14.5 in c:\programdata\anaconda3\lib\site-packages (from opencv-python) (1.18.1)
```

pip install opencv-python

Method 2 (older version)



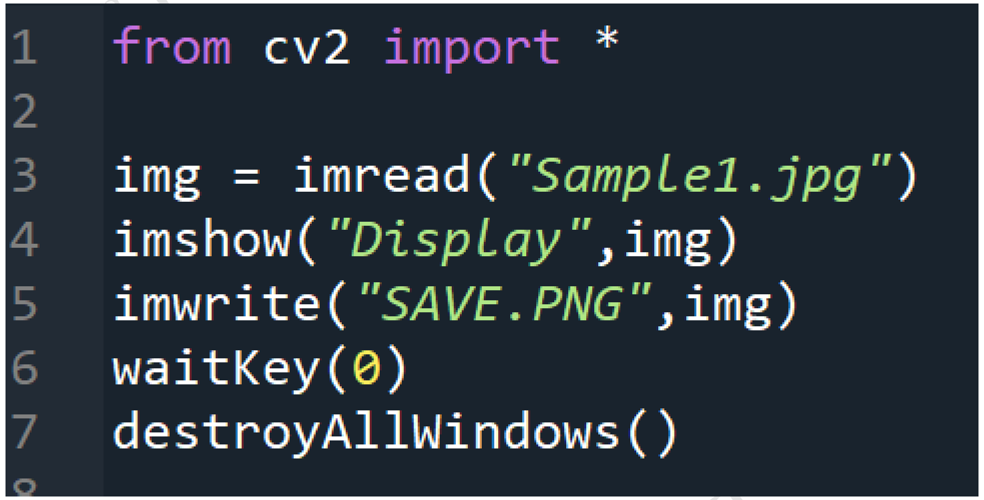
	Name	Description	Version
☐	libopencv	Computer vision and machine learning software library.	4.0.1
☐	opencv	Computer vision and machine learning software library.	4.0.1
☐	py-opencv	Computer vision and machine learning software library.	4.0.1

3 packages available matching "opencv"

Dump Buffer by openCV

- Image in openCV
  - Color: CU_8UC3 (as well as Vec3b) for 24 bit
  - Color in Vec3b represents BGR instead of RGB
  - Image is vertical “Flipped” comparing to Standard-Image

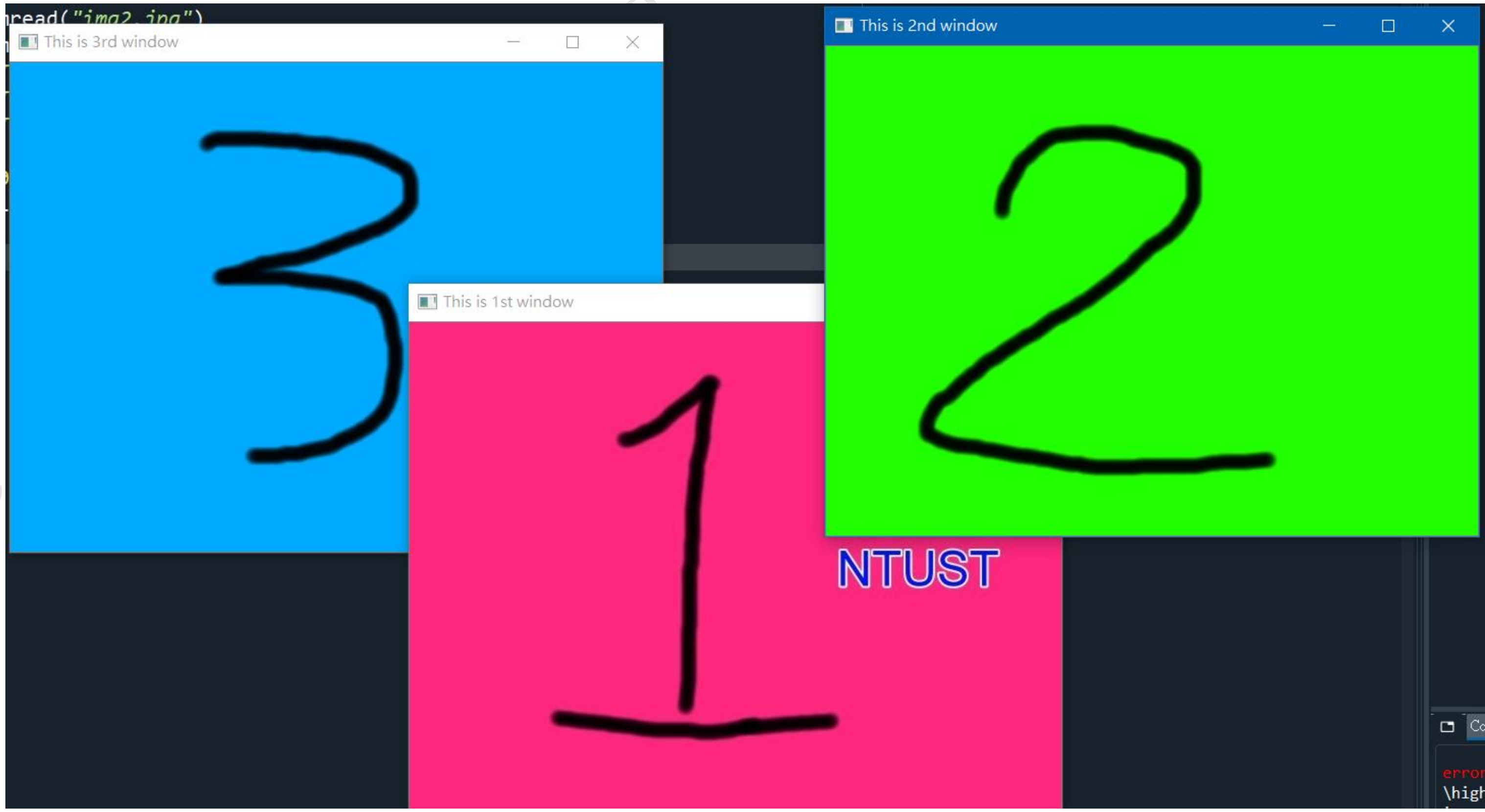
Read images, show images, and save images



```python
from cv2 import *

img = imread("Sample1.jpg")
imshow("Display",img)
imwrite("SAVE.PNG",img)
waitKey(0)
destroyAllWindows()
```

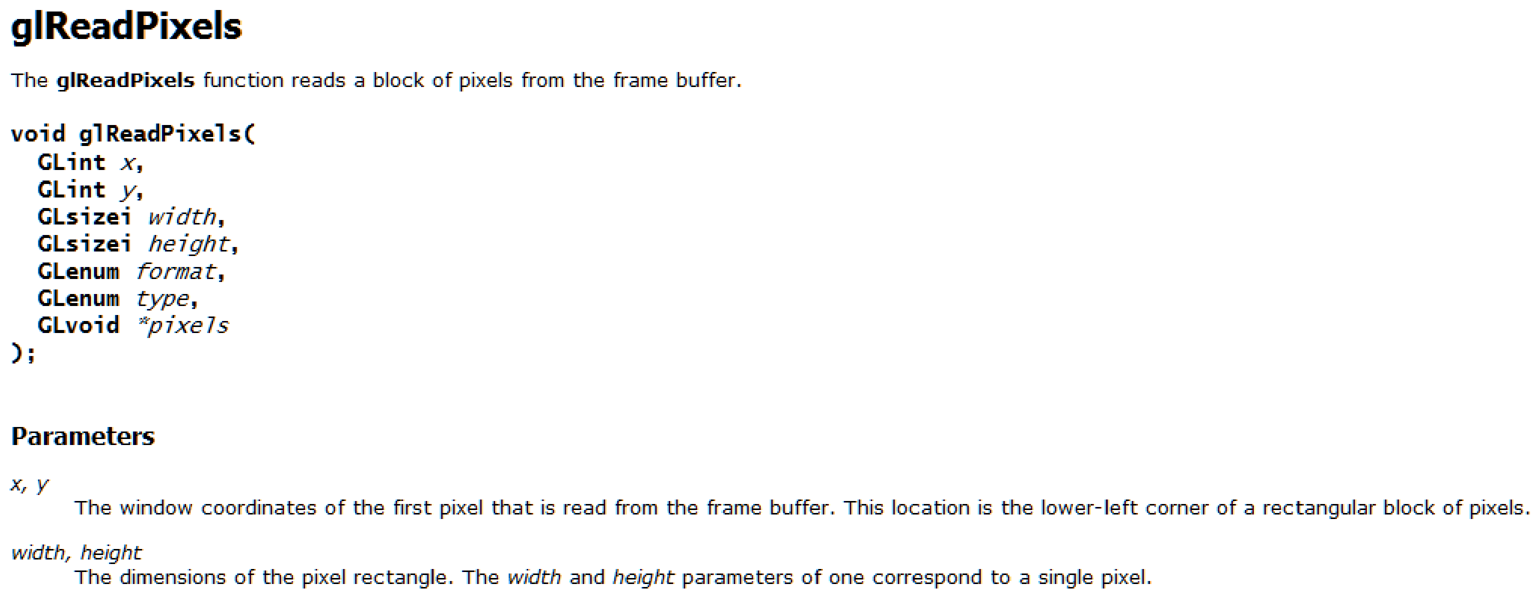
Show many images

```python
from cv2 import *

img1 = imread("img1.jpg")
img2 = imread("img2.jpg")
img3 = imread("img3.jpg")
imshow("This is 1st window",img1)
imshow("This is 2nd window",img2)
imshow("This is 3rd window",img3)

waitKey(0)
destroyAllWindows()
```



Dump Buffer

- "glReadPixels"



glReadPixels

The **glReadPixels** function reads a block of pixels from the frame buffer.

