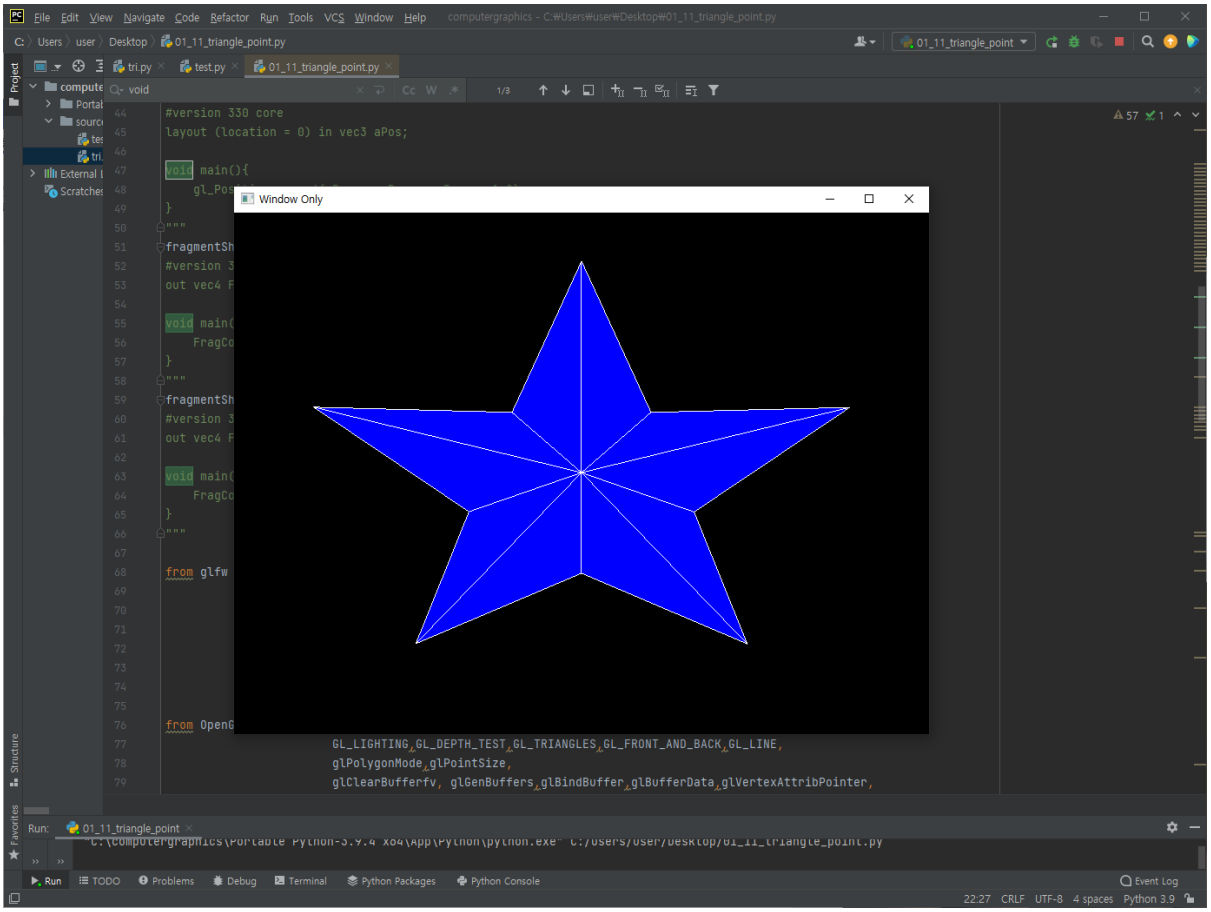
컴퓨터그래픽스 실습1

2016133 이유진

```
import math, numpy
import OpenGL


SCR_WIDTH = 800
SCR_HEIGHT = 600


vertexData = numpy.array(
          [-0.200, 0.234, -0.089,
    -0.771, 0.254, -0.089,
    -0.000, 0.003, -0.089,
    -0.771, 0.254, -0.089,
    -0.324, -0.147, -0.089,
    -0.000, 0.003, -0.089,
    -0.324, -0.147, -0.089,
    -0.477, -0.653, -0.089,
    -0.000, 0.003, -0.089,
    -0.477, -0.653, -0.089,
    -0.000, -0.383, -0.089,
    -0.000, 0.003, -0.089,
    -0.000, -0.383, -0.089,
    0.477, -0.654, -0.089,
    -0.000, 0.003, -0.089,
    0.477, -0.654, -0.089,
    0.324, -0.148, -0.089,
    -0.000, 0.003, -0.089,
    0.324, -0.148, -0.089,
    0.772, 0.253, -0.089,
    -0.000, 0.003, -0.089,
    0.772, 0.253, -0.089,
    0.200, 0.234, -0.089,
    -0.000, 0.003, -0.089,
    0.200, 0.234, -0.089,
    0.000, 0.814, -0.089,
    -0.000, 0.003, -0.089,
    0.000, 0.814, -0.089,
    -0.200, 0.234, -0.089,
    -0.000, 0.003, -0.089
           ], numpy.float32)

vertexShaderSource = """
#version 330 core
layout (location = 0) in vec3 aPos;

void main(){
   gl_Position = vec4(aPos.x, aPos.y, aPos.z, 1.0);
}
"""
fragmentShaderSource = """
#version 330 core
out vec4 FragColor;

void main(){
   FragColor = vec4(0.0f, 0.0f, 1.0f, 1.0f);
}
"""
fragmentShaderSource2 = """
#version 330 core
out vec4 FragColor;

void main(){
   FragColor = vec4(1.0f, 1.0f, 1.0f, 1.0f);
```

```
}
"""

from glfw import (window_hint, init, create_window, terminate,
                  make_context_current, swap_buffers, poll_events,
                  window_should_close,
                  CONTEXT_VERSION_MAJOR,
                  CONTEXT_VERSION_MINOR, OPENGL_FORWARD_COMPAT,
                  OPENGL_PROFILE, OPENGL_CORE_PROFILE)


from OpenGL.GL import ( GL_TRUE, GL_COLOR, GL_POINTS,
GL_LINE_LOOP,GL_ARRAY_BUFFER,GL_STATIC_DRAW,
                        GL_LIGHTING,GL_DEPTH_TEST,GL_TRIANGLES,GL_FRONT_AND_BACK,GL_LINE,
                        glPolygonMode,glPointSize,
                        glClearBufferfv,
glGenBuffers,glBindBuffer,glBufferData,glVertexAttribPointer,
                        glDrawArrays, glLinkProgram,glEnableVertexAttribArray,GL_FLOAT,
GL_FALSE,glDeleteProgram,
                        glGenVertexArrays, glBindVertexArray,glDeleteBuffers ,
glDeleteVertexArrays,
                        glLineWidth,glDisable, glColor3f)

from OpenGL.GL.shaders import (GL_VERTEX_SHADER, GL_FRAGMENT_SHADER,
                           compileShader, glCreateProgram,
