Advanced Computer Graphics

Lecture-08 Introduction to OpenGL-2

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

Program GUI format

Console Mode (win32)

QT/VTK等其他的軟體工具

視窗形式
wxWidget/ MFC / C++ →使用 glu 或 glfw 取代GDI
WindowForm (C#)

Console mode: based on glut

```
glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA)
glutCreateWindow(b'Hello world!')
glutReshapeWindow(512,512)
glutReshapeFunc(reshape)
glutDisplayFunc(display)
glutKeyboardFunc(keyboard)
glutMainLoop()
```

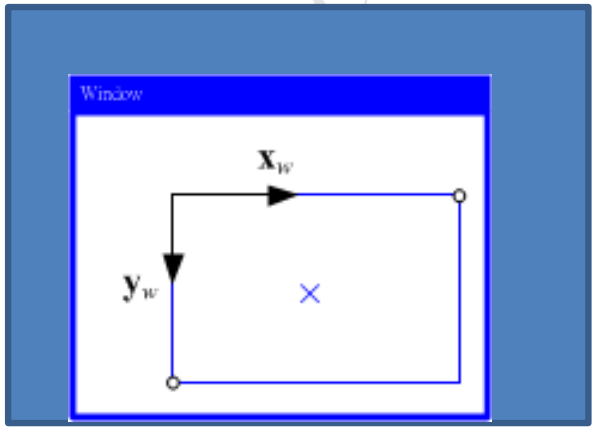
→Initialize device
→Setting display mode
→Create windows with a caption of “Hello world”
→Adjust window size
→Define the action when you resize the window
→Define what you want to draw
→Define the action when you press button
→Entering the main loop (event triggered)

Console mode: based on glfw

```
// glfw: initialize and configure
// ------------------------------
glfwInit();
glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);


// glfw window creation
// --------------------
GLFWwindow* window = glfwCreateWindow(SCR_WIDTH, SCR_HEIGHT, "Advanced CG Example", NULL, NULL);
if (window == NULL) ;
glfwMakeContextCurrent(window);
glfwSetFramebufferSizeCallback(window, framebuffer_size_callback);
```

OpenGL Programming- Preparation

- Before rendering 3D objects (setting for one time only)
  - Initialize GL device (color bit)
  - Define “window” or “full screen” type
  - Define the size of canvas
  - Import Texture Image (optional)
  - Enable all constant setting, ex. GL_DEPTH_TEST



OpenGL Programming- Runtime

- Before drawing something
  - Clear buffer
  - Knowing the position of camera and project type
  - Knowing the “viewing volume”
  - Re-setting the matrix of object/view
- After drawing something
  - Swapping buffer
  - Trigger redraw event (optional)
  - Release “handle” (optional)

Online document or MSDN library

glBegin, glEnd

The **glBegin** and **glEnd** functions delimit the vertices of a primitive or a group of like primitives.

```
void glBegin(
  GLenum mode
);

void glEnd(
  void
);
```

Parameters

mode
The primitive or primitives that will be created from vertices presented between **glBegin** and the subsequent **glEnd**. The following are accepted symbolic constants and their meanings:

GL_POINTS
Treats each vertex as a single point. Vertex *n* defines point *n*. *N* points are drawn.

GL_LINES
Treats each pair of vertices as an independent line segment. Vertices *2n – 1* and *2n* define line *n*. *N/2* lines are drawn.

GL_LINE_STRIP
Draws a connected group of line segments from the first vertex to the last. Vertices *n* and *n+1* define line *n*. *N – 1* lines are drawn.

GL_LINE_LOOP
Draws a connected group of line segments from the first vertex to the last, then back to the first. Vertices *n* and *n+1* define line *n*. The last line, however, is defined by vertices *N* and *1*. *N* lines are drawn.

GL_TRIANGLES
Treats each triplet of vertices as an independent triangle. Vertices *3n – 2*, *3n –1*, and *3n* define triangle *n*. *N/3* triangles are drawn.

GL_TRIANGLE_STRIP
Draws a connected group of triangles. One triangle is defined for each vertex presented after the first two vertices. For odd *n*, vertices *n*, *n + 1*, and *n + 2* define triangle *n*. For even *n*, vertices *n + 1*, *n*, and *n + 2* define triangle *n*. *N – 2* triangles are drawn.