```
void glReadPixels(
  GLint x,
  GLint y,
  GLsizei width,
  GLsizei height,
  GLenum format,
  GLenum type,
  GLvoid *pixels
);
```

Parameters

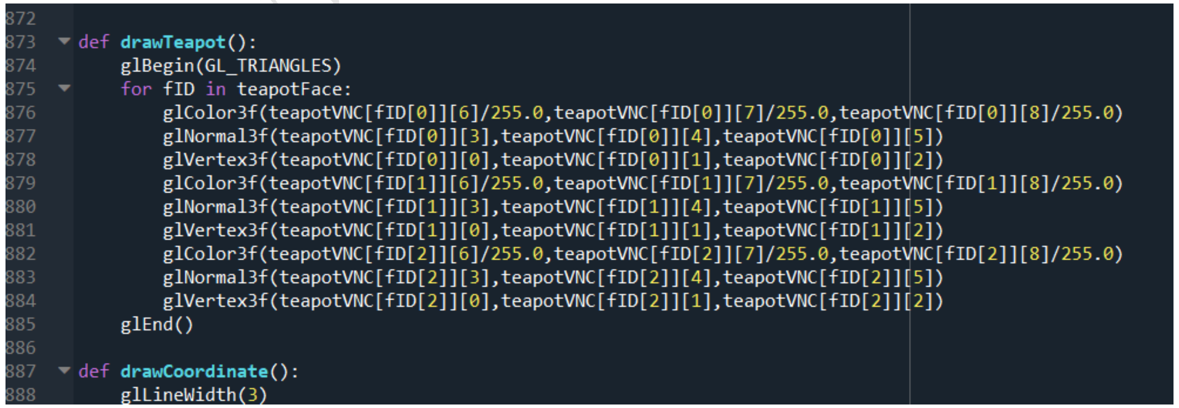
x, y
The window coordinates of the first pixel that is read from the frame buffer. This location is the lower-left corner of a rectangular block of pixels.

width, height
The dimensions of the pixel rectangle. The *width* and *height* parameters of one correspond to a single pixel.

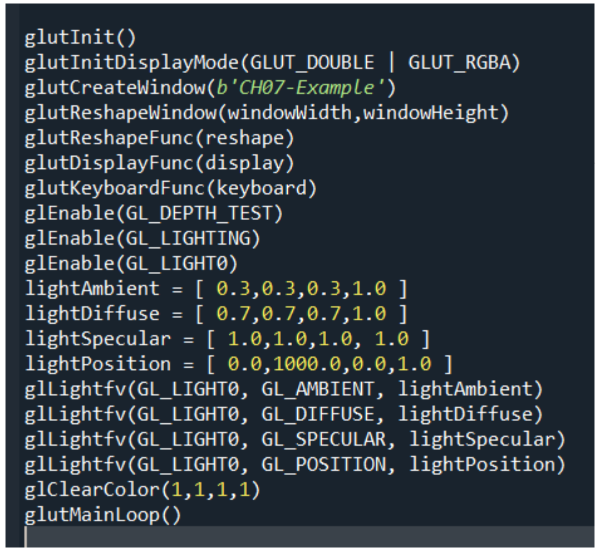
- GL_BGR_EXT → 24bit color image
- GL_BGRA_EXT→ 32bit color image
- GL_DEPTH_COMPONENT→ depth buffer

OpenGL Draw Color (Shade)

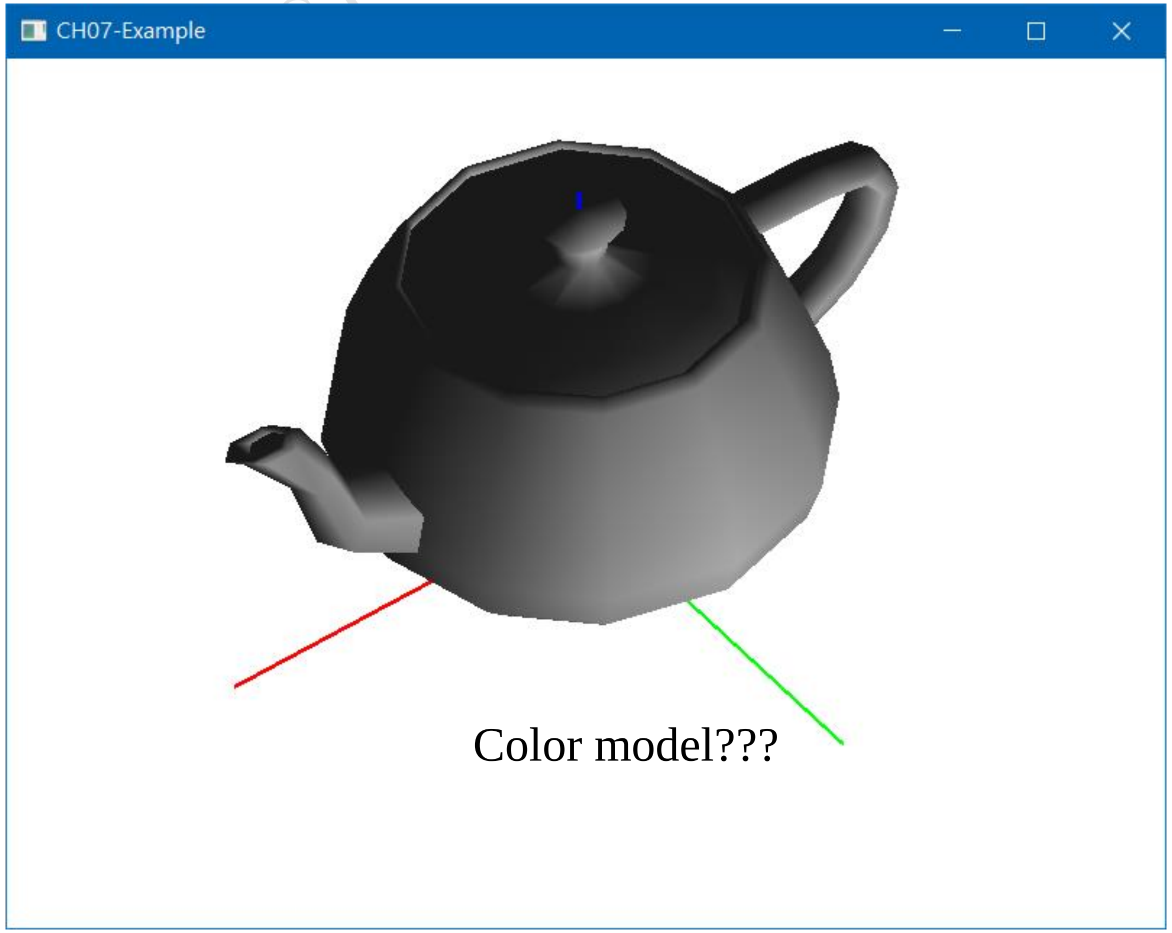
- Disable GL_COLOR_MATERIAL



```
def drawTeapot():
    glBegin(GL_TRIANGLES)
    for fID in teapotFace:
        glColor3f(teapotVNC[fID[0]][6]/255.0,teapotVNC[fID[0]][7]/255.0,teapotVNC[fID[0]][8]/255.0)
        glNormal3f(teapotVNC[fID[0]][3],teapotVNC[fID[0]][4],teapotVNC[fID[0]][5])
        glVertex3f(teapotVNC[fID[0]][0],teapotVNC[fID[0]][1],teapotVNC[fID[0]][2])
        glColor3f(teapotVNC[fID[1]][6]/255.0,teapotVNC[fID[1]][7]/255.0,teapotVNC[fID[1]][8]/255.0)
        glNormal3f(teapotVNC[fID[1]][3],teapotVNC[fID[1]][4],teapotVNC[fID[1]][5])
        glVertex3f(teapotVNC[fID[1]][0],teapotVNC[fID[1]][1],teapotVNC[fID[1]][2])
        glColor3f(teapotVNC[fID[2]][6]/255.0,teapotVNC[fID[2]][7]/255.0,teapotVNC[fID[2]][8]/255.0)
        glNormal3f(teapotVNC[fID[2]][3],teapotVNC[fID[2]][4],teapotVNC[fID[2]][5])
        glVertex3f(teapotVNC[fID[2]][0],teapotVNC[fID[2]][1],teapotVNC[fID[2]][2])
    glEnd()

