BURSA ULUDAĞ ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

2023-2024 EĞİTİM ÖĞRETİM YILI BAHAR DÖNEMİ

BİLGİSAYAR GRAFİKLERİ RAPORU

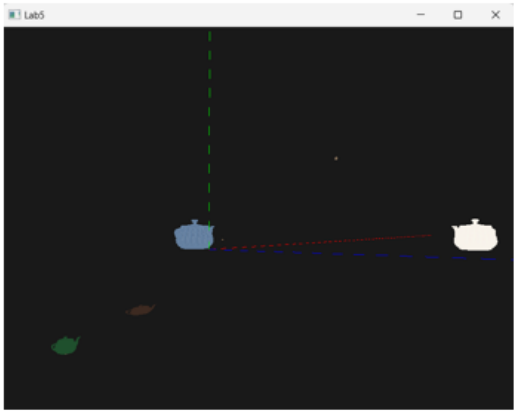
MURAT BERK YETİŞTİRİR

032290008

032290008@ogr.uludag.edu.tr

SORU:

- Çaydanlık modeline farklı doku ve geometrik dönüşümler uygulayarak 3-B bir sahne görünümünü görselleyiniz.
  - Dönüşümler için glm kütüphanesinden yararlanınız.
  - Programdaki modelleri okumada assimp kütüphanesinden yararlanınız.
  - 3-B sahnede gezinme için hazır klavye veya fare fonksiyonlarını ekleyiniz.
  - Bakış dönüşümleri için camera sınıfından yararlanınız.
  - model, camera, filesystem, shader ve stb_image kütüphanelerini ekleyiniz.
  - Sırasıyla uygulanacak dönüşümler için bu slaytın notlar kısmına bakınız.
  - Konumunu belirlemek istediğimiz sabit noktaları ölçeklenmiş gezegen modeli veya kaya modeli şeklinde çizdiriniz.
  - x-y-z eksenlerini R-G-B renklerinde ayrı bir shader uygulaması üzerinden çizdiriniz.
  - Kodunuzu ve ekran çıktısını içeren raporunuzu Teams'deki Lab5 ödevi altına yükleyiniz.



CEVAP KODU:

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

#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtc/type_ptr.hpp>

#include <learnopengl/filesystem.h>
#include <learnopengl/shader.h>
#include <learnopengl/camera.h>
#include <learnopengl/model.h>

#include <iostream>

void framebuffer_size_callback(GLFWwindow* window, int width, int height);
void mouse_callback(GLFWwindow* window, double xpos, double ypos);
void scroll_callback(GLFWwindow* window, double xoffset, double yoffset);
void processInput(GLFWwindow* window);

// settings
const unsigned int SCR_WIDTH = 800;
const unsigned int SCR_HEIGHT = 600;

#define NUM_OF_POINTS 30

// camera
Camera camera(glm::vec3(0.0f, 0.0f, 55.0f));
float lastX = (float)SCR_WIDTH / 2.0;
float lastY = (float)SCR_HEIGHT / 2.0;
bool firstMouse = true;