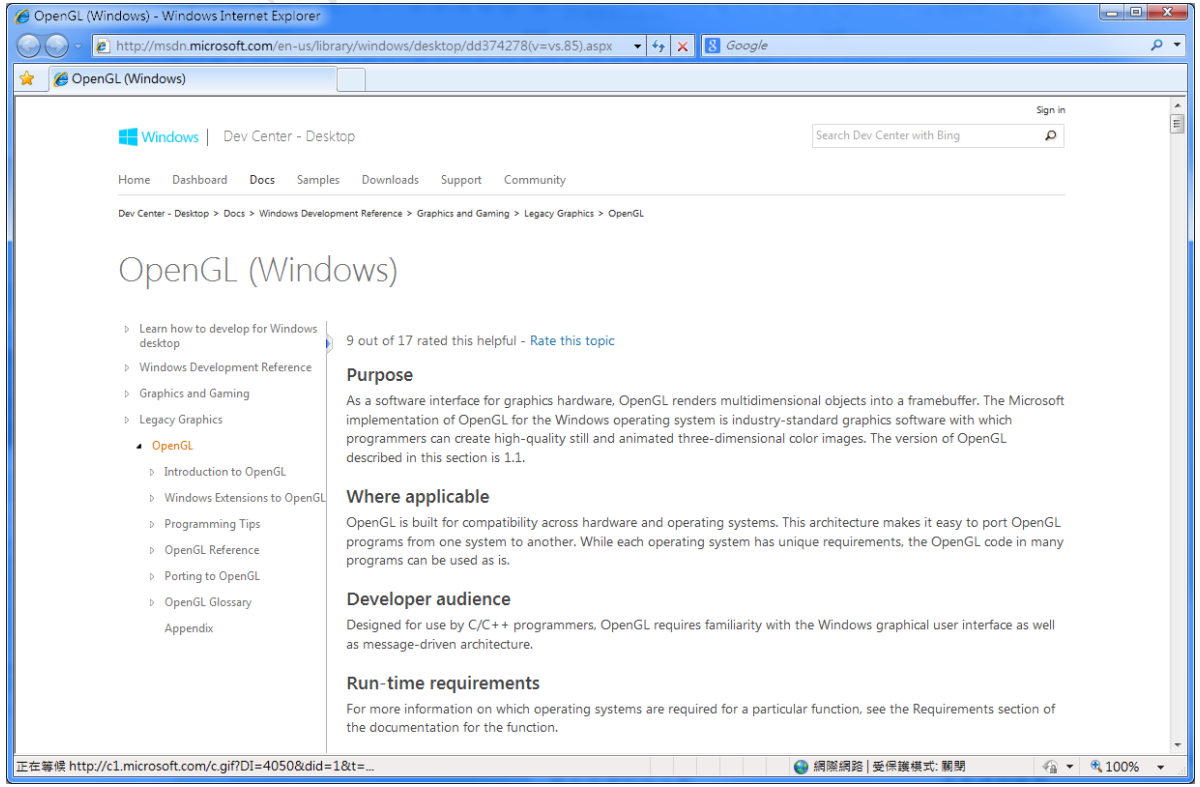
GL_TRIANGLE_FAN
Draws a connected group of triangles. One triangle is defined for each vertex presented after the first two vertices. Vertices *1*, *n + 1*, and *n + 2* define triangle *n*. *N – 2* triangles are drawn.