def drawCoordinate():
    glLineWidth(3)
```



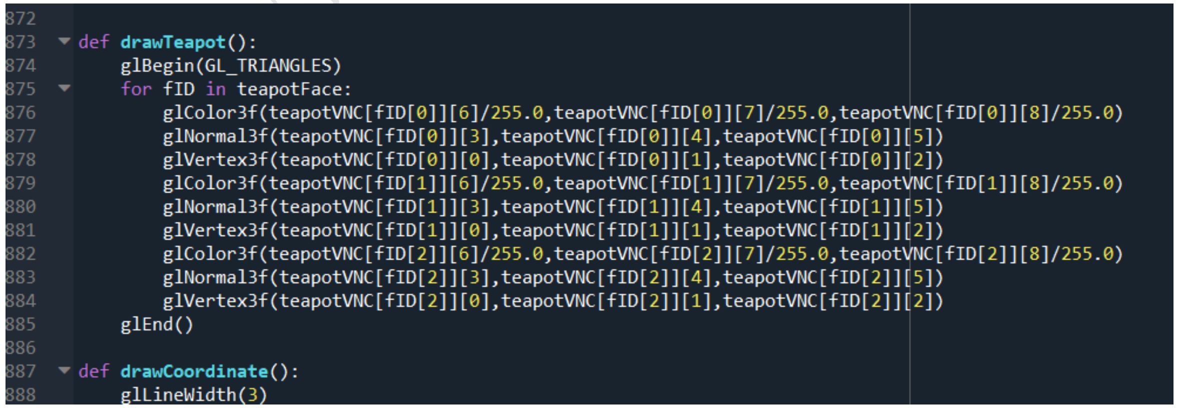
```
glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA)
glutCreateWindow(b'CH07-Example')
glutReshapeWindow(windowWidth,windowHeight)
glutReshapeFunc(reshape)
glutDisplayFunc(display)
glutKeyboardFunc(keyboard)
glEnable(GL_DEPTH_TEST)
glEnable(GL_LIGHTING)
glEnable(GL_LIGHT0)
lightAmbient = [ 0.3,0.3,0.3,1.0 ]
lightDiffuse = [ 0.7,0.7,0.7,1.0 ]
lightSpecular = [ 1.0,1.0,1.0, 1.0 ]
lightPosition = [ 0.0,1000.0,0.0,1.0 ]
glLightfv(GL_LIGHT0, GL_AMBIENT, lightAmbient)
glLightfv(GL_LIGHT0, GL_DIFFUSE, lightDiffuse)
glLightfv(GL_LIGHT0, GL_SPECULAR, lightSpecular)
glLightfv(GL_LIGHT0, GL_POSITION, lightPosition)
glClearColor(1,1,1,1)
glutMainLoop()
```



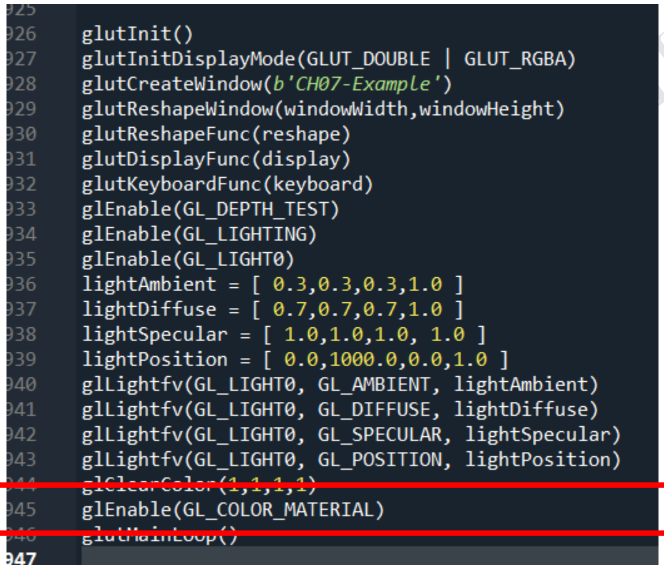
Color model???

OpenGL Draw Color (Shade)

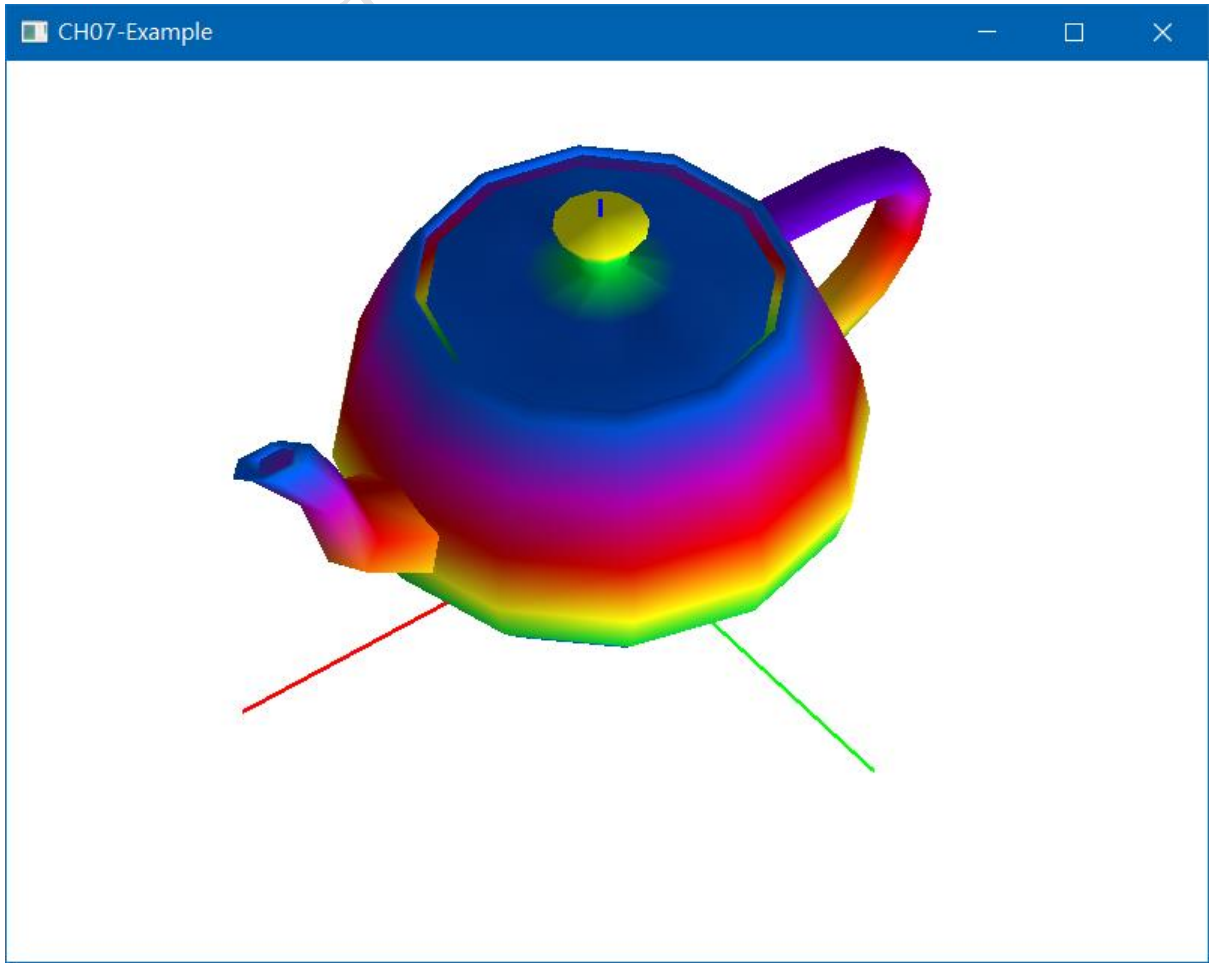
- Enable GL_COLOR_MATERIAL



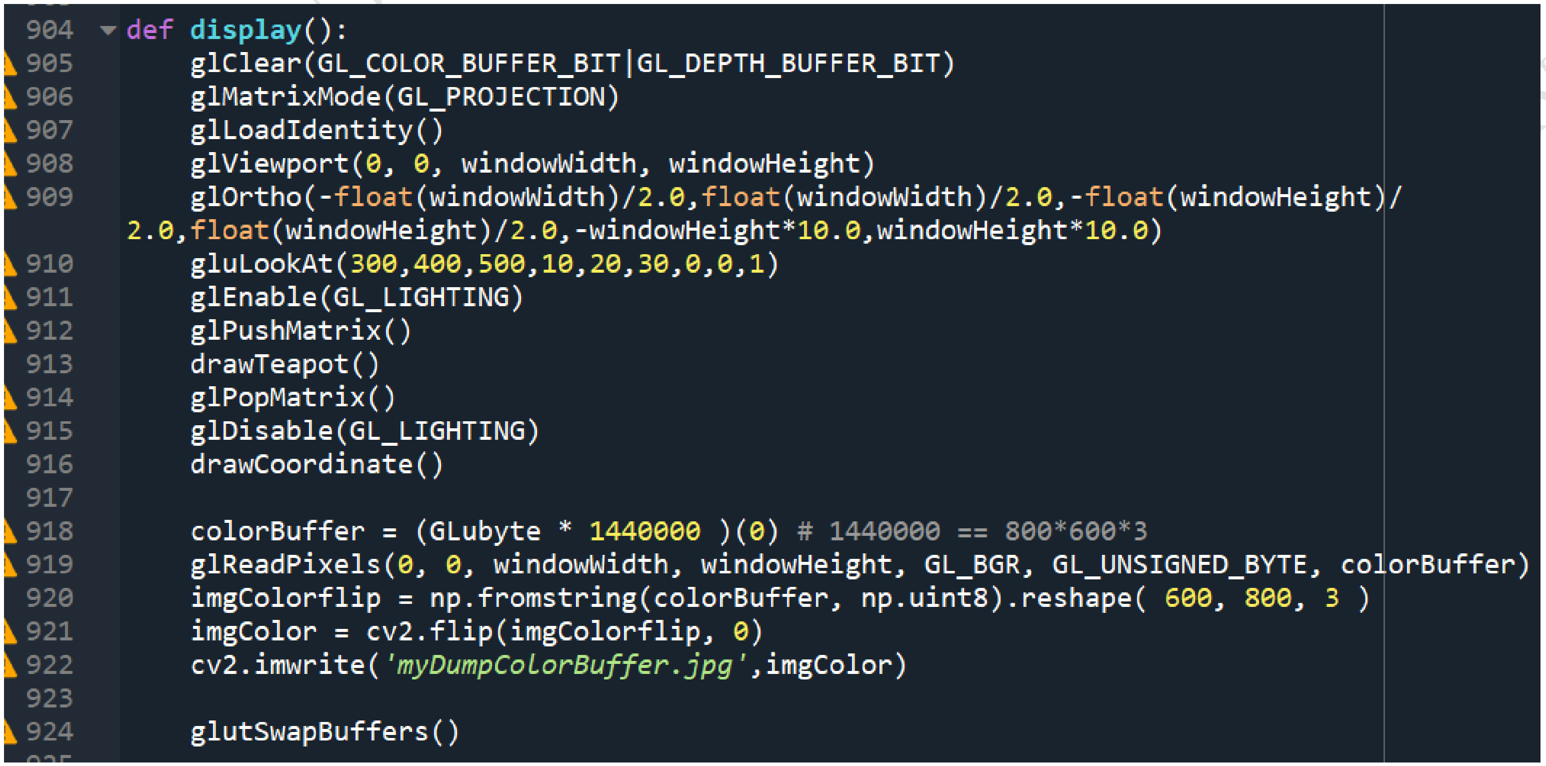
```
def drawTeapot():
    glBegin(GL_TRIANGLES)
    for fID in teapotFace:
        glColor3f(teapotVNC[fID[0]][6]/255.0,teapotVNC[fID[0]][7]/255.0,teapotVNC[fID[0]][8]/255.0)
        glNormal3f(teapotVNC[fID[0]][3],teapotVNC[fID[0]][4],teapotVNC[fID[0]][5])
        glVertex3f(teapotVNC[fID[0]][0],teapotVNC[fID[0]][1],teapotVNC[fID[0]][2])
        glColor3f(teapotVNC[fID[1]][6]/255.0,teapotVNC[fID[1]][7]/255.0,teapotVNC[fID[1]][8]/255.0)
        glNormal3f(teapotVNC[fID[1]][3],teapotVNC[fID[1]][4],teapotVNC[fID[1]][5])
        glVertex3f(teapotVNC[fID[1]][0],teapotVNC[fID[1]][1],teapotVNC[fID[1]][2])
        glColor3f(teapotVNC[fID[2]][6]/255.0,teapotVNC[fID[2]][7]/255.0,teapotVNC[fID[2]][8]/255.0)
        glNormal3f(teapotVNC[fID[2]][3],teapotVNC[fID[2]][4],teapotVNC[fID[2]][5])
        glVertex3f(teapotVNC[fID[2]][0],teapotVNC[fID[2]][1],teapotVNC[fID[2]][2])
    glEnd()

