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?  
#projection은 othgraphic인지 prespective인지  
#view는 카메라위치  
#model은 object를 어떻게 움직일 것이냐  
#transform은 object를 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 #곱하기5는 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

텍스트, 포유류, 집고양이이(가) 표시된 사진

자동 생성된 설명