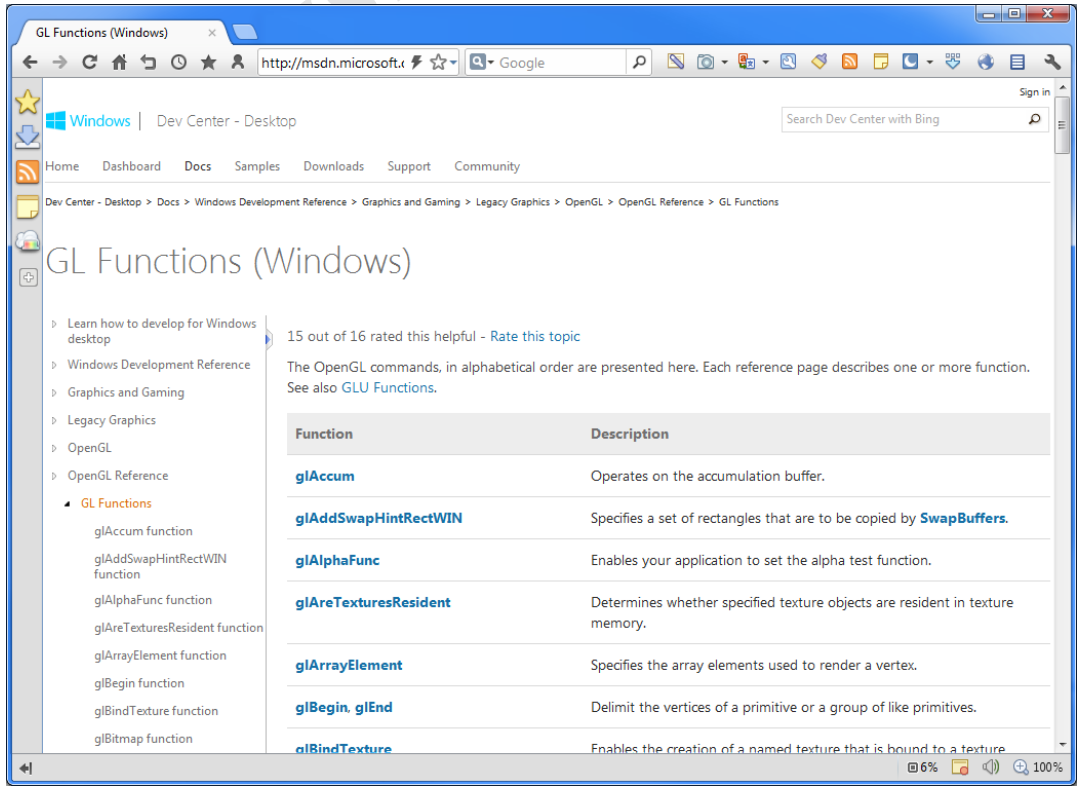
GL_QUADS
Treats each group of four vertices as an independent quadrilateral. Vertices *4n – 3*, *4n – 2*, *4n – 1*, and *4n* define quadrilateral *n*. *N/4* quadrilaterals are drawn.

```
//Draw points
glColor(….);
glBegin(GL_POINTS);
        glVertex……

glEnd();
```

```
glBegin(GL_POINTS);

        glColor(….);
         glVertex……

glEnd();
```

MSDN documentation





http://msdn.microsoft.com/en-us/library/windows/desktop/dd374278(v=vs.85).aspx

OpenGL draw: glBegin() … glEnd()

- Use glBegin() and glEnd() for drawing in a specific shape, ex. point, line, trangle.

glBegin(……)
- GL_POINTS：點
- GL_LINES：線
- GL_LINE_STRIP：線
- GL_LINE_LOOP：線
- GL_TRIANGLES：面
- GL_TRIANGLE_STRIP：面
- GL_TRIANGLE_FAN：面
- GL_QUADS：面
- GL_QUAD_STRIP：面
- GL_POLYGON：面

Example (C/C++)

```
glColor(….);
glBegin(GL_POINTS);
        glVertex……

glEnd();
```