def drawCoordinate():
    glLineWidth(3)
```



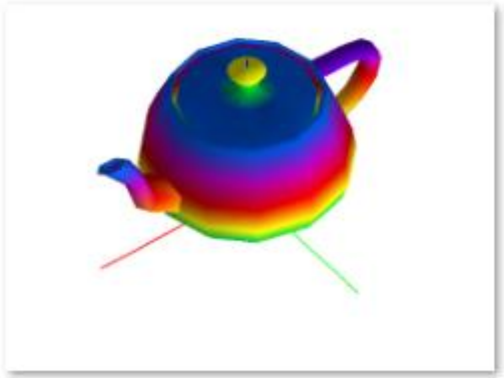
```
glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA)
glutCreateWindow(b'CH07-Example')
glutReshapeWindow(windowWidth,windowHeight)
glutReshapeFunc(reshape)
glutDisplayFunc(display)
glutKeyboardFunc(keyboard)
glEnable(GL_DEPTH_TEST)
glEnable(GL_LIGHTING)
glEnable(GL_LIGHT0)
lightAmbient = [ 0.3,0.3,0.3,1.0 ]
lightDiffuse = [ 0.7,0.7,0.7,1.0 ]
lightSpecular = [ 1.0,1.0,1.0, 1.0 ]
lightPosition = [ 0.0,1000.0,0.0,1.0 ]
glLightfv(GL_LIGHT0, GL_AMBIENT, lightAmbient)
glLightfv(GL_LIGHT0, GL_DIFFUSE, lightDiffuse)
glLightfv(GL_LIGHT0, GL_SPECULAR, lightSpecular)
glLightfv(GL_LIGHT0, GL_POSITION, lightPosition)
glClearColor(1,1,1,1)
glEnable(GL_COLOR_MATERIAL)
glutMainLoop()
```



Dump Color Buffer



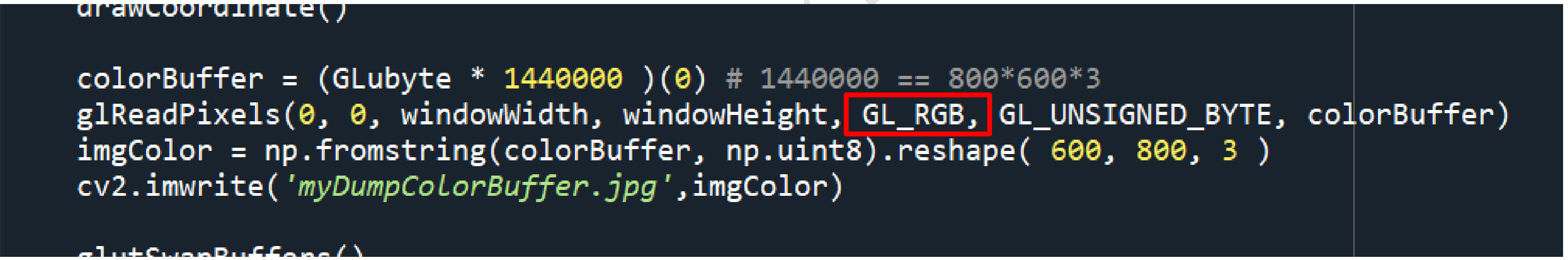
```
def display():
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glViewport(0, 0, windowWidth, windowHeight)
    glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.0,-float(windowHeight)/
2.0,float(windowHeight)/2.0,-windowHeight*10.0,windowHeight*10.0)
    gluLookAt(300,400,500,10,20,30,0,0,1)
    glEnable(GL_LIGHTING)
    glPushMatrix()
    drawTeapot()
    glPopMatrix()
    glDisable(GL_LIGHTING)
    drawCoordinate()

    colorBuffer = (GLubyte * 1440000 )(0) # 1440000 == 800*600*3
    glReadPixels(0, 0, windowWidth, windowHeight, GL_BGR, GL_UNSIGNED_BYTE, colorBuffer)
    imgColorflip = np.fromstring(colorBuffer, np.uint8).reshape( 600, 800, 3 )
    imgColor = cv2.flip(imgColorflip, 0)
    cv2.imwrite('myDumpColorBuffer.jpg',imgColor)

    glutSwapBuffers()
```


myDumpColorBuffer.jpg

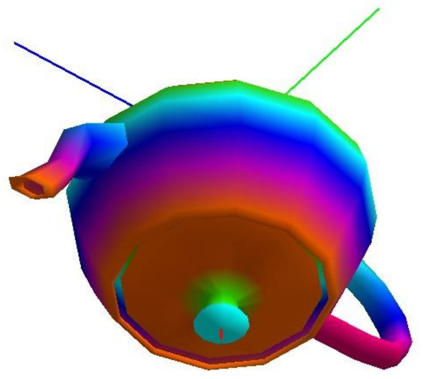
Dump Color Buffer

- Three key points you should know:
  1. color in openCV is stored as B-G-R (not RGB)
  2. openCV has "flip" (upside down) images
  3. You need to allocated memory space (unsigned int format) for storing buffer from glReadPixels

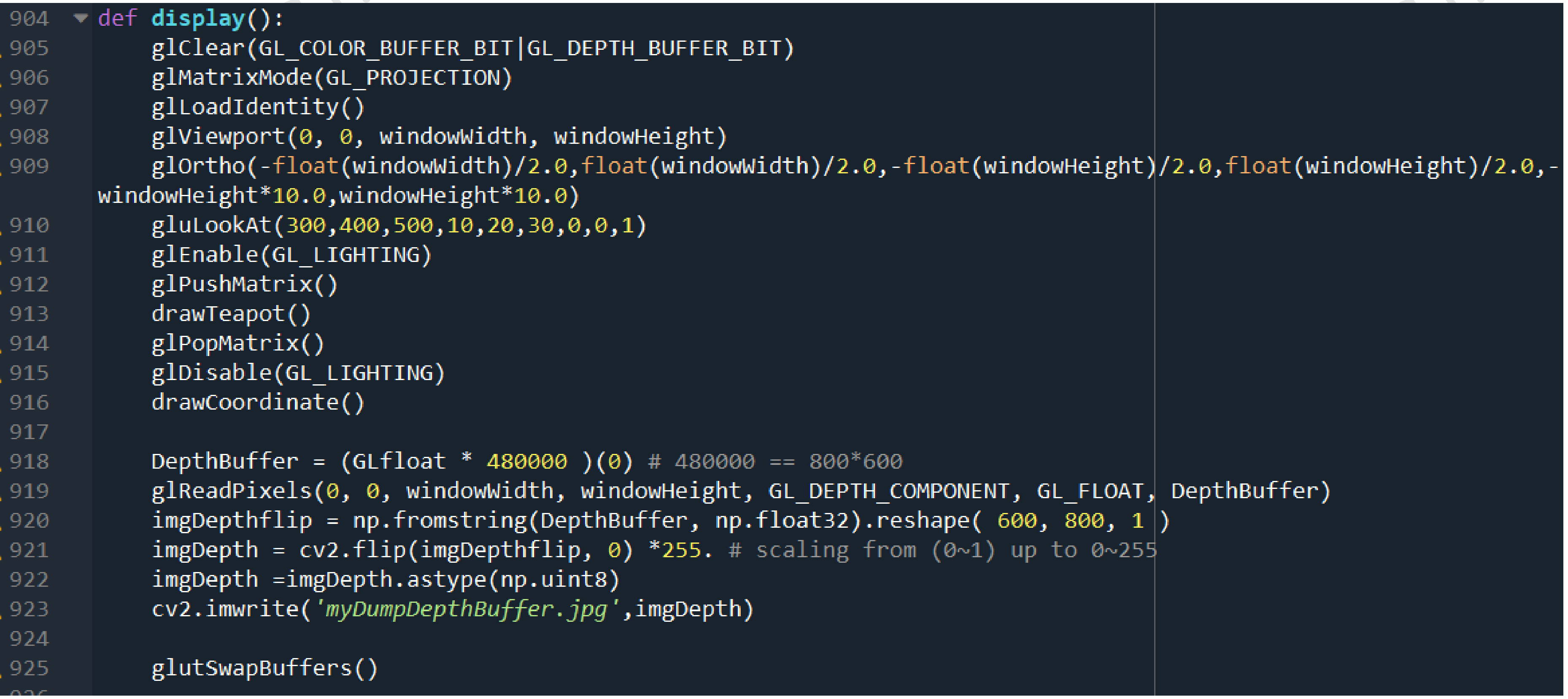
common mistakes



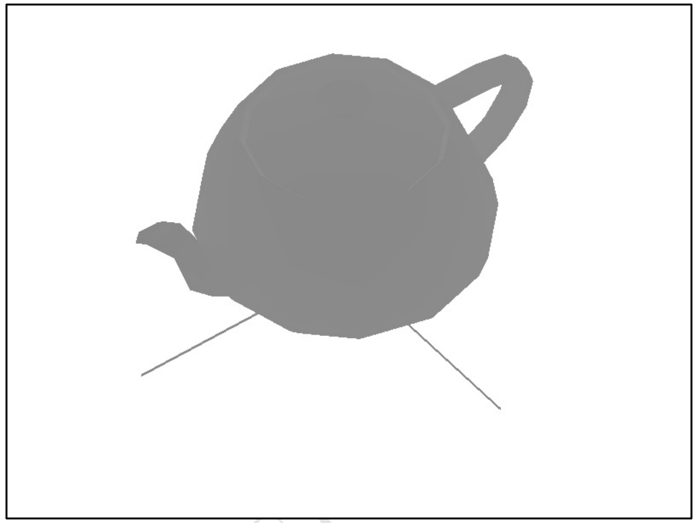
```
    drawCoordinate()

    colorBuffer = (GLubyte * 1440000 )(0) # 1440000 == 800*600*3
    glReadPixels(0, 0, windowWidth, windowHeight, GL_RGB, GL_UNSIGNED_BYTE, colorBuffer)
    imgColor = np.fromstring(colorBuffer, np.uint8).reshape( 600, 800, 3 )
    cv2.imwrite('myDumpColorBuffer.jpg',imgColor)

    glutSwapBuffers()
```



Dump Depth Buffer



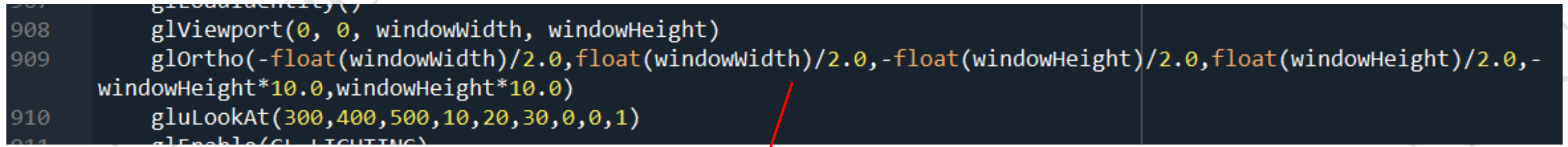
```
def display():
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glViewport(0, 0, windowWidth, windowHeight)
    glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.0,-float(windowHeight)/2.0,float(windowHeight)/2.0,-windowHeight*10.0,windowHeight*10.0)
    gluLookAt(300,400,500,10,20,30,0,0,1)
    glEnable(GL_LIGHTING)
    glPushMatrix()
    drawTeapot()
    glPopMatrix()
    glDisable(GL_LIGHTING)
    drawCoordinate()

