컴퓨터그래픽스 실습3

2016133 이유진

```
import math, numpy, ctypes
import OpenGL
from pyrr import matrix44, Vector3
from ObjLoader import *

SCR_WIDTH = 800
SCR_HEIGHT = 600

vertexShaderSource = """
#version 330 core

uniform mat4 transform;

uniform mat4 view;
uniform mat4 model;
uniform mat4 projection;

layout (location=0) in vec4 vPosition;
layout (location=1) in vec3 vertNormal;

out vec3 fragNormal;

void
main()
{
    fragNormal = vertNormal;//(vec4(vertNormal, 0.0f)).xyz;

   gl_Position = projection * view * model * transform * vPosition;
}
"""
#vertNomal : 면의 방향?
#projection 은 othgraphic 인지 prespective 인지
#view 는 카메라위치
#model 은 object 를 어떻게 움직일 것이냐
#transform 은 object 를 어떻게 이동할 것이냐


fragmentShaderSource = """
#version 330 core

in vec3 fragNormal;
out vec4 FragColor;

void main(){

    vec3 ambientLightIntensity = vec3(0.3f, 0.2f, 0.4f);
    vec3 sunLightIntensity = vec3(0.9f, 0.9f, 0.9f);
    vec3 sunLightDirection = normalize(vec3(-2.0f, 2.0f, 0.0f));

    vec3 lightIntensity = ambientLightIntensity + sunLightIntensity *
max(dot(fragNormal, sunLightDirection), 0.0f);

    FragColor = vec4(1.0f, 1.0f, 1.0f, 1.0f) * vec4(lightIntensity, 1.0);
}
"""


from glfw import (window_hint, init, create_window, terminate,
                  make_context_current, swap_buffers, poll_events,
                  window_should_close,get_time,
                  CONTEXT_VERSION_MAJOR,
```

```
               CONTEXT_VERSION_MINOR, OPENGL_FORWARD_COMPAT,
               OPENGL_PROFILE, OPENGL_CORE_PROFILE)


from OpenGL.GL import *

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

    frame = 0

    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()

    obj = ObjLoader()
    obj.load_model("res/bunny.obj")

    texture_offset = len(obj.vertex_index) * 12
    normal_offset = (texture_offset + len(obj.texture_index) * 8)

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

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

    #glBufferData(GL_ARRAY_BUFFER, 4 * len(vertexData), vertexData,
    #             GL_STATIC_DRAW)

    glBufferData(GL_ARRAY_BUFFER, obj.model.itemsize * len(obj.model), obj.model,
GL_STATIC_DRAW)
    #glBufferData(GL_ARRAY_BUFFER, len(obj.vertex_index), obj.vertex_index,
GL_STATIC_DRAW)
    # enable vertex array
    # set buffer data point
    glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, obj.model.itemsize * 3,
ctypes.c_void_p(0))
    glEnableVertexAttribArray(0)

    # normals
```

```
    glVertexAttribPointer(1, 3, GL_FLOAT, GL_FALSE, obj.model.itemsize * 3,
ctypes.c_void_p(normal_offset))
    glEnableVertexAttribArray(1)

    # unbind VAO
    glBindVertexArray(0)

    glClearColor(0.0, 0.0, 0.0, 1.0)
    glEnable(GL_DEPTH_TEST)

    #glPolygonMode(GL_FRONT_AND_BACK, GL_LINE)

    projection = matrix44.create_perspective_projection_matrix(45.0, SCR_WIDTH /
SCR_HEIGHT, 0.1, 100.0)
    view = matrix44.create_from_translation(Vector3([0.0, 0.0, -3.0]))
    model = matrix44.create_from_translation(Vector3([0.0, 0.0, 0.0]))

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

        poll_events()

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

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


        locMatP = glGetUniformLocation(program, "projection")
        locMatV = glGetUniformLocation(program, "view")
        locMatM = glGetUniformLocation(program, "model")
        glUniformMatrix4fv(locMatP, 1, GL_FALSE, projection)
        glUniformMatrix4fv(locMatV, 1, GL_FALSE, view)
        glUniformMatrix4fv(locMatM, 1, GL_FALSE, model)

        transformLoc = glGetUniformLocation(program, "transform")

# 제일 왼쪽, rotate
        rot_y = matrix44.create_from_y_rotation(0.8 * get_time())
        scale = matrix44.create_from_scale(Vector3([0.45, 0.45, 0.45]))
        matrix = matrix44.multiply(scale, rot_y)
        trans = matrix44.create_from_translation(Vector3([-1.0, 0.0, 0.0]))
        matrix = matrix44.multiply(matrix, trans)

        glUniformMatrix4fv(transformLoc, 1, GL_FALSE, matrix)
        glDrawArrays(GL_TRIANGLES, 0, len(obj.vertex_index))

# 가운데, scale
        sv = math.cos(get_time() * 5.0) * 0.05 + 0.4 #곱하기 5 는 빠르게
        scale = matrix44.create_from_scale(Vector3([sv, sv, sv]))
        trans = matrix44.create_from_translation(Vector3([0, 0, 0]))
        matrix = matrix44.multiply(scale, trans)
        glUniformMatrix4fv(transformLoc, 1, GL_FALSE, matrix)
        glDrawArrays(GL_TRIANGLES, 0, len(obj.vertex_index))

# 제일 오른쪽, trans
        tv = math.cos(get_time() * 5.0) * 0.1
        scale = matrix44.create_from_scale(Vector3([0.45, 0.45, 0.45]))
        trans = matrix44.create_from_translation(Vector3([1, tv, 0]))
        matrix = matrix44.multiply(scale, trans)
        glUniformMatrix4fv(transformLoc, 1, GL_FALSE, matrix)
        glDrawArrays(GL_TRIANGLES, 0, len(obj.vertex_index))
```

```
        # Swap front and back buffers
        swap_buffers(window)

        # Poll for and process events
        #poll_events()

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

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

실행화면