컴퓨터 그래픽스 실습 2

컴퓨터과학과 2016133 이유진

```
import math, numpy
import OpenGL, ctypes


SCR_WIDTH = 600
SCR_HEIGHT = 600

vertexShaderSource = """
#version 330 core
layout (location = 0) in vec3 aPos;
layout (location = 1) in vec3 aColor;

out vec3 outColor;

uniform vec3 scale;
uniform vec3 trans;

void main(){
    vec3 nposition = aPos * scale + trans;
    gl_Position = vec4(nposition, 1.0);
    outColor = aColor;
}
"""
fragmentShaderSource = """
#version 330 core
out vec4 FragColor;
in vec3 outColor;

uniform int incase;

void main(){
    switch (incase)
    {
        case 0:
            FragColor = vec4(outColor, 1.0f);
            break;
        case 1:
            FragColor = vec4(0.0f, 1.0f, 1.0f, 1.0f);
            break;
    }
}
"""



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_LINE_LOOP,GL_ARRAY_BUFFER,GL_STATIC_DRAW,
                        GL_LIGHTING,GL_DEPTH_TEST,GL_TRIANGLES,GL_FRONT_AND_BACK,GL_LINE,
                        glPolygonMode,glUniform1i,glUniform3fv,
                        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)

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

    vertexData = numpy.array([], numpy.float32)
    npos = []
    last_pos = []
    triangle = []
    for x in range(40):
        posx = math.sin(x * math.pi / 40.0 * 2.0) * 0.8
        posy = math.cos(x * math.pi / 40.0 * 2.0) * 0.8
        npos = [posx, posy, 0, 0.0, 1.0 / 40 * x, 1.0 - 1.0 / 40 * x]
        if last_pos:
            triangle = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0] + last_pos + npos
            vertexData = numpy.append(vertexData, numpy.array(triangle, numpy.float32))

        last_pos = npos

    triangle = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0] + last_pos + list(vertexData[6 * 1:6 * 1
+ 3]) + [0.0, 1.0, 0.0]
    vertexData = numpy.append(vertexData, numpy.array(triangle, numpy.float32))

    glBufferData(GL_ARRAY_BUFFER, 4 * len(vertexData), vertexData,
                 GL_STATIC_DRAW)
    # enable vertex array
    glEnableVertexAttribArray(0)


    ## position of the attrib array, POSITION
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 6 * 4, None)
    glEnableVertexAttribArray(0)
```

```
    ## position of the attrib array, COLOR
    glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, 6 * 4, ctypes.c_void_p(3 * 4))
    glEnableVertexAttribArray(1)

    # 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)

        caseM = glGetUniformLocation(program, "incase")
        glUniform1i(caseM, 0)

        scaleM = glGetUniformLocation(program, "scale")
        transM = glGetUniformLocation(program, "trans")

        for x in range(16):
            scale = x * 0.03
            posx = math.sin(math.pi * 2.0 / 16.0 * x) * 0.5
            posy = math.cos(math.pi * 2.0 / 16.0 * x) * 0.5
            glUniform3fv(scaleM, 1, numpy.array([scale, scale, scale], numpy.float32))
            glUniform3fv(transM, 1, numpy.array([posx, posy, 0.0], numpy.float32))

            glDrawArrays(GL_TRIANGLES, 0, len(vertexData))

        # 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()
```

실행결과