    DepthBuffer = (GLfloat * 480000 )(0) # 480000 == 800*600
    glReadPixels(0, 0, windowWidth, windowHeight, GL_DEPTH_COMPONENT, GL_FLOAT, DepthBuffer)
    imgDepthflip = np.fromstring(DepthBuffer, np.float32).reshape( 600, 800, 1 )
    imgDepth = cv2.flip(imgDepthflip, 0) *255. # scaling from (0~1) up to 0~255
    imgDepth =imgDepth.astype(np.uint8)
    cv2.imwrite('myDumpDepthBuffer.jpg',imgDepth)

    glutSwapBuffers()
```



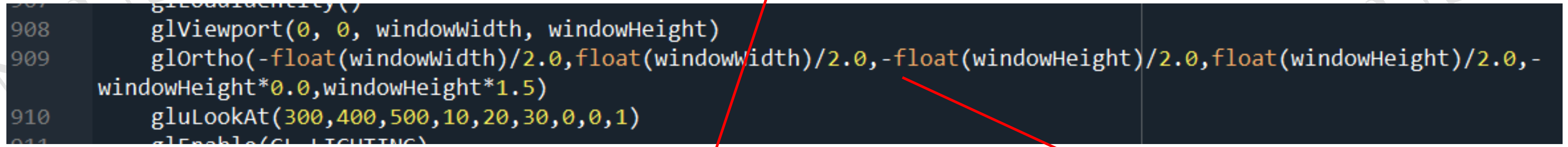
Dump Depth Buffer

- Note:
  - Depth value will be floating point (float32) and data range from 0.0~1.0 by default.
  - To show “Depth” as an image, we need to convert it into 8bit (uint8) by apply a value of 255.0
- Misc.
  - Depth resolution is regarding to “viewing volume”

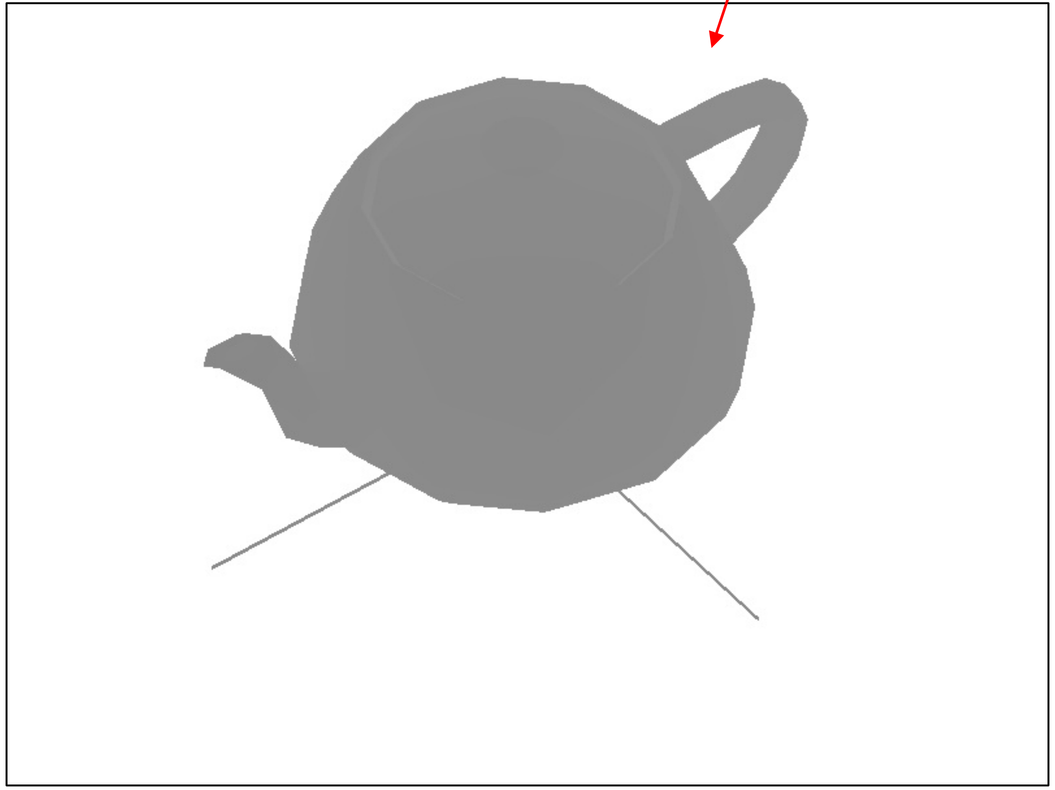
Dump Depth Buffer (adjust depth resolution)

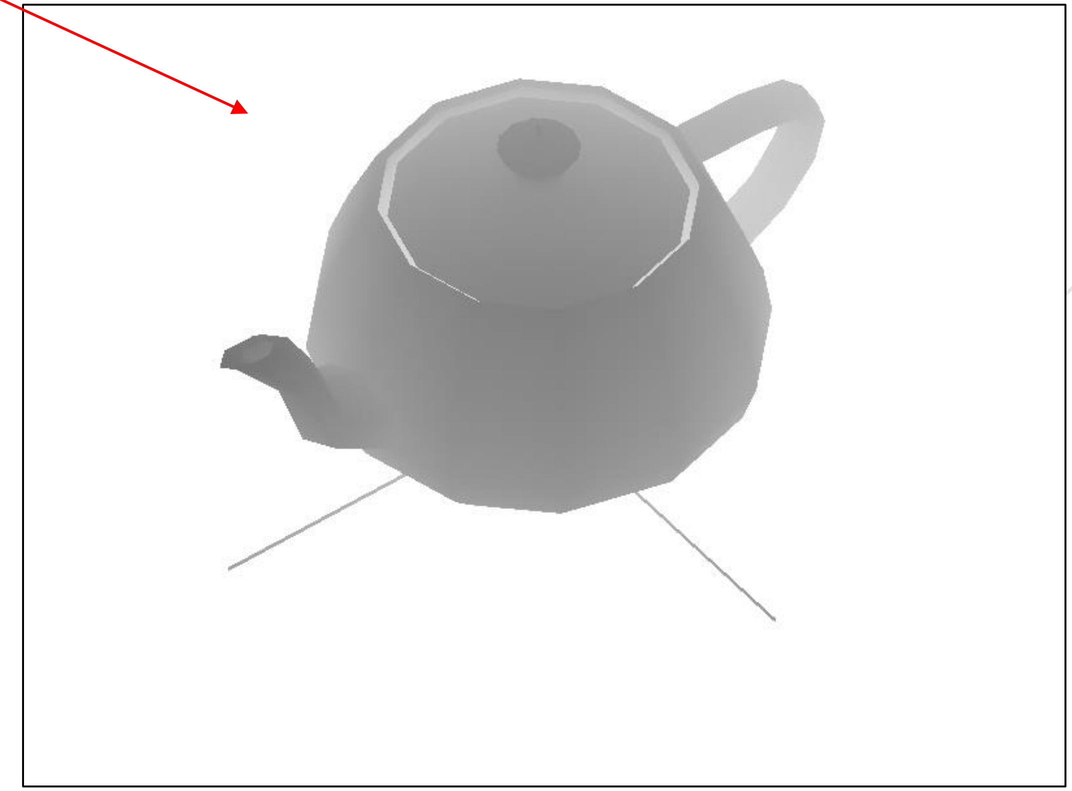


```
    glViewport(0, 0, windowWidth, windowHeight)
    glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.0,-float(windowHeight)/2.0,float(windowHeight)/2.0,-
windowHeight*10.0,windowHeight*10.0)
    gluLookAt(300,400,500,10,20,30,0,0,1)
```

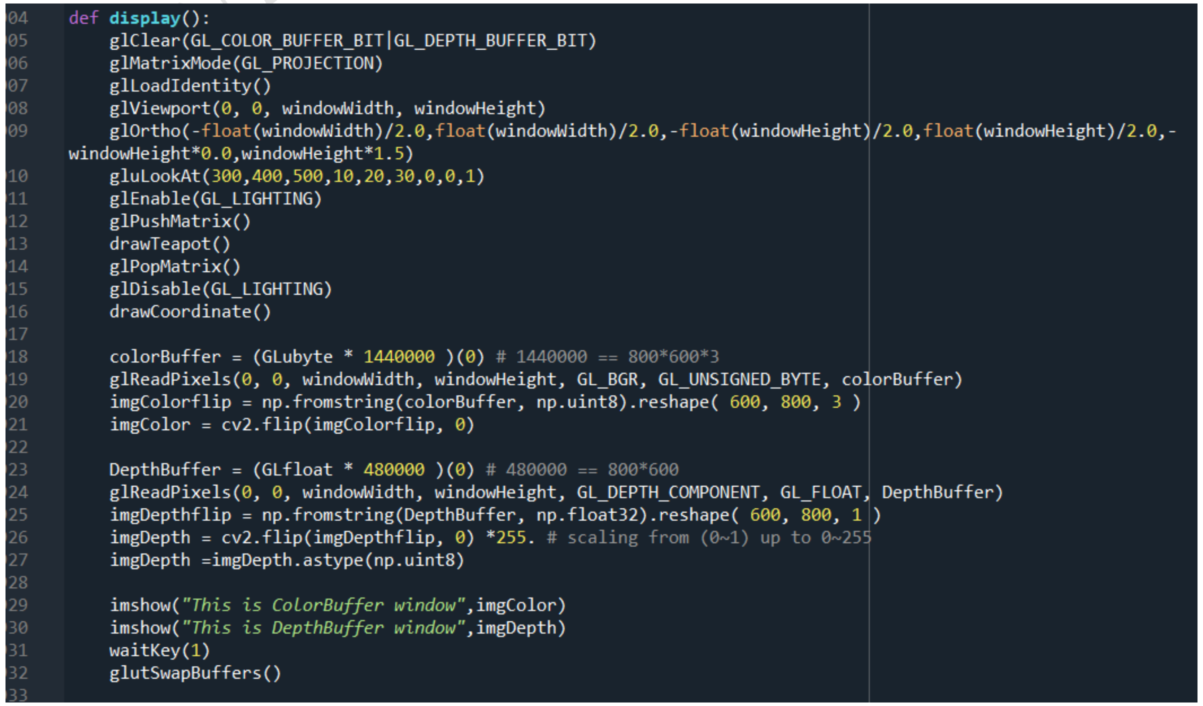


