Advanced Computer Graphics

Lecture-08 Introduction to OpenGL-11

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

- Shader
- OpenGL and Image Processing

Shader

1. Define vertex shader (C/C++ language)
2. Define fragment shader (C/C++ language)
3. Compile shader (in python) and assign a name
4. Use shader (what you compile)

https://www.opengl.org/sdk/docs/tutorials/ClockworkCoders/attributes.php

Shader: vertex shader for Phong shading

- Note: this is python code but we embed “c code” as the comment of the variable “vertex_code”.
  This is the part for assisting “openGL” to communicate with “shader”
- What target we deal with is “vertex”

```
962 vertex_code = """
963 varying vec4 vColor;
964 varying vec3 N;
965 varying vec3 v;
966
967 void main(void)
968 {
969     vColor = gl_Color;
970     v = vec3(gl_ModelViewMatrix * gl_Vertex);
971     N = normalize(gl_NormalMatrix * gl_Normal);
972
973     gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
974 }
975 """
```

Shader: fragment shader for Phong shading

- Since we already get “vertex” properties in “vertex shader”, then what target we deal with is “pixel” (fragment shading)

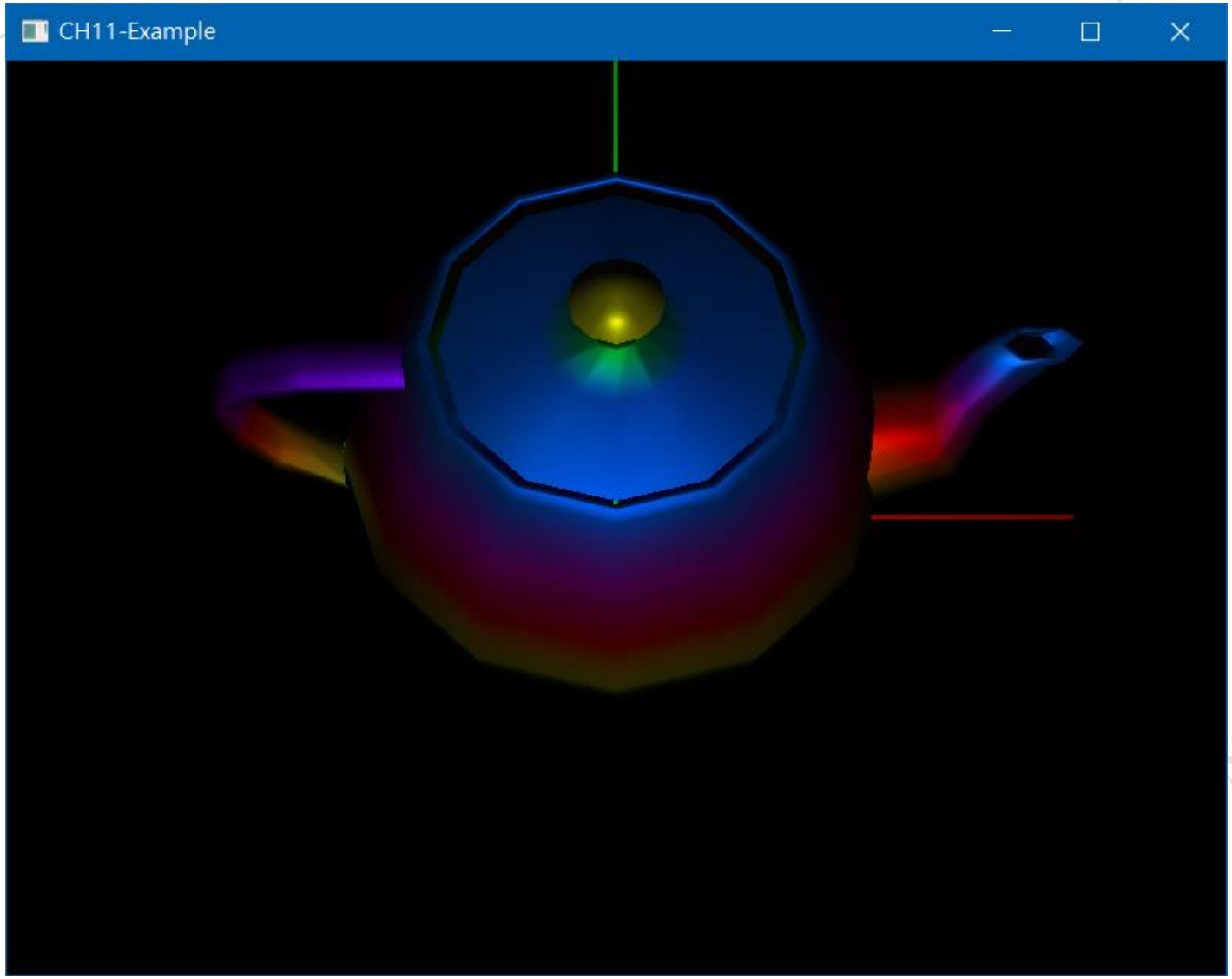
```
977  fragment_code = """
978  varying vec4 vColor;
979  varying vec3 N;
980  varying vec3 v;
981
982  void main (void)
983  {
984     vec3 L = normalize(gl_LightSource[0].position.xyz - v);
985     vec3 V = normalize(-v); // we are in Eye Coordinates, so EyePos is (0,0,0)
986     vec3 R = normalize(-reflect(L,N));
987
988     //calculate Ambient Term:
989     vec4 Iamb = gl_FrontLightProduct[0].ambient*vColor;;
990
991     //calculate Diffuse Term:
992     vec4 Idiff = gl_FrontLightProduct[0].diffuse * max(dot(N,L), 0.0)*vColor;
993     Idiff = clamp(Idiff, 0.0, 1.0);
994
995     // calculate Specular Term:
996     vec4 Ispec = gl_FrontLightProduct[0].specular * pow( max(dot(R,V),0.0),2.0)*vColor;
997     Ispec = clamp(Ispec, 0.0, 1.0);
998
999     // write Total Color:
1000    gl_FragColor = Iamb + Idiff + Ispec;
1001 }
1002 """
```

Shader: SOP

```
program  = glCreateProgram()
vertex   = glCreateShader(GL_VERTEX_SHADER)
fragment = glCreateShader(GL_FRAGMENT_SHADER)
```

```
1005    # Set shaders source
1006    glShaderSource(vertex, vertex_code)
1007    glShaderSource(fragment, fragment_code)
1008
1009    # Compile shaders
1010    glCompileShader(vertex)
1011    glCompileShader(fragment)
1012    my_shader = glCreateProgram()
1013    glAttachShader(my_shader, vertex)
1014    glAttachShader(my_shader, fragment)
1015
1016    glLinkProgram(my_shader)
1017    glUseProgram(my_shader)
1018    glClearColor(0,0,0,1)
1019    glutMainLoop()
```

Phong Shading: Result



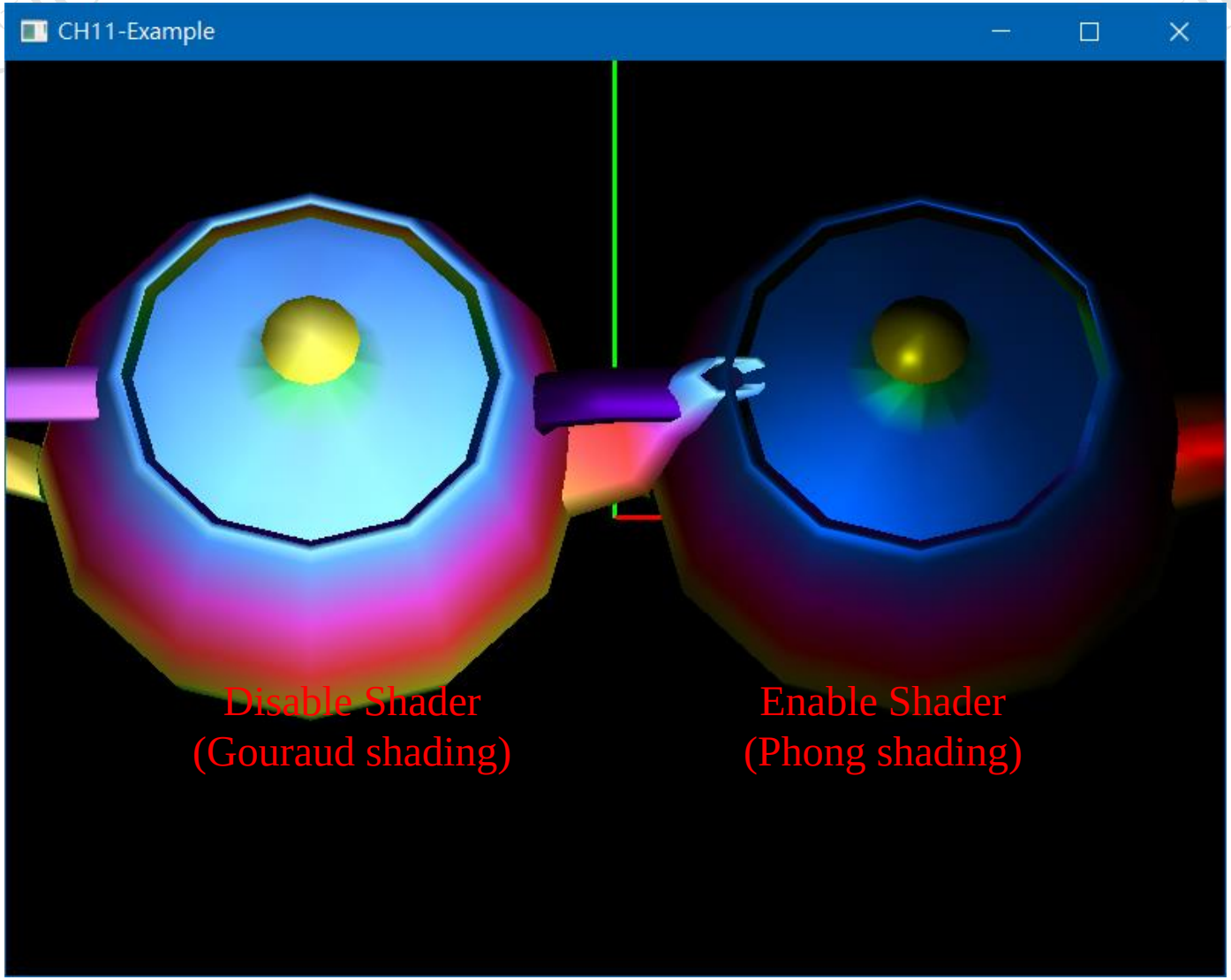
Enable/Disable Shading language

```
927     glMaterialfv(GL_FRONT_AND_BACK, GL_SHININESS,2)
928
929     glEnable(GL_LIGHTING)
930     glTranslatef(0,0,-1000)
931     glPushMatrix()
932     global angle
933     glRotatef(angle,1,0,0)
934     angle = angle + 0.5
935     global my_shader
936     glUseProgram(my_shader)
937     glTranslatef(200,0,0)
938     drawTeapot()
939
940     glUseProgram(0)
941     glTranslatef(-400,0,0)
942     drawTeapot()
943     glPopMatrix()
944     glDisable(GL_LIGHTING)
945     drawCoordinate()
946     glutSwapBuffers()
947     glutPostRedisplay()
948
```

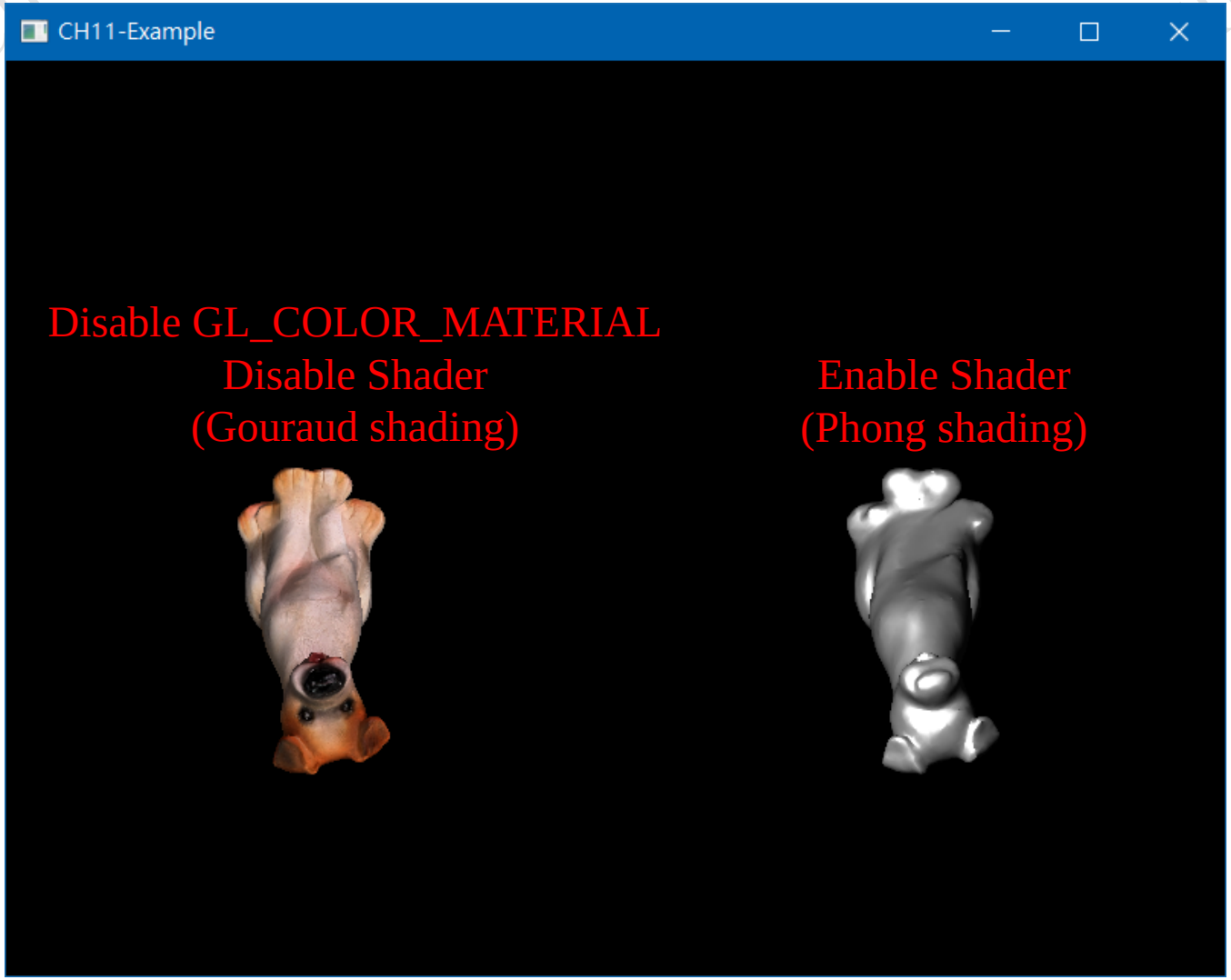
Draw something with shader

Draw something with Nothing
(switch back to conventional openGL)