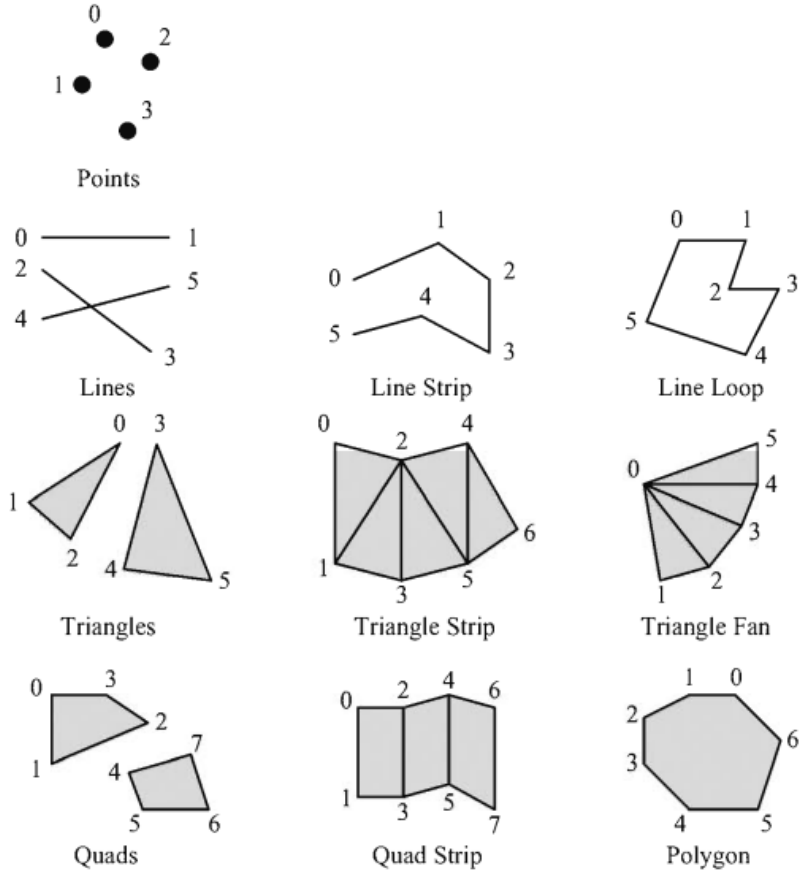
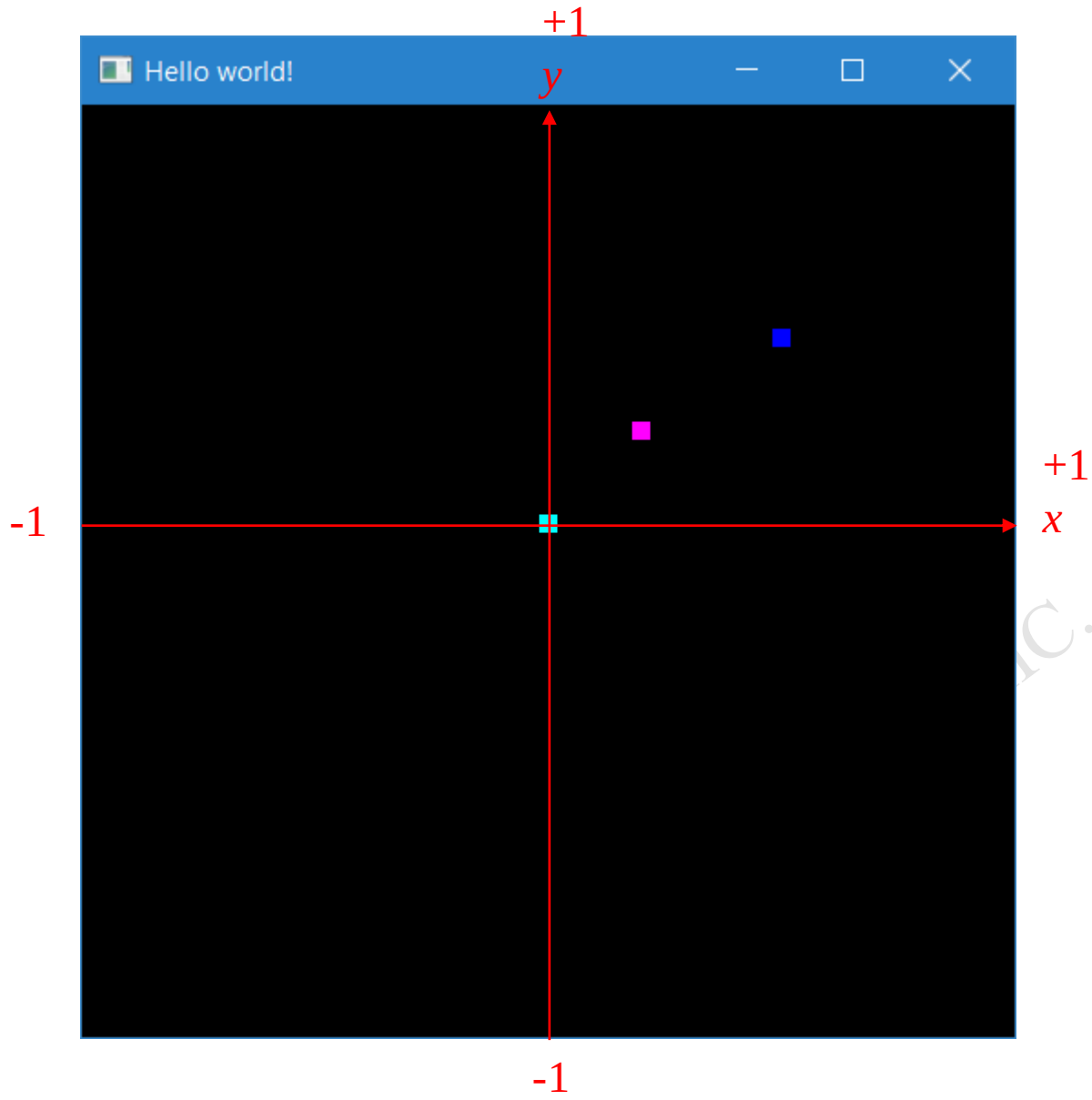