// timing
float deltaTime = 0.0f;
float lastFrame = 0.0f;
```

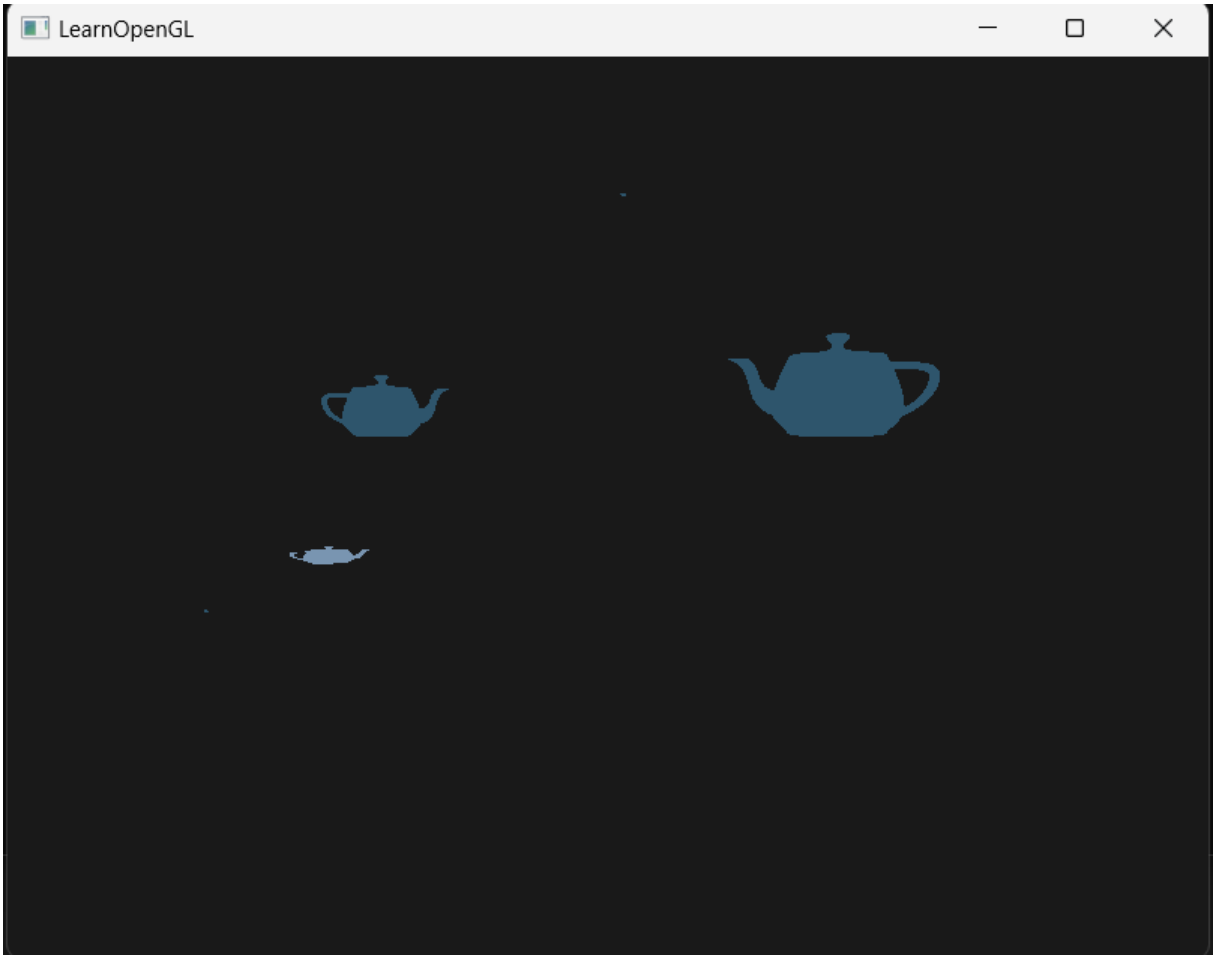
```
int main()
{
	// glfw: initialize and configure
	// ------------------------------
	glfwInit();
	glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
	glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
	glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);

#ifdef __APPLE__
	glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
#endif

	// glfw window creation
	// --------------------
	GLFWwindow* window = glfwCreateWindow(SCR_WIDTH, SCR_HEIGHT, "LearnOpenGL", NULL,
NULL);
	if (window == NULL)
	{
		std::cout << "Failed to create GLFW window" << std::endl;
		glfwTerminate();
		return -1;
	}
	glfwMakeContextCurrent(window);
	glfwSetFramebufferSizeCallback(window, framebuffer_size_callback);
	glfwSetCursorPosCallback(window, mouse_callback);
	glfwSetScrollCallback(window, scroll_callback);

	// tell GLFW to capture our mouse
	glfwSetInputMode(window, GLFW_CURSOR, GLFW_CURSOR_DISABLED);

	// glad: load all OpenGL function pointers
	// ---------------------------------------
	if (!gladLoadGLLoader((GLADloadproc)glfwGetProcAddress))
	{
		std::cout << "Failed to initialize GLAD" << std::endl;
		return -1;
	}

	// configure global opengl state
	// -----------------------------
	glEnable(GL_DEPTH_TEST);

	// build and compile shaders
	// -------------------------
	Shader shader("10.2.instancing.vs", "10.2.instancing.fs");

	// load models
	// -----------
	Model planet(FileSystem::getPath("resources/objects/planet/planet.obj"));

	unsigned int texture1 = TextureFromFile("top.jpg",
FileSystem::getPath("resources/objects/lab4/").c_str(),0);

	unsigned int texture2 = TextureFromFile("green.png",
FileSystem::getPath("resources/objects/lab4/").c_str(),0);
	Model teapot(FileSystem::getPath("resources/objects/lab4/teapot/teapot.obj"));

	Shader shader1("eksen.vs", "eksen.fs");


	float vertices[NUM_OF_POINTS * 3];
	for (int i = 0; i < NUM_OF_POINTS; i++)
	{
		vertices[3 * i] = i;
		vertices[3 * i+1] = 0;
		vertices[3 * i+2] = 0;
	}
```

```
	unsigned int VBO, VAO;
	glGenVertexArrays(1,&VAO);
	glGenBuffers(1, &VBO);

	glBindVertexArray(VAO);
	glBindBuffer(GL_ARRAY_BUFFER, VBO);

	glVertexAttribPointer(0, 3, GL_FLOAT, GL_FALSE, 3 * sizeof(float), (void*)0);
	glEnableVertexAttribArray(0);

	glm::mat4 model_axis;
	// render loop
	// -----------
	while (!glfwWindowShouldClose(window))
	{
		// per-frame time logic
		// --------------------
		float currentFrame = static_cast<float>(glfwGetTime());
		deltaTime = (currentFrame - lastFrame) * 10;
		lastFrame = currentFrame;

		// input
		// -----
		processInput(window);

		// render
		// ------
		glClearColor(0.1f, 0.1f, 0.1f, 1.0f);
		glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

		// configure transformation matrices
		glm::mat4 projection = glm::perspective(glm::radians(45.0f), (float)SCR_WIDTH /
(float)SCR_HEIGHT, 0.1f, 1000.0f);
		glm::mat4 view = camera.GetViewMatrix();;
		shader.use();
		shader.setMat4("projection", projection);
		shader.setMat4("view", view);

		// draw planet
		glm::mat4 model = glm::mat4(1.0f);
		model = glm::translate(model, glm::vec3(10.0f, 10.0f, 10.0f));
		float rotAngle = 150;
		model = glm::rotate(model, glm::radians(rotAngle), glm::vec3(0.0f, 1.0f, 0.0f));
		model = glm::translate(model, glm::vec3(-10.0f, -10.0f, -10.0f));
		shader.setMat4("model", model);
		teapot.Draw(shader);

		// draw fish
		glm::mat4 model2 = glm::mat4(1.0f);
		model2 = glm::translate(model2, glm::vec3(-10.0f, -10.0f, -10.0f));
		model2 = glm::scale(model2, glm::vec3(0.7f, 0.3f, 0.3f));
		model2 = glm::translate(model2, glm::vec3(10.0f, 10.0f, 10.0f));
		glBindTexture(GL_TEXTURE_2D, texture1);
		shader.setMat4("model", model2);
		teapot.Draw(shader);

		// draw teapot
		glm::mat4 model3 = glm::mat4(1.0f);
		model3 = glm::translate(model3, glm::vec3(-1.0f, -1.0f, -1.0f));
		model3 = glm::scale(model3, glm::vec3(1.0f, 1.0f, -1.0f));
		model3 = glm::translate(model3, glm::vec3(1.0f, 1.0f, 1.0f));
		glBindTexture(GL_TEXTURE_2D, texture2);
		shader.setMat4("model", model3);
		teapot.Draw(shader);

		// draw teapot
		glm::mat4 model4 = glm::mat4(1.0f);
		model4 = glm::translate(model4, glm::vec3(-10.0f, -10.0f, -10.0f));
		model4 = glm::scale(model4, glm::vec3(0.05f, 0.05f, 0.05f));
```

```
            glBindTexture(GL_TEXTURE_2D, texture2);
            shader.setMat4("model", model4);
            teapot.Draw(shader);