Enable/Disable Shading language



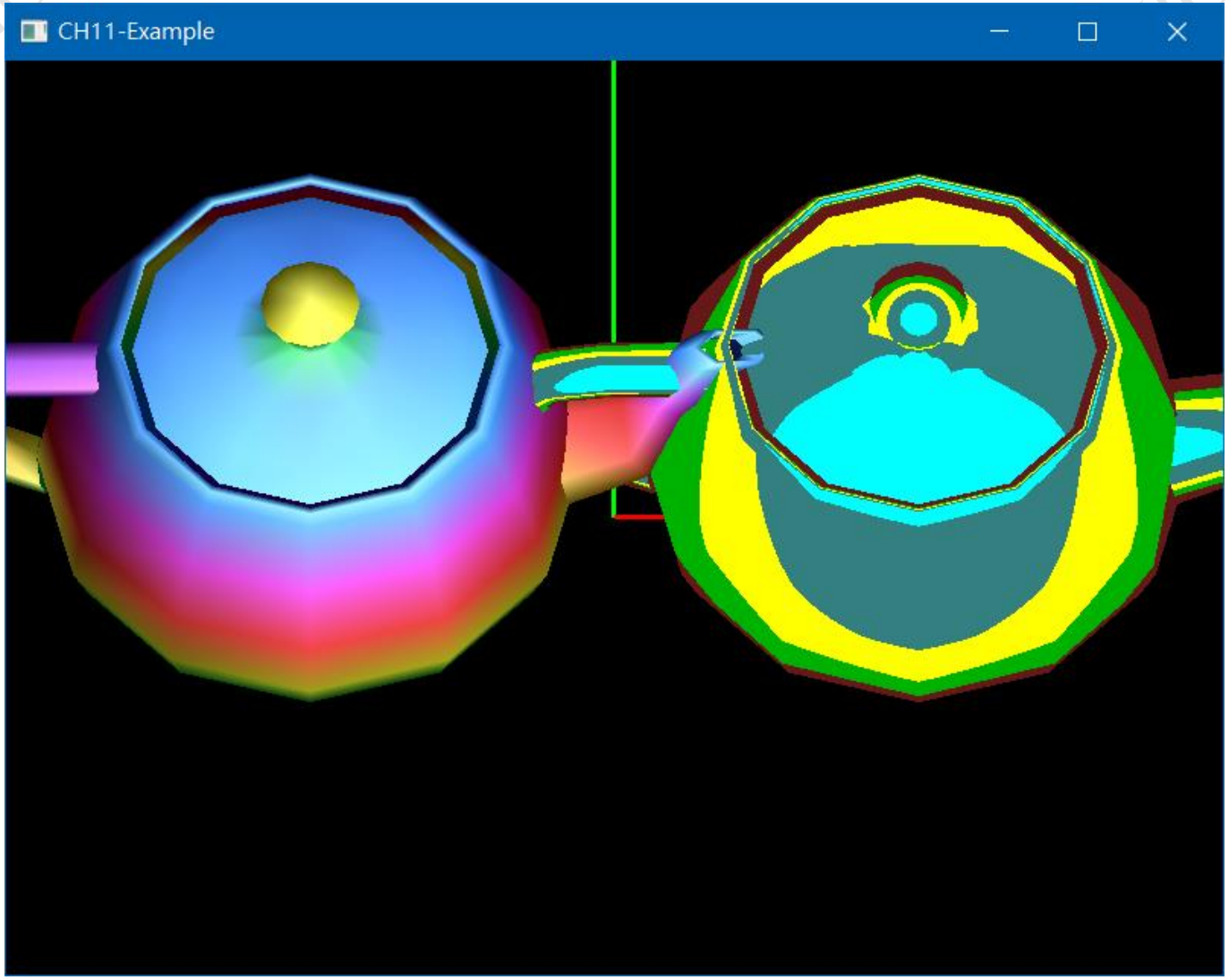
Enable/Disable Shading language



Cartoon shading

```
program  = glCreateProgram()
vertex   = glCreateShader(GL_VERTEX_SHADER)
fragment = glCreateShader(GL_FRAGMENT_SHADER)
vertex_code = """
varying vec3 normal, lightDir;

void main()
{
    lightDir = normalize(vec3(gl_LightSource[0].position));
    normal = normalize(gl_NormalMatrix * gl_Normal);

    gl_Position = ftransform();
}
"""

fragment_code = """
varying vec3 normal, lightDir;

void main()
{
    float intensity;
    vec3 n;
    vec4 color;

    n = normalize(normal);
    intensity = max(dot(lightDir,n),0.0);

    if (intensity > 0.9) color = vec4(0.0,1.0,1.0,1.0);
    else if (intensity > 0.7) color = vec4(0.2,0.5,0.5,1.0);
    else if (intensity > 0.5) color = vec4(1.0,1.0,0.0,1.0);
    else if (intensity > 0.25) color = vec4(0.0,0.7,0.0,1.0);
    else color = vec4(0.4,0.1,0.1,1.0);

    gl_FragColor = color;
}

"""
```

