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