Figure 0.1 The OpenGL library defines ten types of graphics primitive. The numbers indicate the order in which the vertices are specified for each primitive type.

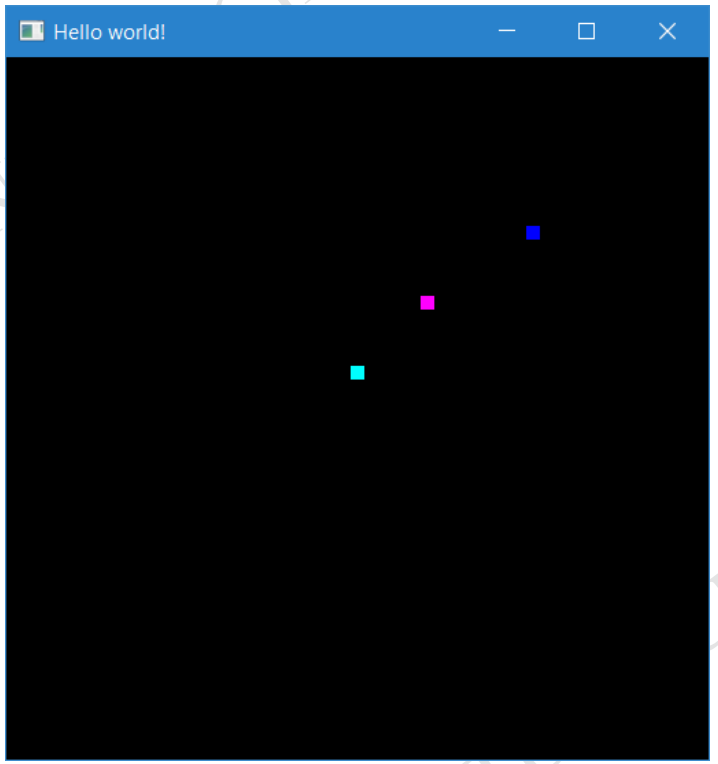
OpenGL draw: example

```
import sys
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *

def display():
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
    glPushMatrix()
    glPointSize(10)
    glBegin(GL_POINTS)
    glColor3f(0.0, 0.0, 1.0)
    glVertex3f(0.5,0.5,0)
    glColor3f(1.0, 0.0, 1.0)
    glVertex3f(0.2,0.3,0)
    glColor3f(0.0, 1.0, 1.0)
    glVertex3f(0.0,0.1,0)
    glEnd()
    glPopMatrix()
    glutSwapBuffers()

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'Hello world!')
glutReshapeWindow(512,512)
glutReshapeFunc(reshape)
glutDisplayFunc(display)
glutKeyboardFunc(keyboard)
glutMainLoop()
```