                           glAttachShader, glUseProgram, glGetUniformLocation,
glUniform1f)


def main():
    # Initialize the library
    if not init():
        return

    window_hint(CONTEXT_VERSION_MAJOR, 3)
    window_hint(CONTEXT_VERSION_MINOR, 3)
    window_hint(OPENGL_FORWARD_COMPAT, GL_TRUE)
    window_hint(OPENGL_PROFILE, OPENGL_CORE_PROFILE)

    # program = glutils.loadShaders(vs_source, fs_source)
    # Create a windowed mode window and its OpenGL context
    window = create_window(SCR_WIDTH, SCR_HEIGHT, "Window Only", None, None)
    if not window:
        terminate()
        return

    # Make the window's context current
    make_context_current(window)

    shaderV = compileShader([vertexShaderSource], GL_VERTEX_SHADER)
    shaderF = compileShader([fragmentShaderSource], GL_FRAGMENT_SHADER)
    program = glCreateProgram()


    glAttachShader(program, shaderV)
    glAttachShader(program, shaderF)
    glLinkProgram(program)

    # ----------- NEW LINE
    shaderF2 = compileShader([fragmentShaderSource2], GL_FRAGMENT_SHADER)
    program2 = glCreateProgram()

    glAttachShader(program2, shaderV)
    glAttachShader(program2, shaderF2)
    glLinkProgram(program2)
    # ----------- NEW END
```

```
    vao = glGenVertexArrays(1)
    glBindVertexArray(vao)
    vertexBuffer = glGenBuffers(1)
    glBindBuffer(GL_ARRAY_BUFFER, vertexBuffer)

    glBufferData(GL_ARRAY_BUFFER, 4 * len(vertexData), vertexData,
                 GL_STATIC_DRAW)
    # enable vertex array
    glEnableVertexAttribArray(0)
    # set buffer data pointer
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 0, None)
    # unbind VAO
    glBindVertexArray(0)


    # Loop until the user closes the window
    while not window_should_close(window):
        # Render here, e.g. using pyOpenGL

        glClearBufferfv(GL_COLOR, 0, (0.0, 0.0, 0.0, 1.0))

        glUseProgram(program)
        glBindVertexArray(vao)
        glDisable(GL_DEPTH_TEST)

        #
        glPointSize(20.0)
        glDrawArrays(GL_TRIANGLES, 0, 30)

        glUseProgram(program2)
        glDrawArrays(GL_LINE_LOOP, 0, 30)
        # Swap front and back buffers
        swap_buffers(window)

        # Poll for and process events
        poll_events()

    glBindVertexArray(0)
    glDeleteBuffers(1, [vertexBuffer])
    glDeleteProgram(program)
    glDeleteVertexArrays(1, [vao])
    terminate()

if __name__ == "__main__":
    main()
```



별이 찌그러져서 윈도우 크기를 800,800으로 변경한 실행 결과도 첨부합니다.