Cartoon shading



Cartoon shading

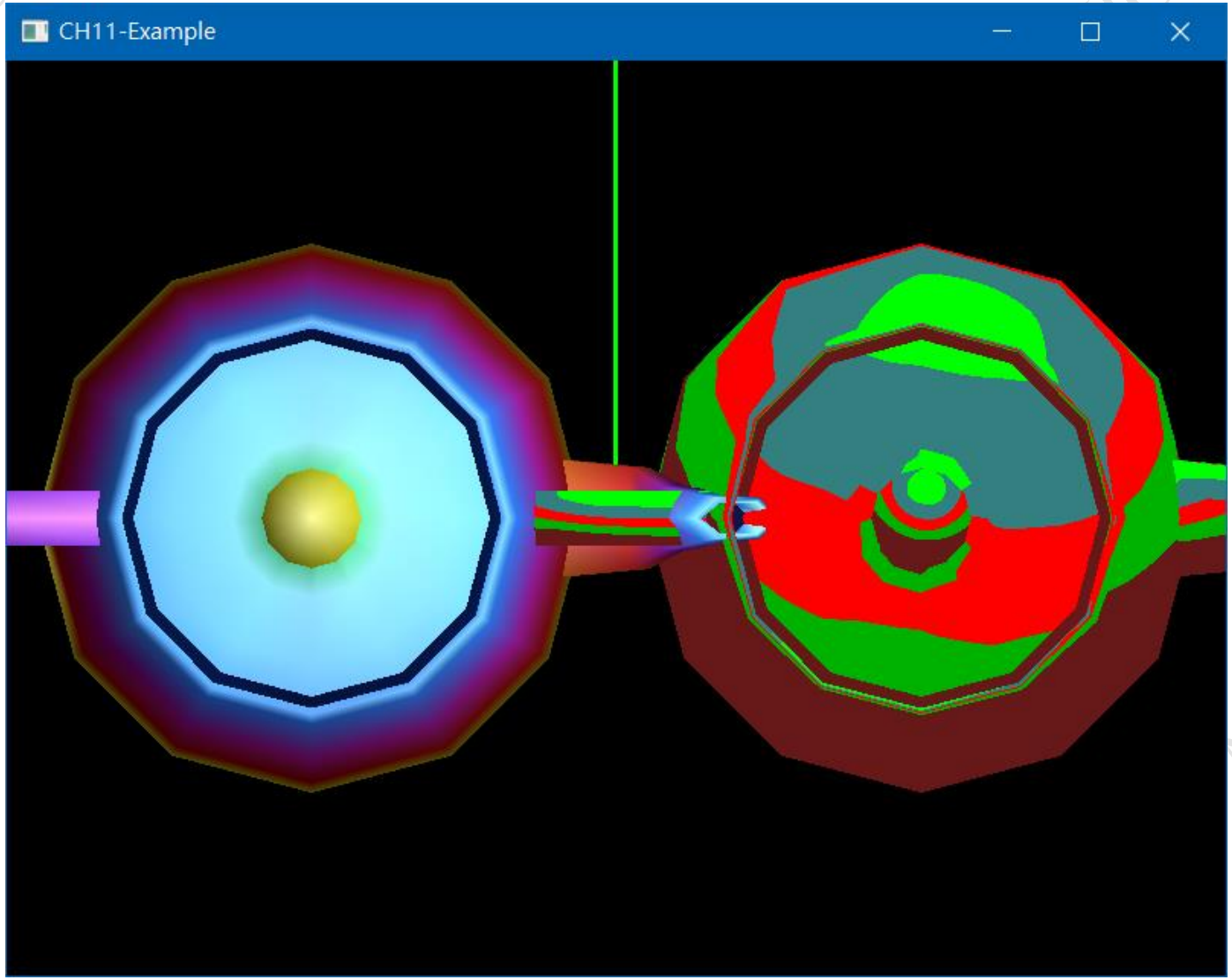
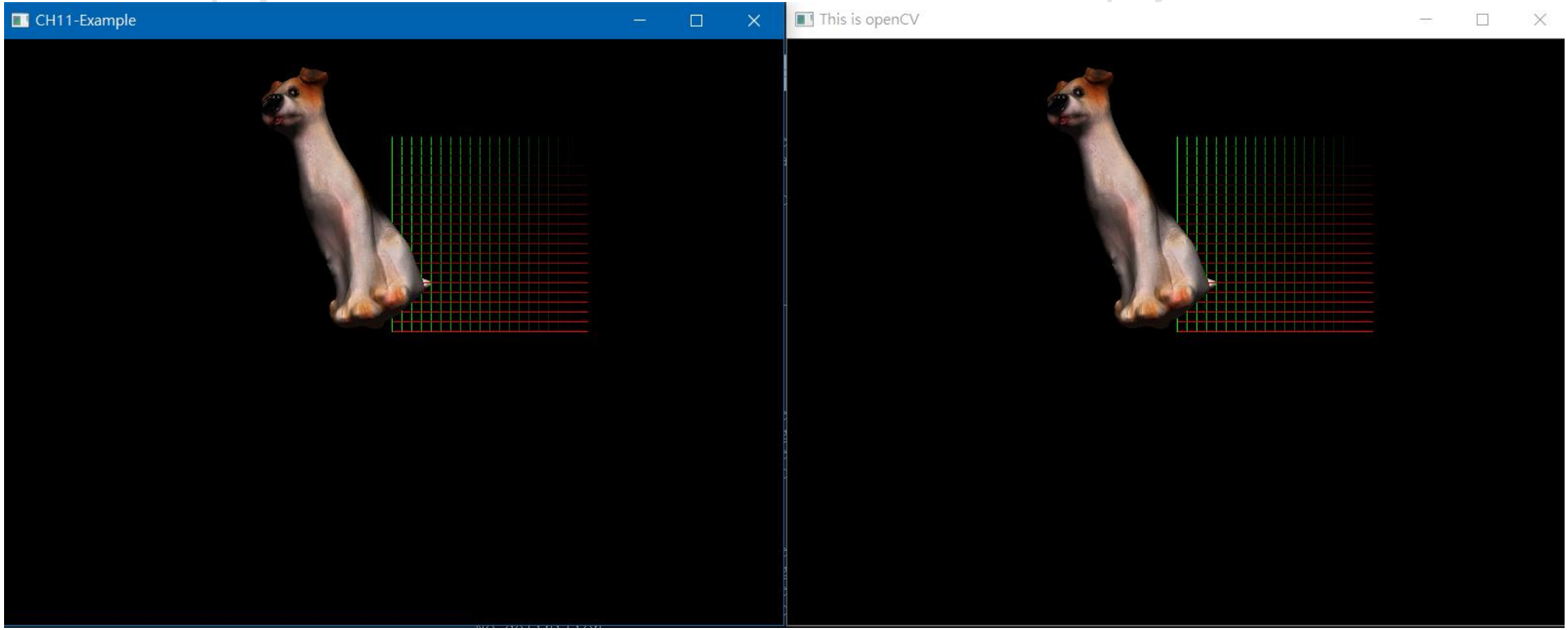
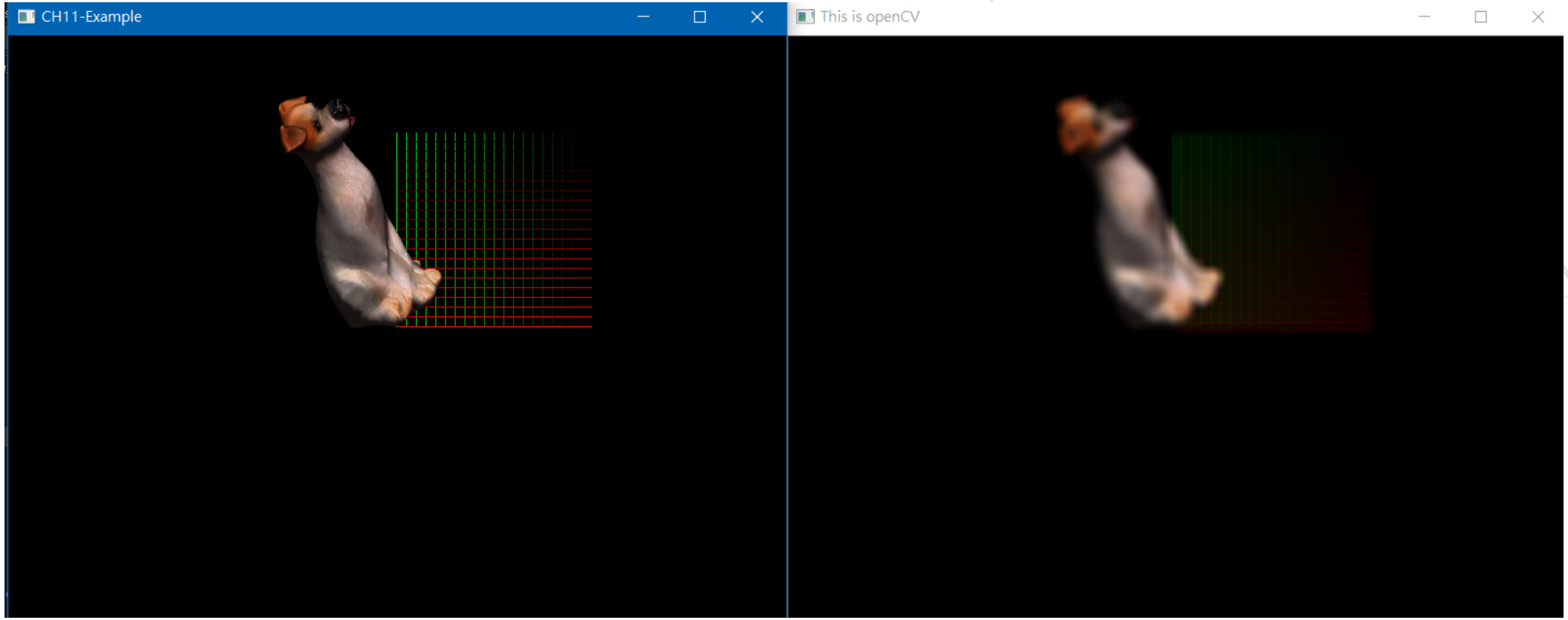