OpenGL draw: example

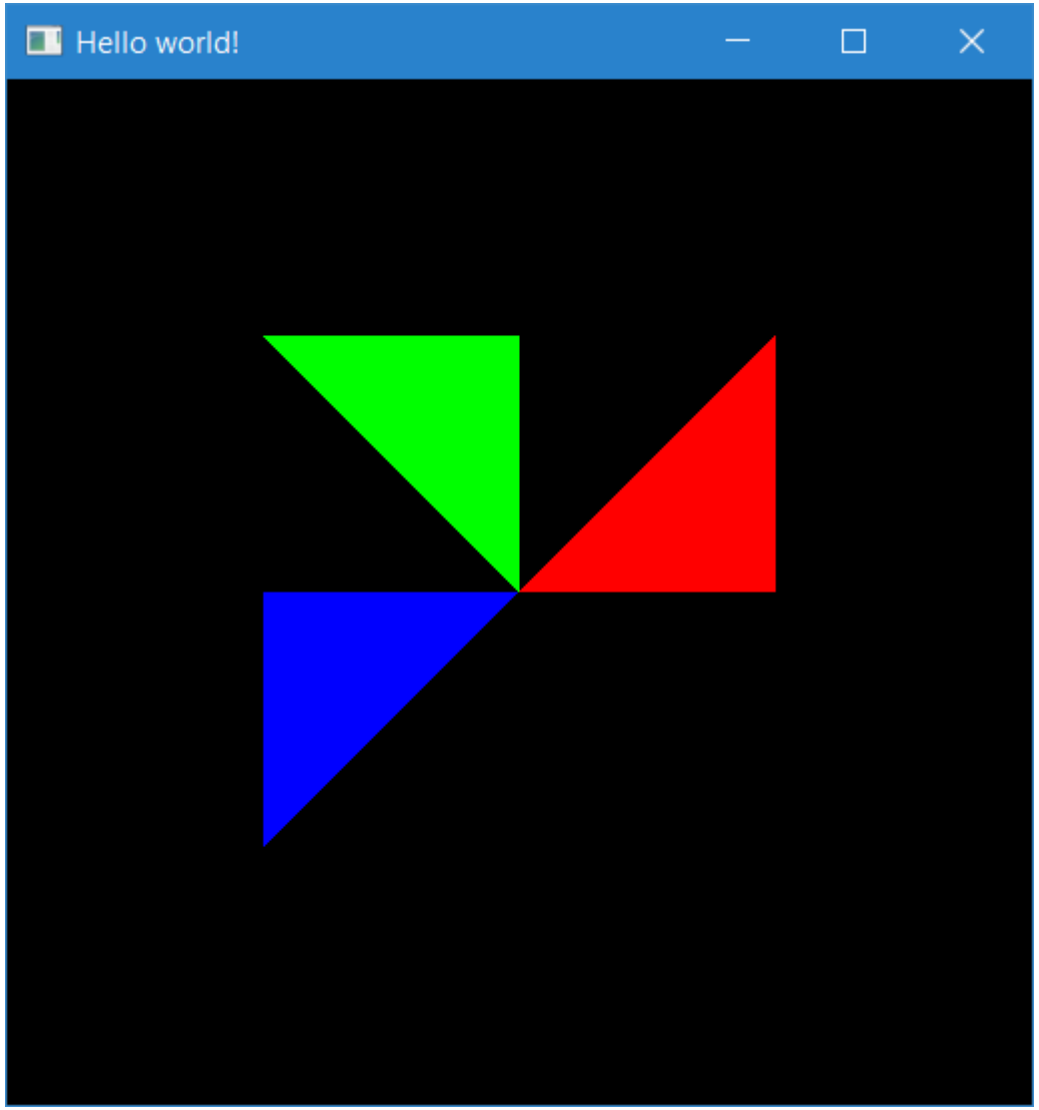
```
//Draw 3 color points
        glBegin(GL_POINTS)
        glColor3f(0.0, 0.0, 1.0)
        glVertex3f(0.5,0.5,0)
        glColor3f(1.0, 0.0, 1.0)
        glVertex3f(0.2,0.3,0)
        glColor3f(0.0, 1.0, 1.0)
        glVertex3f(0.0,0.1,0)
        glEnd()
```



Not: do NOT turn on light, when you are drawing the specific color.