            // draw teapot
            glm::mat4 model5 = glm::mat4(1.0f);
            model5 = glm::translate(model5, glm::vec3(10.0f, 10.0f, 10.0f));
            model5 = glm::scale(model5, glm::vec3(0.05f, 0.05f, 0.05f));
            glBindTexture(GL_TEXTURE_2D, texture2);
            shader.setMat4("model", model5);
            teapot.Draw(shader);

            // draw teapot
            glm::mat4 model6 = glm::mat4(1.0f);
            model6 = glm::translate(model6, glm::vec3(1.0f, 1.0f, 1.0f));
            model6 = glm::scale(model6, glm::vec3(0.05f, 0.05f, 0.05f));
            glBindTexture(GL_TEXTURE_2D, texture2);
            shader.setMat4("model", model6);
            teapot.Draw(shader);

            shader1.use();
            shader1.setMat4("projection", projection);
            shader1.setMat4("view",view);
            model_axis = glm::mat4(1.0f);
            shader1.setMat4("model",model_axis);

            shader1.setVec4("ourColor",1.0f,0.0f,0.0f,1.0f);
            glBindVertexArray(VAO);
            glDrawArrays(GL_LINES,0,NUM_OF_POINTS);

            model_axis = glm::rotate(model_axis, glm::radians(90.0f), glm::vec3(0, 0, 1));
            shader1.setMat4("model", model_axis);
            shader1.setVec4("ourColor", 0.0f, 1.0f, 0.0f, 1.0f);
            glDrawArrays(GL_LINES, 0, NUM_OF_POINTS);

            model_axis = glm::rotate(model_axis, glm::radians(-90.0f), glm::vec3(0, 0, 1));
            shader1.setMat4("model", model_axis);
            shader1.setVec4("ourColor", 0.0f, 0.0f, 1.0f, 1.0f);
            glDrawArrays(GL_LINES, 0, NUM_OF_POINTS);


            // glfw: swap buffers and poll IO events (keys pressed/released, mouse moved
etc.)
            // -------------------------------------------------------------------------
-
            glfwSwapBuffers(window);
            glfwPollEvents();
      }

      glfwTerminate();
      return 0;
}

// process all input: query GLFW whether relevant keys are pressed/released this frame and
react accordingly
// ---------------------------------------------------------------------------------------
--------------
void processInput(GLFWwindow* window)
{
      if (glfwGetKey(window, GLFW_KEY_ESCAPE) == GLFW_PRESS)
            glfwSetWindowShouldClose(window, true);

      if (glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS)
            camera.ProcessKeyboard(FORWARD, deltaTime);
      if (glfwGetKey(window, GLFW_KEY_S) == GLFW_PRESS)
            camera.ProcessKeyboard(BACKWARD, deltaTime);
      if (glfwGetKey(window, GLFW_KEY_A) == GLFW_PRESS)
            camera.ProcessKeyboard(LEFT, deltaTime);
      if (glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS)
```

```
            camera.ProcessKeyboard(RIGHT, deltaTime);
}

// glfw: whenever the window size changed (by OS or user resize) this callback function
executes
// ---------------------------------------------------------------------------------------
--
void framebuffer_size_callback(GLFWwindow* window, int width, int height)
{
      // make sure the viewport matches the new window dimensions; note that width and
      // height will be significantly larger than specified on retina displays.
      glViewport(0, 0, width, height);
}

// glfw: whenever the mouse moves, this callback is called
// -------------------------------------------------------
void mouse_callback(GLFWwindow* window, double xposIn, double yposIn)
{
      float xpos = static_cast<float>(xposIn);
      float ypos = static_cast<float>(yposIn);

      if (firstMouse)
      {
            lastX = xpos;
            lastY = ypos;
            firstMouse = false;
      }

      float xoffset = xpos - lastX;
      float yoffset = lastY - ypos; // reversed since y-coordinates go from bottom to top

      lastX = xpos;
      lastY = ypos;

      camera.ProcessMouseMovement(xoffset, yoffset);
}

// glfw: whenever the mouse scroll wheel scrolls, this callback is called
// ----------------------------------------------------------------------
void scroll_callback(GLFWwindow* window, double xoffset, double yoffset)
{
      camera.ProcessMouseScroll(static_cast<float>(yoffset));
}
```

CEVAP EKRAN GÖRÜNTÜSÜ


LearnOpenGL