Image Processing by openCV



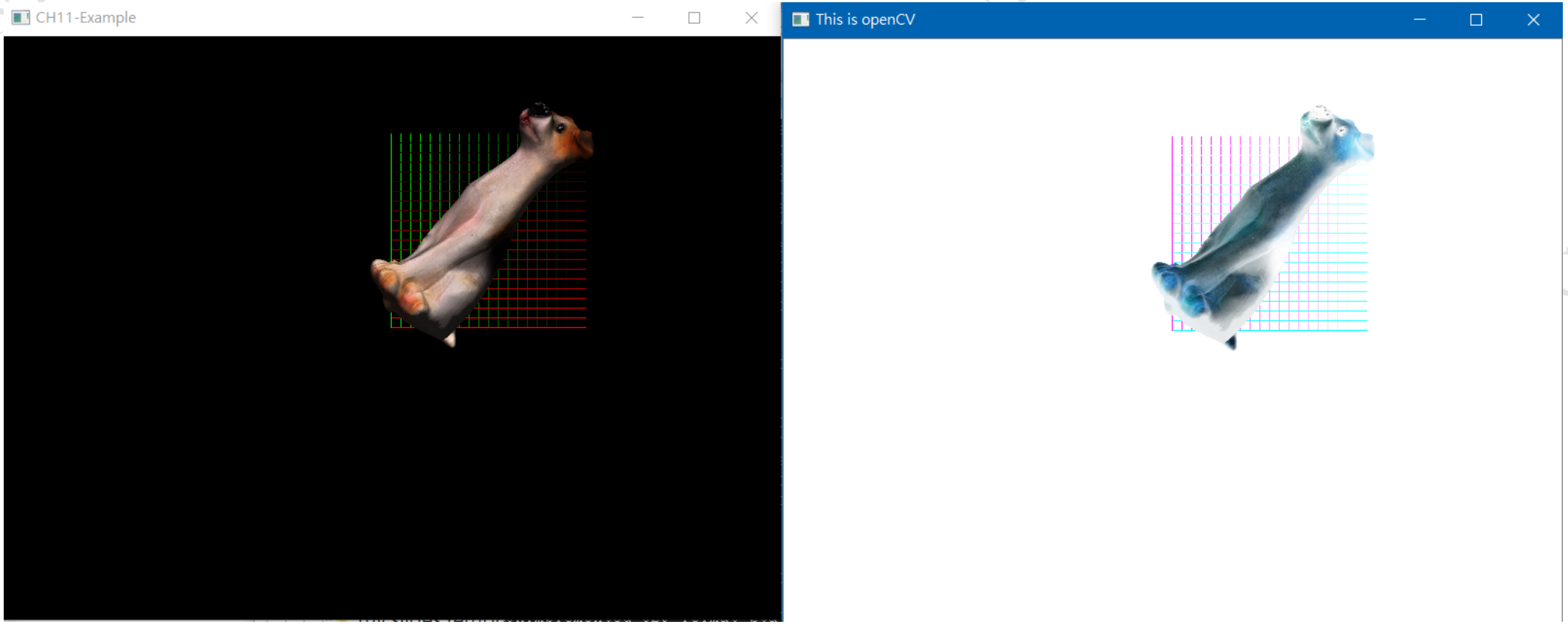
Blurring effect

```
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)

imgColor = cv2.blur(imgColor,(11,11))

imshow("This is openCV",imgColor)
waitKey(1)
```