```
    glViewport(0, 0, windowWidth, windowHeight)
    glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.0,-float(windowHeight)/2.0,float(windowHeight)/2.0,-
windowHeight*0.0,windowHeight*1.5)
    gluLookAt(300,400,500,10,20,30,0,0,1)
```

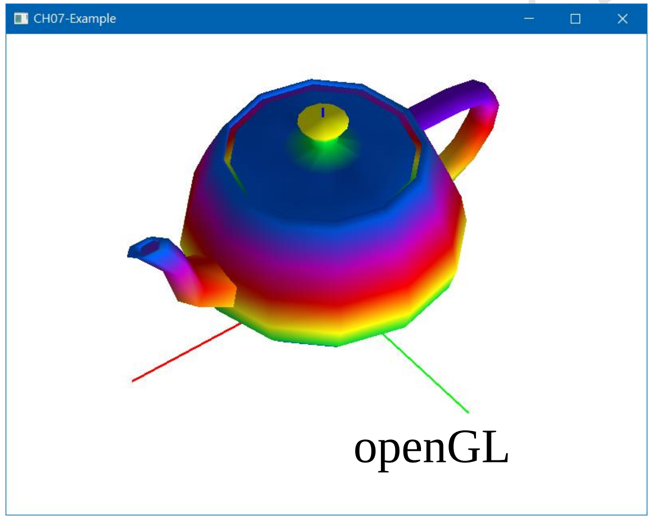




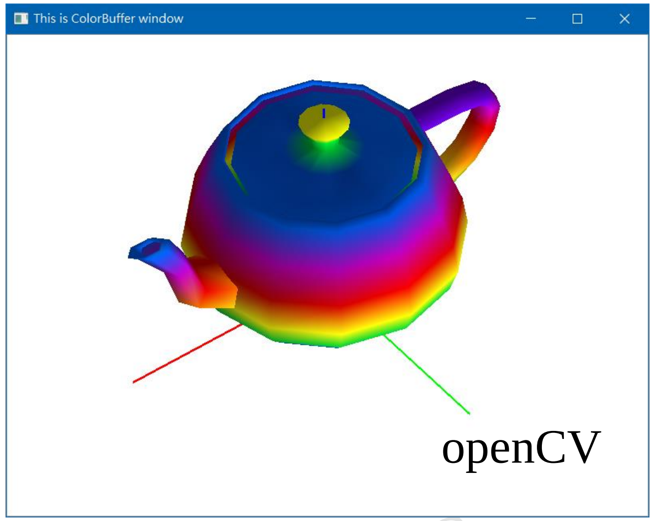
Show images when display openGL



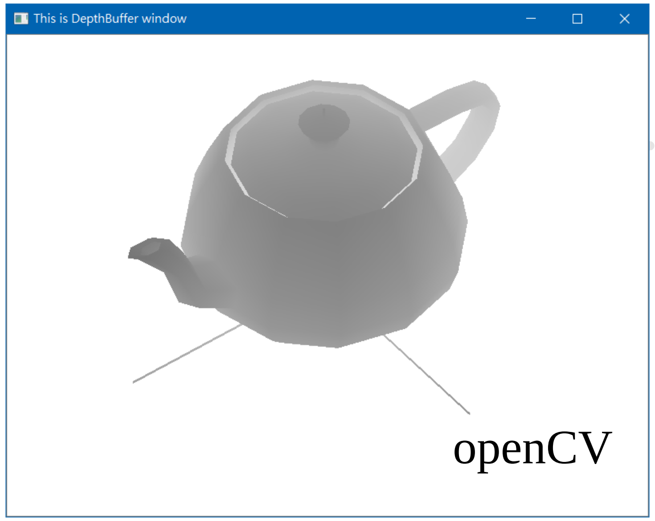
```python
def display():
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glViewport(0, 0, windowWidth, windowHeight)
    glOrtho(-float(windowWidth)/2.0,float(windowWidth)/2.0,-float(windowHeight)/2.0,float(windowHeight)/2.0,-
windowHeight*0.0,windowHeight*1.5)
    gluLookAt(300,400,500,10,20,30,0,0,1)
    glEnable(GL_LIGHTING)
    glPushMatrix()
    drawTeapot()
    glPopMatrix()
    glDisable(GL_LIGHTING)
    drawCoordinate()

    colorBuffer = (GLubyte * 1440000 )(0) # 1440000 == 800*600*3
    glReadPixels(0, 0, windowWidth, windowHeight, GL_BGR, GL_UNSIGNED_BYTE, colorBuffer)
    imgColorflip = np.fromstring(colorBuffer, np.uint8).reshape( 600, 800, 3 )
    imgColor = cv2.flip(imgColorflip, 0)

    DepthBuffer = (GLfloat * 480000 )(0) # 480000 == 800*600
    glReadPixels(0, 0, windowWidth, windowHeight, GL_DEPTH_COMPONENT, GL_FLOAT, DepthBuffer)
    imgDepthflip = np.fromstring(DepthBuffer, np.float32).reshape( 600, 800, 1 )
    imgDepth = cv2.flip(imgDepthflip, 0) *255. # scaling from (0~1) up to 0~255
    imgDepth =imgDepth.astype(np.uint8)

    imshow("This is ColorBuffer window",imgColor)
    imshow("This is DepthBuffer window",imgDepth)
    waitKey(1)
    glutSwapBuffers()
```



openGL

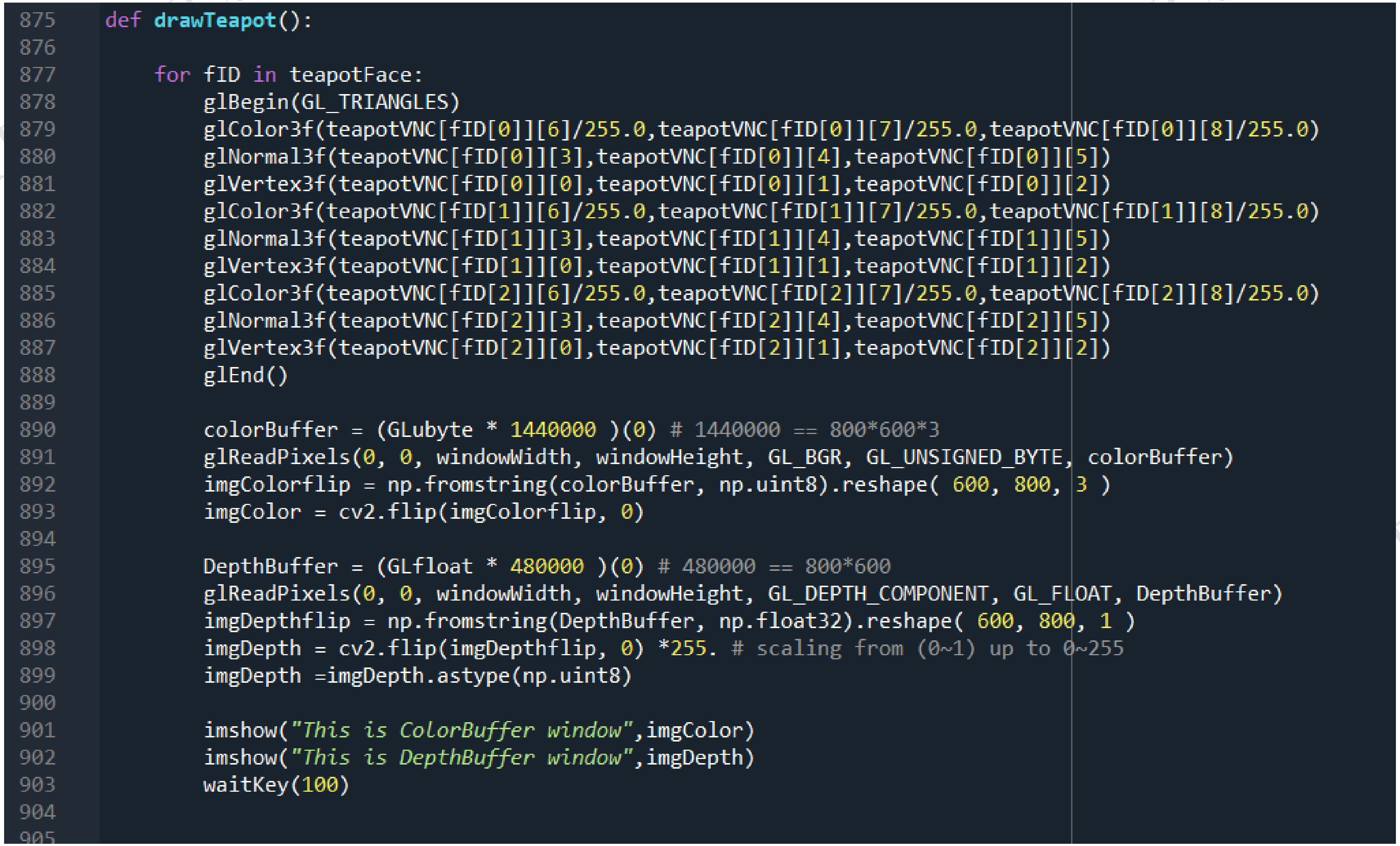


openCV



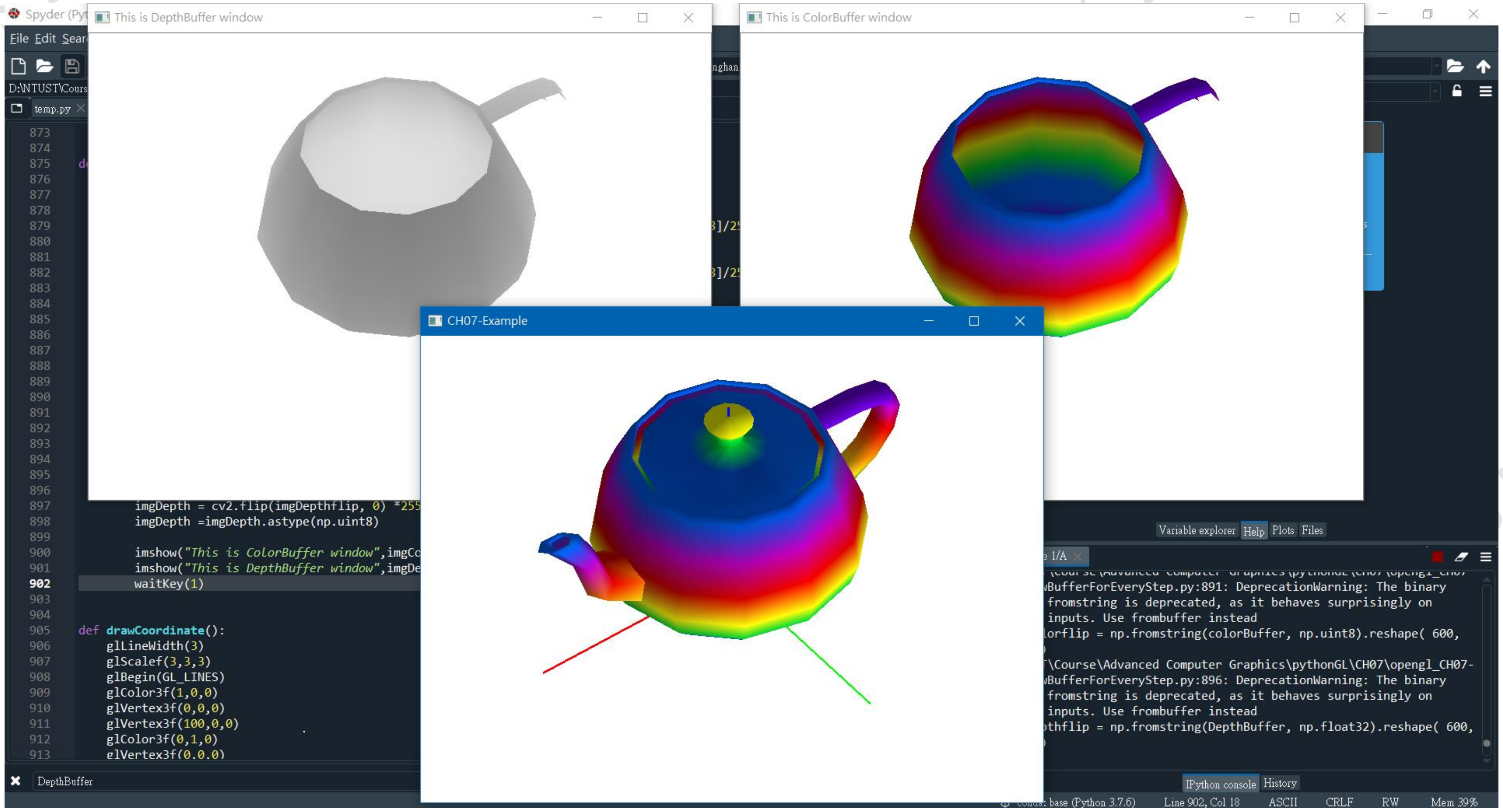
openCV

Show buffer for every step