Draw three triangles

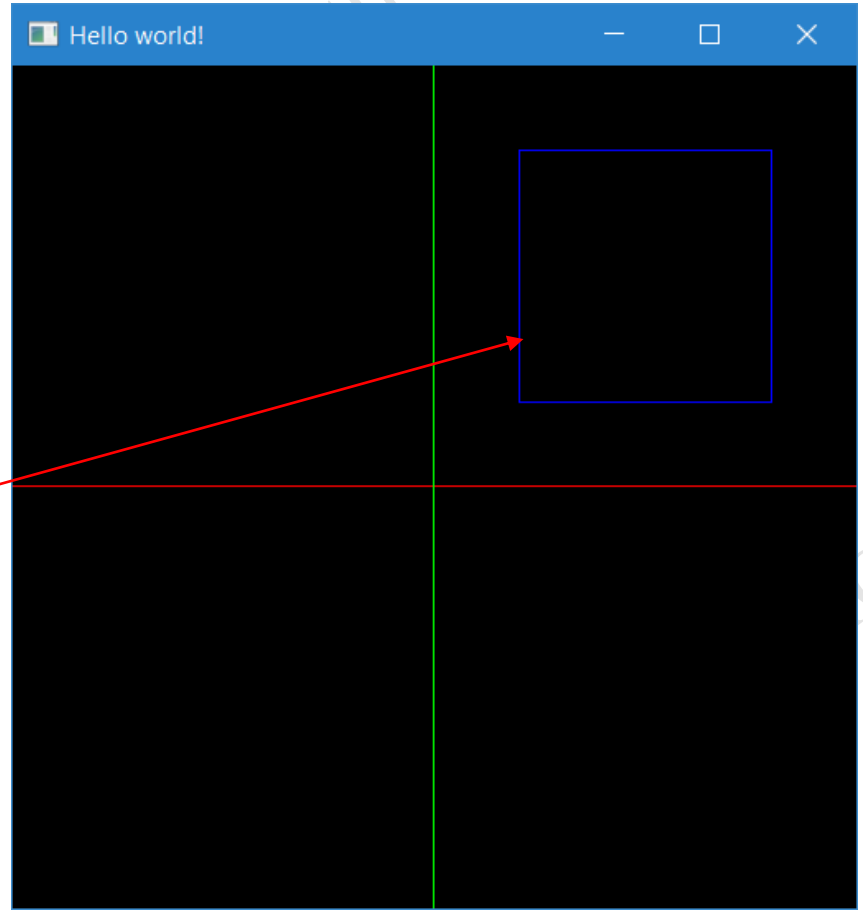
```
def display():
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
    glPushMatrix()
    glBegin(GL_TRIANGLES)
    glColor3f(1,0,0)
    glNormal3f(0,0,1)
    glVertex3f(0,0,0)
    glVertex3f(0.5,0,0)
    glVertex3f(0.5,0.5,0)
    glColor3f(0,1,0)
    glNormal3f(0,0,1)
    glVertex3f(0,0,0)
    glVertex3f(0,0.5,0)
    glVertex3f(-0.5,0.5,0)
    glColor3f(0,0,1);
    glNormal3f(0,0,1)
    glVertex3f(0,0,0)
    glVertex3f(-0.5,0,0)
    glVertex3f(-0.5,-0.5,0)
    glEnd()
    glPopMatrix()
    glutSwapBuffers()
```



NOTE: Do NOT turn ON the light, when you are drawing the “specific color”.