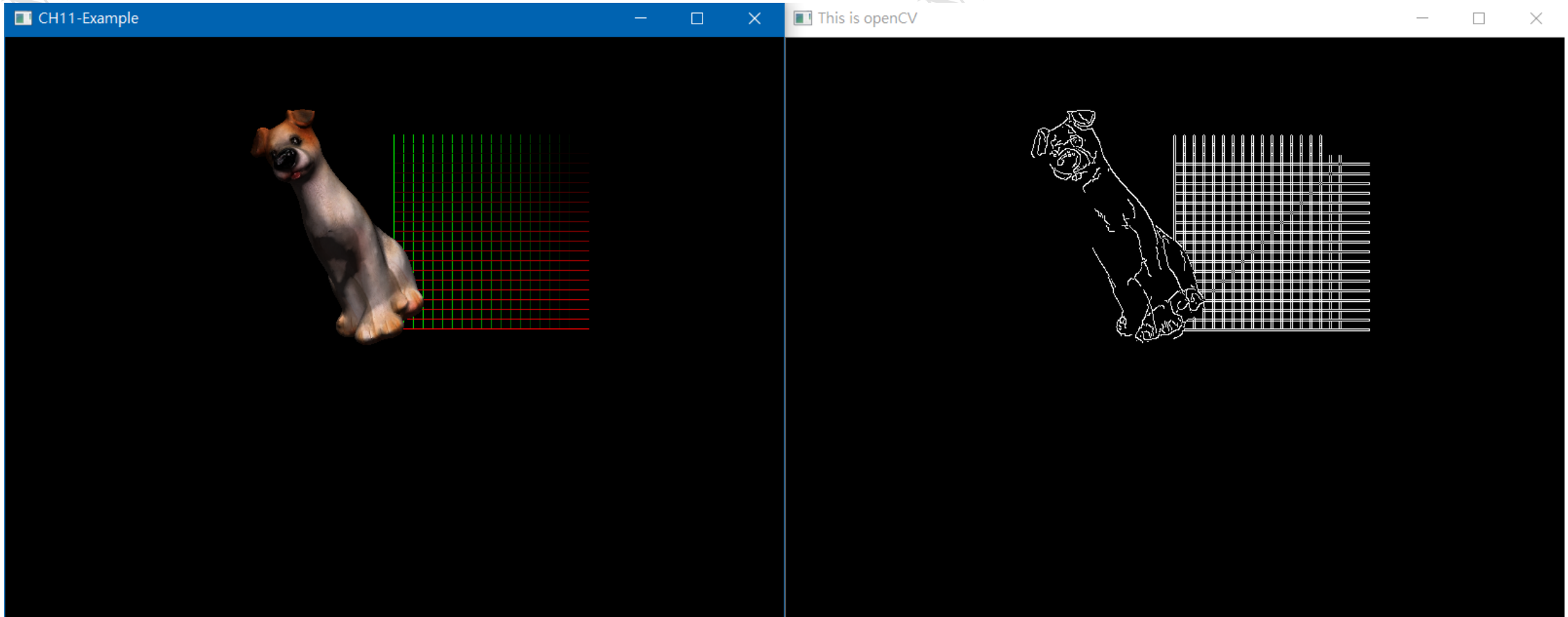
Inverse Color

```
62
63      colorBuffer = (GLubyte * 1440000 )(0) # 1440000 == 800*600*3
64      glReadPixels(0, 0, windowWidth, windowHeight, GL_BGR, GL_UNSIGNED_BYTE, colorBuffer)
65      imgColorflip = np.fromstring(colorBuffer, np.uint8).reshape( 600, 800, 3 )
66      imgColor = cv2.flip(imgColorflip, 0)
67
68      imgColor = 255- imgColor
69
70      imshow("This is openCV",imgColor)
71      waitKey(1)
```