```python
def drawTeapot():

    for fID in teapotFace:
        glBegin(GL_TRIANGLES)
        glColor3f(teapotVNC[fID[0]][6]/255.0,teapotVNC[fID[0]][7]/255.0,teapotVNC[fID[0]][8]/255.0)
        glNormal3f(teapotVNC[fID[0]][3],teapotVNC[fID[0]][4],teapotVNC[fID[0]][5])
        glVertex3f(teapotVNC[fID[0]][0],teapotVNC[fID[0]][1],teapotVNC[fID[0]][2])
        glColor3f(teapotVNC[fID[1]][6]/255.0,teapotVNC[fID[1]][7]/255.0,teapotVNC[fID[1]][8]/255.0)
        glNormal3f(teapotVNC[fID[1]][3],teapotVNC[fID[1]][4],teapotVNC[fID[1]][5])
        glVertex3f(teapotVNC[fID[1]][0],teapotVNC[fID[1]][1],teapotVNC[fID[1]][2])
        glColor3f(teapotVNC[fID[2]][6]/255.0,teapotVNC[fID[2]][7]/255.0,teapotVNC[fID[2]][8]/255.0)
        glNormal3f(teapotVNC[fID[2]][3],teapotVNC[fID[2]][4],teapotVNC[fID[2]][5])
        glVertex3f(teapotVNC[fID[2]][0],teapotVNC[fID[2]][1],teapotVNC[fID[2]][2])
        glEnd()

        colorBuffer = (GLubyte * 1440000 )(0) # 1440000 == 800*600*3
        glReadPixels(0, 0, windowWidth, windowHeight, GL_BGR, GL_UNSIGNED_BYTE, colorBuffer)
        imgColorflip = np.fromstring(colorBuffer, np.uint8).reshape( 600, 800, 3 )
        imgColor = cv2.flip(imgColorflip, 0)

        DepthBuffer = (GLfloat * 480000 )(0) # 480000 == 800*600
        glReadPixels(0, 0, windowWidth, windowHeight, GL_DEPTH_COMPONENT, GL_FLOAT, DepthBuffer)
        imgDepthflip = np.fromstring(DepthBuffer, np.float32).reshape( 600, 800, 1 )
        imgDepth = cv2.flip(imgDepthflip, 0) *255. # scaling from (0~1) up to 0~255
        imgDepth =imgDepth.astype(np.uint8)

        imshow("This is ColorBuffer window",imgColor)
        imshow("This is DepthBuffer window",imgDepth)
        waitKey(100)

```

Show buffer for every step

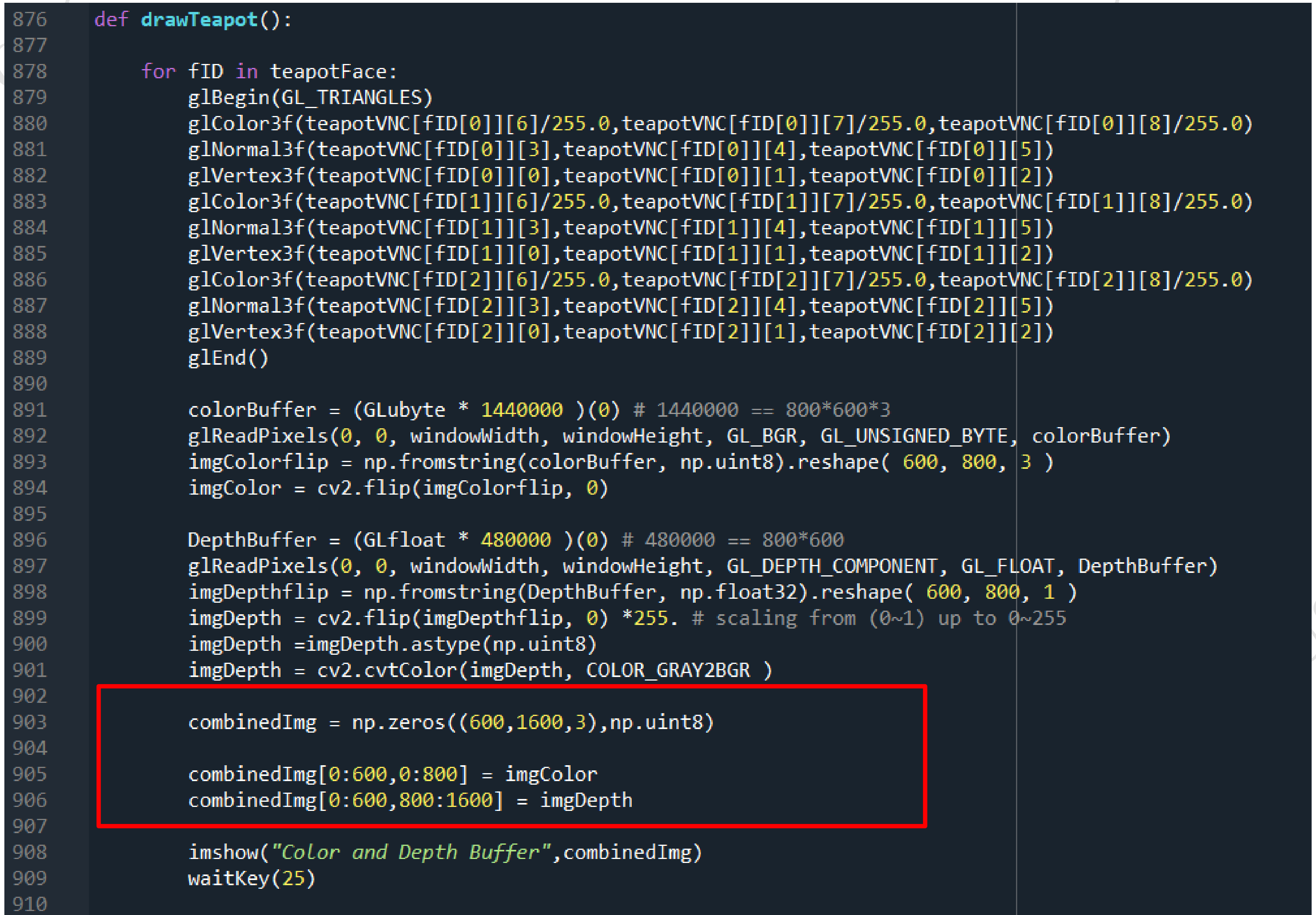


```
        imgDepth = cv2.flip(imgDepthflip, 0) *255
        imgDepth =imgDepth.astype(np.uint8)

        imshow("This is ColorBuffer window",imgCo
        imshow("This is DepthBuffer window",imgDe
        waitKey(1)


def drawCoordinate():
    glLineWidth(3)
    glScalef(3,3,3)
    glBegin(GL_LINES)
    glColor3f(1,0,0)
    glVertex3f(0,0,0)
    glVertex3f(100,0,0)
    glColor3f(0,1,0)
    glVertex3f(0.0.0)
```

```
wBufferForEveryStep.py:891: DeprecationWarning: The binary
 fromstring is deprecated, as it behaves surprisingly on
 inputs. Use frombuffer instead
lorflip = np.fromstring(colorBuffer, np.uint8).reshape( 600,

