BURSA ULUDAĞ ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

2023-2024 EĞİTİM ÖĞRETİM YILI BAHAR DÖNEMİ

BİLGİSAYAR GRAFİKLERİ RAPORU

MURAT BERK YETİŞTİRİR

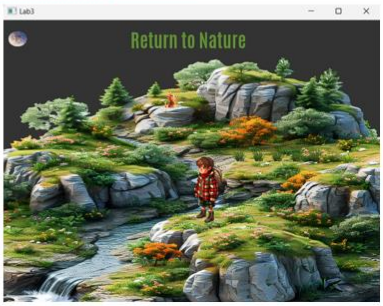
032290008

032290008@ogr.uludag.edu.tr

Soru:

Lab3

- Verilen png görüntülerinden oluşan 2-B bir doğa sahnesi tarayınız.
  - Farklı görüntüler için farklı konum ve ölçeklerle taramayı gerçekleştiriniz.
  - Programı esneterek değişen sayıda görüntüyü aynı gölgelendirici program içerisinde farklı vertex dizi objeleriyle çizdiriniz.
  - Renge alfa parametresini de ekleyiniz ve renk harmanlamayı aktifleştiriniz.
  - filesystem, shader_s ve stb_image kütüphanelerini ekleyiniz.
- Aynı sahneye metin taramak için diğer bir gölgelendirici kodunu geliştiriniz.
  - Sahne taramadan sonra metin taramayı aktifleştiriniz.
  - Metin için uygun bir konum ve renk ayarını gerçekleştiriniz.
  - freetype, glm kütüphanelerinden yararlanınız.