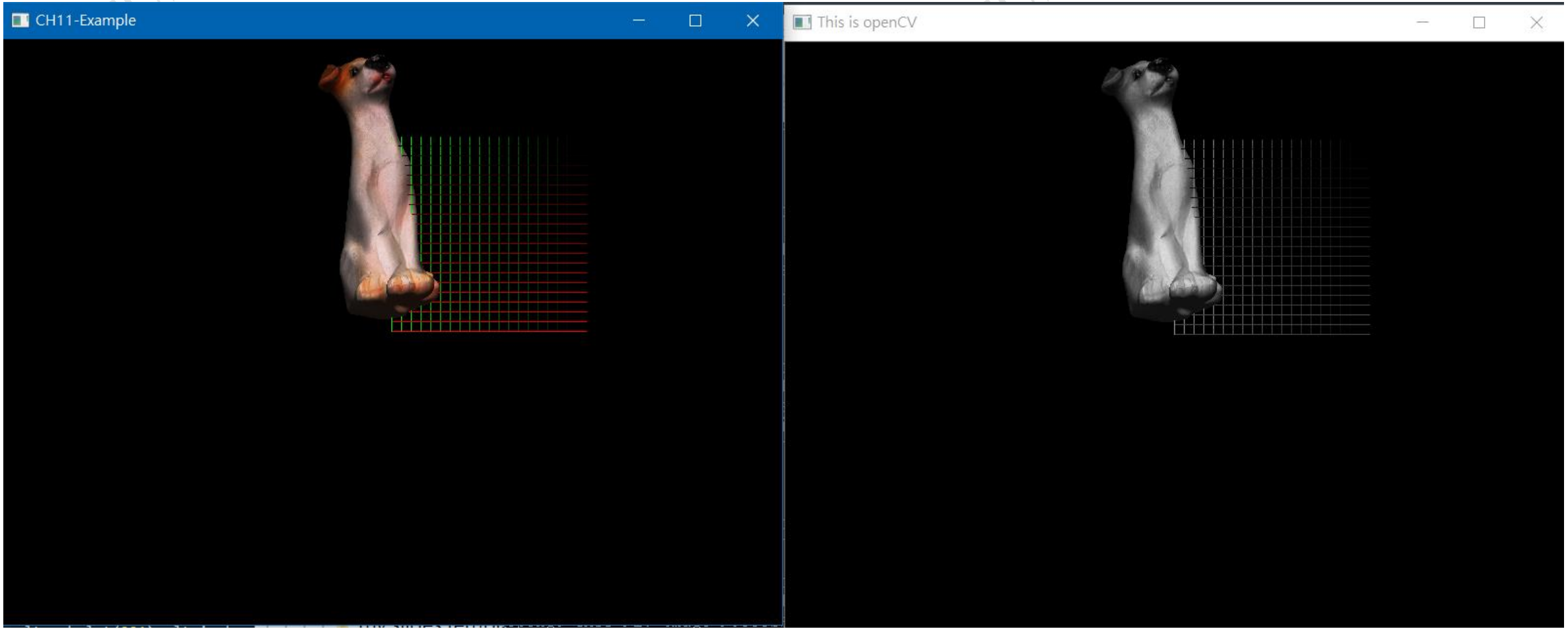
Edge detection

```
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)

imgEdges = cv2.Canny(imgColor,100,200)

imshow("This is openCV",imgEdges)
waitKey(1)
```



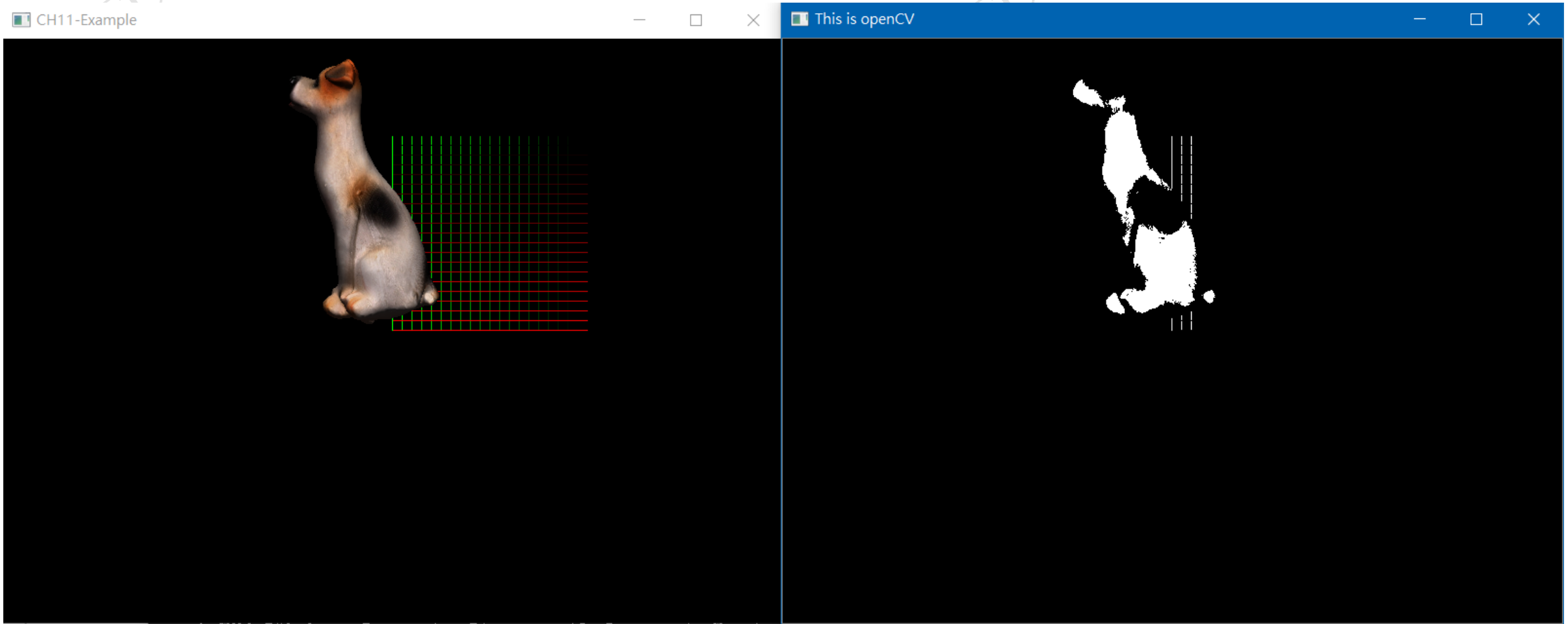
Grey image

```
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)

imgGrey = cv2.cvtColor(imgColor, cv2.COLOR_BGR2GRAY)

imshow("This is openCV",imgGrey)
waitKey(1)
```



Binary

```
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)

imgGrey = cv2.cvtColor(imgColor, cv2.COLOR_BGR2GRAY)

ret, imgBinary = cv2.threshold(imgGrey,127,255,cv2.THRESH_BINARY)

imshow("This is openCV",imgBinary)
waitKey(1)
```



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

Find us on: facebook