T\Course\Advanced Computer Graphics\pythonGL\CH07\opengl_CH07-
wBufferForEveryStep.py:896: DeprecationWarning: The binary
 fromstring is deprecated, as it behaves surprisingly on
 inputs. Use frombuffer instead
pthflip = np.fromstring(DepthBuffer, np.float32).reshape( 600,
```

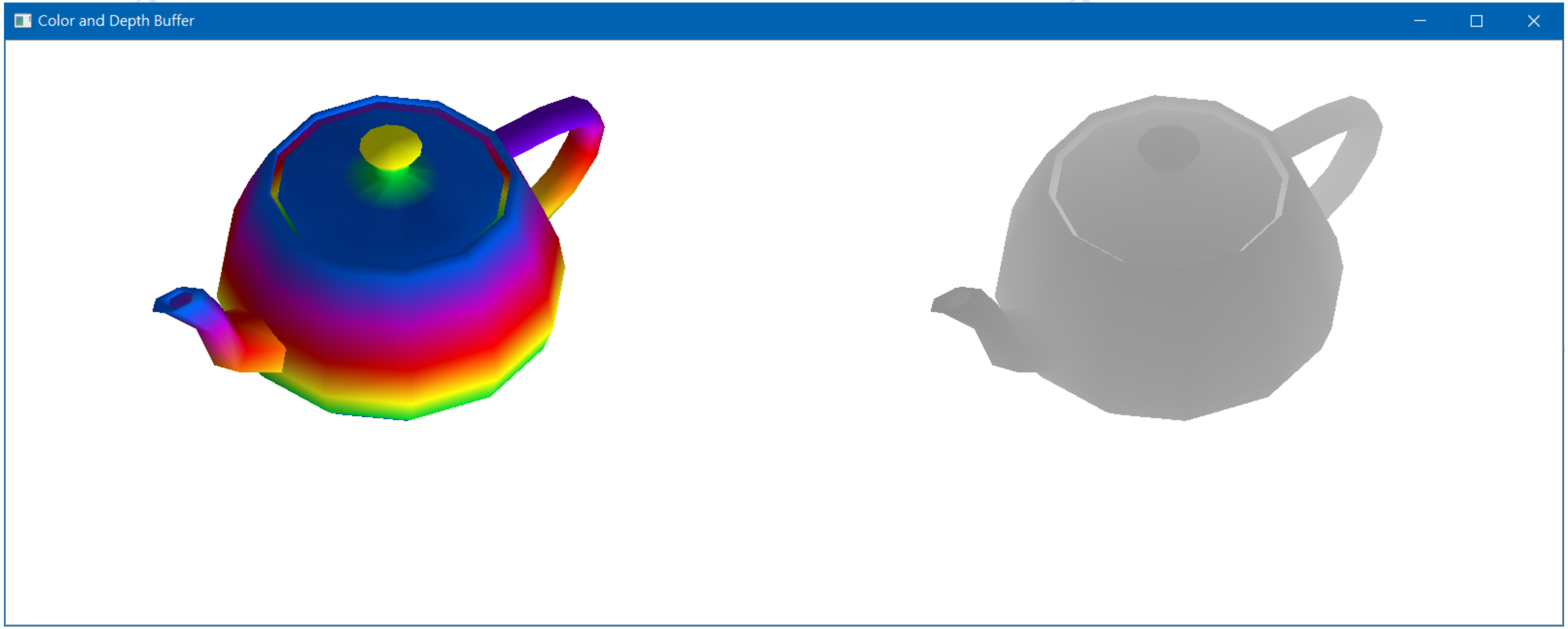
```
conda: base (Python 3.7.6)   Line 902, Col 18   ASCII   CRLF   RW   Mem 39%
```

openCV (combined images by ROI)

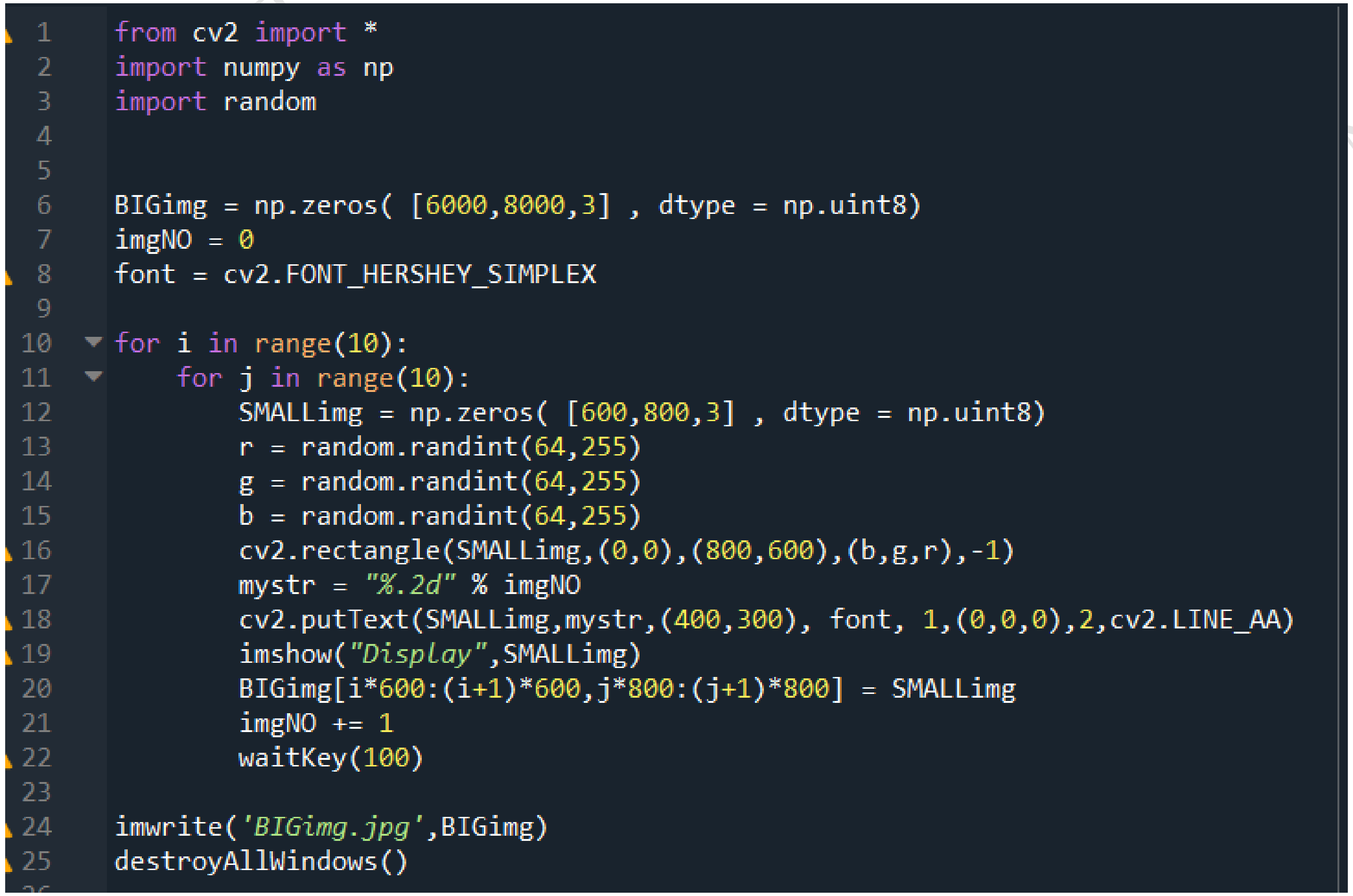


```
def drawTeapot():

    for fID in teapotFace:
        glBegin(GL_TRIANGLES)
        glColor3f(teapotVNC[fID[0]][6]/255.0,teapotVNC[fID[0]][7]/255.0,teapotVNC[fID[0]][8]/255.0)
        glNormal3f(teapotVNC[fID[0]][3],teapotVNC[fID[0]][4],teapotVNC[fID[0]][5])
        glVertex3f(teapotVNC[fID[0]][0],teapotVNC[fID[0]][1],teapotVNC[fID[0]][2])
        glColor3f(teapotVNC[fID[1]][6]/255.0,teapotVNC[fID[1]][7]/255.0,teapotVNC[fID[1]][8]/255.0)
        glNormal3f(teapotVNC[fID[1]][3],teapotVNC[fID[1]][4],teapotVNC[fID[1]][5])
        glVertex3f(teapotVNC[fID[1]][0],teapotVNC[fID[1]][1],teapotVNC[fID[1]][2])
        glColor3f(teapotVNC[fID[2]][6]/255.0,teapotVNC[fID[2]][7]/255.0,teapotVNC[fID[2]][8]/255.0)
        glNormal3f(teapotVNC[fID[2]][3],teapotVNC[fID[2]][4],teapotVNC[fID[2]][5])
        glVertex3f(teapotVNC[fID[2]][0],teapotVNC[fID[2]][1],teapotVNC[fID[2]][2])
        glEnd()

        colorBuffer = (GLubyte * 1440000 )(0) # 1440000 == 800*600*3
        glReadPixels(0, 0, windowWidth, windowHeight, GL_BGR, GL_UNSIGNED_BYTE, colorBuffer)
        imgColorflip = np.fromstring(colorBuffer, np.uint8).reshape( 600, 800, 3 )
        imgColor = cv2.flip(imgColorflip, 0)

        DepthBuffer = (GLfloat * 480000 )(0) # 480000 == 800*600
        glReadPixels(0, 0, windowWidth, windowHeight, GL_DEPTH_COMPONENT, GL_FLOAT, DepthBuffer)
        imgDepthflip = np.fromstring(DepthBuffer, np.float32).reshape( 600, 800, 1 )
        imgDepth = cv2.flip(imgDepthflip, 0) *255. # scaling from (0~1) up to 0~255
        imgDepth =imgDepth.astype(np.uint8)
        imgDepth = cv2.cvtColor(imgDepth, COLOR_GRAY2BGR )

        combinedImg = np.zeros((600,1600,3),np.uint8)

        combinedImg[0:600,0:800] = imgColor
        combinedImg[0:600,800:1600] = imgDepth

        imshow("Color and Depth Buffer",combinedImg)
        waitKey(25)

```

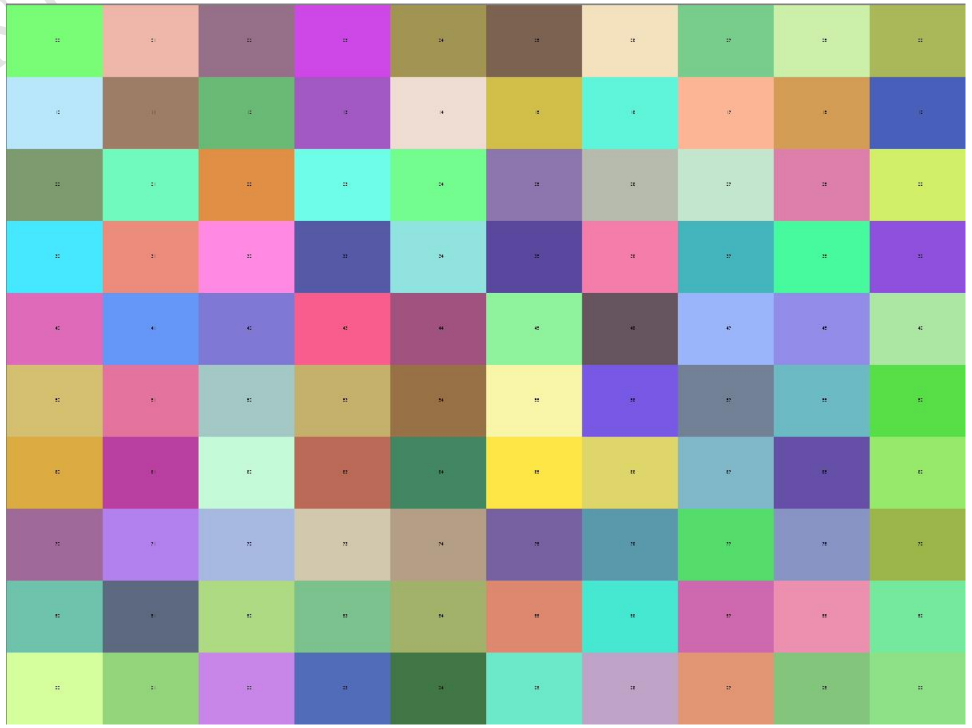
openCV (combined images by ROI)



openCV ROI (mosaic)



```python
from cv2 import *
import numpy as np
import random


BIGimg = np.zeros( [6000,8000,3] , dtype = np.uint8)
imgNO = 0
font = cv2.FONT_HERSHEY_SIMPLEX

for i in range(10):
    for j in range(10):
        SMALLimg = np.zeros( [600,800,3] , dtype = np.uint8)
        r = random.randint(64,255)
        g = random.randint(64,255)
        b = random.randint(64,255)
        cv2.rectangle(SMALLimg,(0,0),(800,600),(b,g,r),-1)
        mystr = "%.2d" % imgNO
        cv2.putText(SMALLimg,mystr,(400,300), font, 1,(0,0,0),2,cv2.LINE_AA)
        imshow("Display",SMALLimg)
        BIGimg[i*600:(i+1)*600,j*800:(j+1)*800] = SMALLimg
        imgNO += 1
        waitKey(100)

imwrite('BIGimg.jpg',BIGimg)
destroyAllWindows()
```



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

Find us on: facebook