Cevap Kodu:

```
#include<iostream>
#include "glad.h"
#include<GLFW/glfw3.h>
#include"stb_image.h"

#include"Texture.h"
#include"shaderClass.h"
#include"VAO.h"
#include"VBO.h"
#include"EBO.h"

// Dag
GLfloat vertices[] =
{ //     COORDINATES     /        COLORS      /   TexCoord  //
    -0.7f, -0.7f, 0.0f,     1.0f, 0.0f, 0.0f,   0.0f, 0.0f, // Lower left corner
    -0.7f,  0.7f, 0.0f,     0.0f, 1.0f, 0.0f,   0.0f, 1.0f, // Upper left corner
     0.7f,  0.7f, 0.0f,     0.0f, 0.0f, 1.0f,   1.0f, 1.0f, // Upper right corner
     0.7f, -0.7f, 0.0f,     1.0f, 1.0f, 1.0f,   1.0f, 0.0f  // Lower right corner
};

// Indices for vertices order
GLuint indices[] =
{
    0, 2, 1, // Upper triangle
    0, 3, 2 // Lower triangle
};

// Cocuk
GLfloat vertices1[] =
{ //     COORDINATES     /        COLORS      /   TexCoord  //
    -0.4f, -0.2f, 0.0f,     1.0f, 0.0f, 0.0f,   0.0f, 0.0f, // Lower left corner
    -0.4f,  0.3f, 0.0f,     0.0f, 1.0f, 0.0f,   0.0f, 1.0f, // Upper left corner
```

```
     0.0f,  0.3f, 0.0f,     0.0f, 0.0f, 1.0f,   1.0f, 1.0f, // Upper right corner
     0.0f, -0.2f, 0.0f,     1.0f, 1.0f, 1.0f,   1.0f, 0.0f  // Lower right corner
};

// Indices for vertices order
GLuint indices1[] =
{
    0, 2, 1, // Upper triangle
    0, 3, 2 // Lower triangle
};

// Kedi
GLfloat vertices2[] =
{ //     COORDINATES     /        COLORS      /   TexCoord  //
     0.3f, -0.3f, 0.0f,     1.0f, 0.0f, 0.0f,   0.0f, 0.0f, // Lower left corner
     0.3f,  0.3f, 0.0f,     0.0f, 1.0f, 0.0f,   0.0f, 1.0f, // Upper left corner
    -0.1f,  0.3f, 0.0f,     0.0f, 0.0f, 1.0f,   1.0f, 1.0f, // Upper right corner
    -0.1f, -0.3f, 0.0f,     1.0f, 1.0f, 1.0f,   1.0f, 0.0f  // Lower right corner
};

// Indices for vertices order
GLuint indices2[] =
{
    0, 2, 1, // Upper triangle
    0, 3, 2 // Lower triangle
};

// Ay
GLfloat vertices3[] =
{ //     COORDINATES     /        COLORS      /   TexCoord  //
    -0.6f,  0.6f, 0.0f,     1.0f, 0.0f, 0.0f,   0.0f, 0.0f, // Lower left corner
    -0.6f,  0.5f, 0.0f,     0.0f, 1.0f, 0.0f,   0.0f, 1.0f, // Upper left corner
    -0.5f,  0.5f, 0.0f,     0.0f, 0.0f, 1.0f,   1.0f, 1.0f, // Upper right corner
    -0.5f,  0.6f, 0.0f,     1.0f, 1.0f, 1.0f,   1.0f, 0.0f  // Lower right corner
};

// Indices for vertices order
GLuint indices3[] =
{
    0, 2, 1, // Upper triangle
    0, 3, 2 // Lower triangle
};



int main()
{
    // Initialize GLFW
    glfwInit();

    // Tell GLFW what version of OpenGL we are using
    // In this case we are using OpenGL 3.3
    glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
    glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
```

```
    // Tell GLFW we are using the CORE profile
    // So that means we only have the modern functions
    glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);

    // Create a GLFWwindow object of 800 by 800 pixels, naming it "YoutubeOpenGL"
    GLFWwindow* window = glfwCreateWindow(800, 800, "YoutubeOpenGL", NULL, NULL);
    // Error check if the window fails to create
    if (window == NULL)
    {
        std::cout << "Failed to create GLFW window" << std::endl;
        glfwTerminate();
        return -1;
    }
    // Introduce the window into the current context
    glfwMakeContextCurrent(window);

    //Load GLAD so it configures OpenGL
    gladLoadGL();
    // Specify the viewport of OpenGL in the Window
    // In this case the viewport goes from x = 0, y = 0, to x = 800, y = 800
    glViewport(0, 0, 800, 800);



    // Generates Shader object using shaders default.vert and default.frag
    Shader shaderProgram("default.vert", "default.frag");



    // Generates Vertex Array Object and binds it
    VAO VAO1;
    VAO1.Bind();

    // Generates Vertex Buffer Object and links it to vertices
    VBO VBO1(vertices, sizeof(vertices));
    // Generates Element Buffer Object and links it to indices
    EBO EBO1(indices, sizeof(indices));

    // Links VBO attributes such as coordinates and colors to VAO
    VAO1.LinkAttrib(VBO1, 0, 3, GL_FLOAT, 8 * sizeof(float), (void*)0);
    VAO1.LinkAttrib(VBO1, 1, 3, GL_FLOAT, 8 * sizeof(float), (void*)(3 *
sizeof(float)));
    VAO1.LinkAttrib(VBO1, 2, 2, GL_FLOAT, 8 * sizeof(float), (void*)(6 *
sizeof(float)));
    // Unbind all to prevent accidentally modifying them
    VAO1.Unbind();
    VBO1.Unbind();
    EBO1.Unbind();

    // Gets ID of uniform called "scale"
    GLuint uniID = glGetUniformLocation(shaderProgram.ID, "scale");

    Texture mountain("mountain.png", GL_TEXTURE_2D, GL_TEXTURE0, GL_RGBA,
GL_UNSIGNED_BYTE);
```

```
    mountain.texUnit(shaderProgram, "tex0", 0);



    VAO VAO2;
    VAO2.Bind();

    // Generates Vertex Buffer Object and links it to vertices
    VBO VBO2(vertices1, sizeof(vertices1));
    // Generates Element Buffer Object and links it to indices
    EBO EBO2(indices1, sizeof(indices1));

    // Links VBO attributes such as coordinates and colors to VAO
    VAO2.LinkAttrib(VBO2, 0, 3, GL_FLOAT, 8 * sizeof(float), (void*)0);
    VAO2.LinkAttrib(VBO2, 1, 3, GL_FLOAT, 8 * sizeof(float), (void*)(3 *
