```
# coding: utf-8
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *
import sys

###############################################################################
# OpenGL funcs
###############################################################################
def initialize():
    glClearColor(0.0, 0.0, 0.0, 0.0)
    glClearDepth(1.0)
    glDepthFunc(GL_LESS)
    glEnable(GL_DEPTH_TEST)

def resize(Width, Height):
    # viewport
    if Height == 0:
        Height = 1
    glViewport(0, 0, Width, Height)
    # projection
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluPerspective(45.0, float(Width)/float(Height), 0.1, 100.0)

yaw=0
pitch=0
def draw():
    global yaw, pitch
    # clear
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

    # view
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
    yaw+=0.39
    pitch+=0.27
    glTranslatef(0.0, 0.0, -2.0)
    glRotatef(yaw, 0, 1, 0)
    glRotatef(pitch, 1, 0, 0)

    # cube
    #draw_cube0()
    #draw_cube1()
    #draw_cube2()
    draw_cube3()

    glFlush()

###############################################################################
# Shader
###############################################################################
# Checks for GL posted errors after appropriate calls
def printOpenGLError():
    err = glGetError()
    if (err != GL_NO_ERROR):
        print('GLERROR: ', gluErrorString(err))
        #sys.exit()

class Shader(object):

    def initShader(self, vertex_shader_source, fragment_shader_source):
        # create program
        self.program=glCreateProgram()
        print('create program')
        printOpenGLError()
```

```
        # vertex shader
        print('compile vertex shader...')
        self.vs = glCreateShader(GL_VERTEX_SHADER)
        glShaderSource(self.vs, [vertex_shader_source])
        glCompileShader(self.vs)
        glAttachShader(self.program, self.vs)
        printOpenGLError()

        # fragment shader
        print('compile fragment shader...')
        self.fs = glCreateShader(GL_FRAGMENT_SHADER)
        glShaderSource(self.fs, [fragment_shader_source])
        glCompileShader(self.fs)
        glAttachShader(self.program, self.fs)
        printOpenGLError()

        print('link...')
        glLinkProgram(self.program)
        printOpenGLError()

    def begin(self):
        if glUseProgram(self.program):
            printOpenGLError()

    def end(self):
        glUseProgram(0)


################################################################################
# vertices
################################################################################
s=0.5
vertices=[
        -s, -s, -s,
         s, -s, -s,
         s,  s, -s,
        -s,  s, -s,
        -s, -s,  s,
         s, -s,  s,
         s,  s,  s,
        -s,  s,  s,
        ]
colors=[
        0, 0, 0,
        1, 0, 0,
        0, 1, 0,
        0, 0, 1,
        0, 1, 1,
        1, 0, 1,
        1, 1, 1,
        1, 1, 0,
        ]
indices=[
        0, 1, 2, 2, 3, 0,
        0, 4, 5, 5, 1, 0,
        1, 5, 6, 6, 2, 1,
        2, 6, 7, 7, 3, 2,
        3, 7, 4, 4, 0, 3,
        4, 7, 6, 6, 5, 4,
        ]

def draw_cube0():
    glBegin(GL_TRIANGLES)
    for i in range(0, len(indices), 3):
        index=indices[i]*3
```

```
        glColor3f(*colors[index:index+3])
        glVertex3f(*vertices[index:index+3])

        index=indices[i+1]*3
        glColor3f(*colors[index:index+3])
        glVertex3f(*vertices[index:index+3])

        index=indices[i+2]*3
        glColor3f(*colors[index:index+3])
        glVertex3f(*vertices[index:index+3])
    glEnd()

def draw_cube1():
    glEnableClientState(GL_VERTEX_ARRAY);
    glEnableClientState(GL_COLOR_ARRAY);
    glVertexPointer(3, GL_FLOAT, 0, vertices);
    glColorPointer(3, GL_FLOAT, 0, colors)
    glDrawElements(GL_TRIANGLES, len(indices), GL_UNSIGNED_INT, indices);
    glDisableClientState(GL_COLOR_ARRAY)
    glDisableClientState(GL_VERTEX_ARRAY);

buffers=None
def create_vbo():
    buffers = glGenBuffers(3)
    glBindBuffer(GL_ARRAY_BUFFER, buffers[0])
    glBufferData(GL_ARRAY_BUFFER,
            len(vertices)*4,  # byte size
            (ctypes.c_float*len(vertices))(*vertices),
            GL_STATIC_DRAW)
    glBindBuffer(GL_ARRAY_BUFFER, buffers[1])
    glBufferData(GL_ARRAY_BUFFER,
            len(colors)*4, # byte size
            (ctypes.c_float*len(colors))(*colors),
            GL_STATIC_DRAW)
    glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, buffers[2])
    glBufferData(GL_ELEMENT_ARRAY_BUFFER,
            len(indices)*4, # byte size
            (ctypes.c_uint*len(indices))(*indices),
            GL_STATIC_DRAW)
    return buffers

def draw_vbo():
    glEnableClientState(GL_VERTEX_ARRAY);
    glEnableClientState(GL_COLOR_ARRAY);
    glBindBuffer(GL_ARRAY_BUFFER, buffers[0]);
    glVertexPointer(3, GL_FLOAT, 0, None);
    glBindBuffer(GL_ARRAY_BUFFER, buffers[1]);
    glColorPointer(3, GL_FLOAT, 0, None);
    glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, buffers[2]);
    glDrawElements(GL_TRIANGLES, len(indices), GL_UNSIGNED_INT, None);
    glDisableClientState(GL_COLOR_ARRAY)
    glDisableClientState(GL_VERTEX_ARRAY);

def draw_cube2():
    global buffers
    if buffers==None:
        buffers=create_vbo()
    draw_vbo()

shader=None
def draw_cube3():
    global shader, buffers
    if shader==None:
        shader=Shader()
        shader.initShader('''
void main()
```

```
{
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
    gl_FrontColor = gl_Color;
}
        ''',
        '''
void main()
{
    gl_FragColor = gl_Color;
}
        ''')
        buffers=create_vbo()

    shader.begin()
    draw_vbo()
    shader.end()

################################################################################
def reshape_func(w, h):
    resize(w, h == 0 and 1 or h)

def disp_func():
    draw()
    glutSwapBuffers()

if __name__=="__main__":
    glutInit(sys.argv)
    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH)
    glutInitWindowSize(256, 256)
    glutCreateWindow(b"vbo")
    glutDisplayFunc(disp_func)
    glutIdleFunc(disp_func)
    glutReshapeFunc(reshape_func)

    initialize()

    glutMainLoop()
```