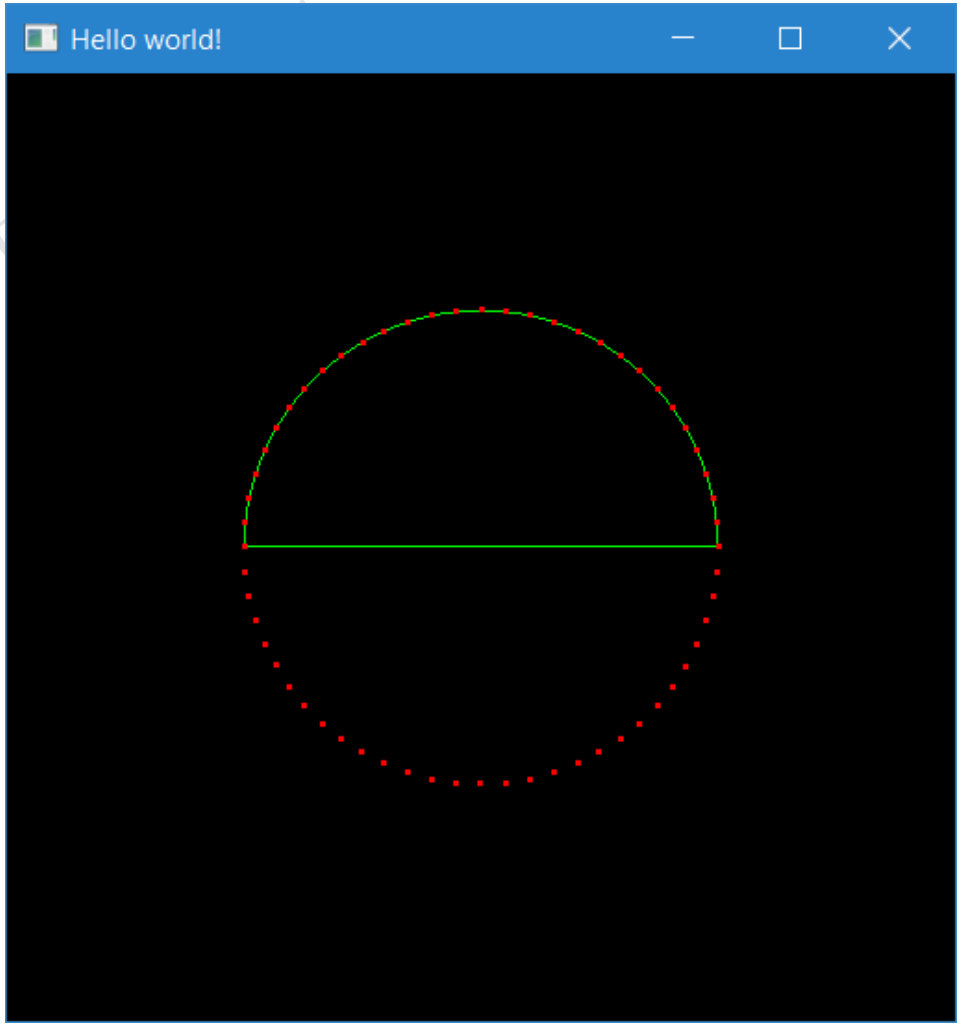
Draw lines

```
def display():
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
    glPushMatrix()
    glBegin(GL_LINES)
    glColor3f(1,0,0)
    glVertex3f(-1,0,0)
    glVertex3f(1,0,0)
    glColor3f(0,1,0)
    glVertex3f(0,-1,0)
    glVertex3f(0,1,0)
    glEnd()
    glBegin(GL_LINE_LOOP)
    glColor3f(0,0,1)
    glVertex3f(0.2,0.2,0)
    glVertex3f(0.8,0.2,0)
    glVertex3f(0.8,0.8,0)
    glVertex3f(0.2,0.8,0)
    glEnd()
    glPopMatrix()
    glutSwapBuffers()
```



Draw something systematically

```
def drawMyFig():
    glColor3f(0,1,0)
    glBegin(GL_LINE_LOOP)
    for i in range(31):
        glVertex3f(0.5*cos(i*6*3.1415/180.0),0.5*sin(i*6*3.1415/180.0),0)
    glEnd()
    glPointSize(3)
    glColor3f(1,0,0)
    glBegin(GL_POINTS)
    for i in range(60):
        glVertex3f(0.5*cos(i*6*3.1415/180.0),0.5*sin(i*6*3.1415/180.0),0)
    glEnd()


def display():
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)
    glPushMatrix()
    drawMyFig()
    glPopMatrix()
    glutSwapBuffers()
```



Practice for dynamic scenes

- glEnable(……)
- glutPostRedisplay();

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

Find us on: facebook