sizeof(float)));
    VAO2.LinkAttrib(VBO2, 2, 2, GL_FLOAT, 8 * sizeof(float), (void*)(6 *
sizeof(float)));
    // Unbind all to prevent accidentally modifying them
    VAO2.Unbind();
    VBO2.Unbind();
    EBO2.Unbind();

    // Gets ID of uniform called "scale"
    GLuint uniID2 = glGetUniformLocation(shaderProgram.ID, "scale");

    Texture man("man.png", GL_TEXTURE_2D, GL_TEXTURE0, GL_RGBA, GL_UNSIGNED_BYTE);
    man.texUnit(shaderProgram, "tex0", 0);



    VAO VAO3;
    VAO3.Bind();

    // Generates Vertex Buffer Object and links it to vertices
    VBO VBO3(vertices2, sizeof(vertices2));
    // Generates Element Buffer Object and links it to indices
    EBO EBO3(indices2, sizeof(indices2));

    // Links VBO attributes such as coordinates and colors to VAO
    VAO3.LinkAttrib(VBO3, 0, 3, GL_FLOAT, 8 * sizeof(float), (void*)0);
    VAO3.LinkAttrib(VBO3, 1, 3, GL_FLOAT, 8 * sizeof(float), (void*)(3 *
sizeof(float)));
    VAO3.LinkAttrib(VBO3, 2, 2, GL_FLOAT, 8 * sizeof(float), (void*)(6 *
sizeof(float)));
    // Unbind all to prevent accidentally modifying them
    VAO3.Unbind();
    VBO3.Unbind();
    EBO3.Unbind();

    // Gets ID of uniform called "scale"
    GLuint uniID3 = glGetUniformLocation(shaderProgram.ID, "scale");

    Texture cat("cat.png", GL_TEXTURE_2D, GL_TEXTURE0, GL_RGBA, GL_UNSIGNED_BYTE);
```

```
    cat.texUnit(shaderProgram, "tex0", 0);



    VAO VAO4;
    VAO4.Bind();

    // Generates Vertex Buffer Object and links it to vertices
    VBO VBO4(vertices3, sizeof(vertices3));
    // Generates Element Buffer Object and links it to indices
    EBO EBO4(indices3, sizeof(indices3));

    // Links VBO attributes such as coordinates and colors to VAO
    VAO4.LinkAttrib(VBO4, 0, 3, GL_FLOAT, 8 * sizeof(float), (void*)0);
    VAO4.LinkAttrib(VBO4, 1, 3, GL_FLOAT, 8 * sizeof(float), (void*)(3 *
sizeof(float)));
    VAO4.LinkAttrib(VBO4, 2, 2, GL_FLOAT, 8 * sizeof(float), (void*)(6 *
sizeof(float)));
    // Unbind all to prevent accidentally modifying them
    VAO4.Unbind();
    VBO4.Unbind();
    EBO4.Unbind();

    // Gets ID of uniform called "scale"
    GLuint uniID4 = glGetUniformLocation(shaderProgram.ID, "scale");


    Texture moon("moon.png", GL_TEXTURE_2D, GL_TEXTURE0, GL_RGBA, GL_UNSIGNED_BYTE);
    moon.texUnit(shaderProgram, "tex0", 0);


    /*
    * I'm doing this relative path thing in order to centralize all the resources
into one folder and not
    * duplicate them between tutorial folders. You can just copy paste the resources
from the 'Resources'
    * folder and then give a relative path from this folder to whatever resource you
want to get to.
    * Also note that this requires C++17, so go to Project Properties, C/C++,
Language, and select C++17
    */

    // Texture




    // Original code from the tutorial
    /*Texture popCat("pop_cat.png", GL_TEXTURE_2D, GL_TEXTURE0, GL_RGBA,
GL_UNSIGNED_BYTE);
    popCat.texUnit(shaderProgram, "tex0", 0);*/

    glEnable(GL_BLEND);
```

```
    glBlendFunc(GL_ONE, GL_ONE_MINUS_SRC_ALPHA);


    // Main while loop
    while (!glfwWindowShouldClose(window))
    {
        // Specify the color of the background
        glClearColor(0.07f, 0.13f, 0.17f, 1.0f);
        // Clean the back buffer and assign the new color to it
        glClear(GL_COLOR_BUFFER_BIT);
        // Tell OpenGL which Shader Program we want to use
        shaderProgram.Activate();
        // Assigns a value to the uniform; NOTE: Must always be done after activating
the Shader Program

        glUniform1f(uniID, 0.5f);
        mountain.Bind();
        VAO1.Bind();
        glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, 0);



        shaderProgram.Activate();
        glUniform1f(uniID2, 0.5f);
        man.Bind();
        VAO2.Bind();
        glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, 0);


        shaderProgram.Activate();
        glUniform1f(uniID3, 0.5f);
        cat.Bind();
        VAO3.Bind();
        glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, 0);


        shaderProgram.Activate();
        glUniform1f(uniID4, 0.5f);
        moon.Bind();
        VAO4.Bind();
        // Draw primitives, number of indices, datatype of indices, index of indices
        glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, 0);
        // Swap the back buffer with the front buffer
        glfwSwapBuffers(window);
        // Take care of all GLFW events
        glfwPollEvents();
    }



    // Delete all the objects we've created
    VAO1.Delete();
    VBO1.Delete();
    EBO1.Delete();
```

```
    mountain.Delete();

    VAO2.Delete();
    VBO2.Delete();
    EBO2.Delete();
    man.Delete();

    VAO3.Delete();
    VBO3.Delete();
    EBO3.Delete();
    cat.Delete();

    VAO4.Delete();
    VBO4.Delete();
    EBO4.Delete();
    moon.Delete();

    shaderProgram.Delete();
    // Delete window before ending the program
    glfwDestroyWindow(window);
    // Terminate GLFW before ending the program
    glfwTerminate();
    return 0;
}
```

Cevap Ekran Çıktısı: