ФГБОУ ВО “Чувашский государственный университет им.

И.Н. Ульянова”

Кафедра вычислительной техники

Лабораторная работа № 2

По дисциплине “Алгоритмические основы компьютерной графики”

Размещение объектов по сцене.

Выполнил: студент группы ИВТ-42-19

Кузьмина Екатерина Андреевна

Проверил:

Степанов В.В.

Чебоксары 2021

**Цель работы:** вывести на экран несколько трехмерных объектов в разных местах сцены.

**Выполнение работы**

Лабораторная работа №2 строится на основе предыдущей работы с

внесением следующих изменений:

1. В отдельном модуле создать класс GraphicObject вышеописанной структуры для работы с трехмерным объектом.
2. Реализовать вывод четырех чайников разных цветов расположенных на осях OX и OZ с носиками, повернутыми в центр сцены, как изображено выше. Для хранения и вывода трехмерных объектов использовать ранее созданный класс GraphicObject.
3. Вывести все объекты, реализуя классический цикл программы с анимацией:
4. Очистить буфер экрана (буферы цвета, глубины и т.д.)
5. Установить матрицу камеры
6. Для каждого объекта (в методе draw):
   1. сохранит матрицу наблюдения в стеке
   2. умножить матрицу наблюдения на матрицу модели
   3. Вывести модель (установив так же её цвет)
   4. Восстановить матрицу наблюдения из стека
7. Поменять местами передний и задний буферы
8. Перейти к шагу 1

Текст заголовочного файла модуля с классом GraphicObject.h:

#pragma once

#include<windows.h>

#include<stdio.h>

#include <GL/gl.h>

#include <GL/glut.h>

#include"GL/freeglut.h"

#include"math.h"

#include"GLM/GLM/glm.hpp"

#include"GLM/GLM/gtx/transform.hpp"

#include"GLM/GLM/gtc/type\_ptr.hpp"

using namespace glm;

// КЛАСС ДЛЯ ПРЕДСТАВЛЕНИЯ ОДНОГО ГРАФИЧЕСКОГО ОБЪЕКТА

class GraphicObject{

private:

// Позиция и угол поворота для объекта

//Glfloat position[3];

vec3 position;

GLfloat angle;

// Матрица модели (расположение объекта) ‐ чтоб не вычислять каждый раз

//Glfloat modelMatrix[16];

mat4 modelMatrix;

// Цвет модели

//Glfloat color[3];

vec3 color;

public:

// Конструктор

GraphicObject(void);

// Задать позицию объекта

void setPosition(float x, float y, float z);

// Задать угол поворота в градусах относительно оси OY

void setAngle(float a);

// Задать цвет модели

void setСolor(float r, float g, float b);

// Вывести объект

void draw(void);

};

Текст файла реализации модуля с классом GraphicObject.cpp:

#include "GraphicObject.h"

using namespace glm;

// Конструктор

GraphicObject::GraphicObject(void){}

// задать позицию объекта

void GraphicObject::setPosition(float x, float y, float z){

position=vec3(x,y,z);

}

// задать угол поворота в градусах относительно оси OY

void GraphicObject::setAngle(float a){

angle = a;

}

// Задать цвет модели

void GraphicObject::setСolor(float r, float g, float b){

color=vec3(r,g,b);

}

// вывести объект

void GraphicObject::draw(void){

glPushMatrix();

glColor3fv(value\_ptr(color));

modelMatrix = translate(position) \* rotate(radians(angle), vec3(0.0, 1.0, 0.0));

glMultMatrixf(value\_ptr(modelMatrix));

glutWireTeapot(1.0);

glPopMatrix();

}

Текст основной программы с комментариями:

#include <windows.h>

#include <GL/gl.h>

#include <GL/glut.h>

#include "GL/freeglut.h"

#include <iostream>

#include"GraphicObject.h"

using namespace glm;

using namespace std;

const int gObjCount = 4;

GraphicObject gObjs[gObjCount];

// функция вызывается каждые 20 мс

void Simulation(int value)

{

// устанавливаем признак того, что окно нуждается в перерисовке

glutPostRedisplay();

// эта же функция будет вызвана еще раз через 20 мс

glutTimerFunc(20, Simulation, 0);

};

// функция, вызываемая при изменении размеров окна

void Reshape(int w, int h)

{

// установить новую область просмотра, равную всей области окна

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

// установить матрицу проекции с правильным аспектом

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(25.0, (float)w / h, 0.2, 70.0);

};

// функция вызывается при перерисовке окна

// в том числе и принудительно, по командам glutPostRedisplay

void Display(void)

{

// отчищаем буфер цвета

glClearColor(0.22, 0.88, 0.11, 1.0);

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

// включаем тест глубины

glEnable(GL\_DEPTH\_TEST);

// устанавливаем камеру

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

//gluLookAt(5, 5, 7.5, 0, 0, 0, 0, 1, 0);

gluLookAt(15, 15, 7.5, 0, 0, 0, 0, 1, 0);

// выводим объекты

for (int i = 0; i < gObjCount; i++)

{

gObjs[i].draw();

}

// смена переднего и заднего буферов

glutSwapBuffers();

};

// Функция обработки нажатия клавиш

void KeyboardFunc(unsigned char key, int x, int y)

{

printf("Key code is %i\n", (unsigned int)key);

};

int main(int argc, char\*\* argv) {

// инициализация библиотеки GLUT

glutInit(&argc, argv);

// инициализация дисплея (формат вывода)

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DOUBLE | GLUT\_DEPTH | GLUT\_MULTISAMPLE);

// создание окна:

// 1. устанавливаем верхний левый угол окна

glutInitWindowPosition(200, 200);

// 2. устанавливаем размер окна

glutInitWindowSize(600, 600);

// 3. создаем окно

glutCreateWindow("laba2");

// устанавливаем функцию, которая будет вызываться для перерисовки окна

glutDisplayFunc(Display);

// устанавливаем функцию, которая будет вызываться при изменении размеров окна

glutReshapeFunc(Reshape);

// устанавливаем функцию которая будет вызвана через 20 мс

glutTimerFunc(20, Simulation, 0);

// устанавливаем функцию, которая будет вызываться при нажатии на клавишу

glutKeyboardFunc(KeyboardFunc);

gObjs[0].setPosition(-3.0,0.0,0.0);

gObjs[0].setAngle(0);

gObjs[0].setСolor(1.0,0.5,0.0);

gObjs[1].setPosition(3.0,0.0,0.0);

gObjs[1].setAngle(180);

gObjs[1].setСolor(0.0,0.8,1.0);

gObjs[2].setPosition(0.0,0.0,3.0);

gObjs[2].setAngle(90);

gObjs[2].setСolor(1.0,0.0,0.0);

gObjs[3].setPosition(0.0,0.0,-3.0);

gObjs[3].setAngle(-90);

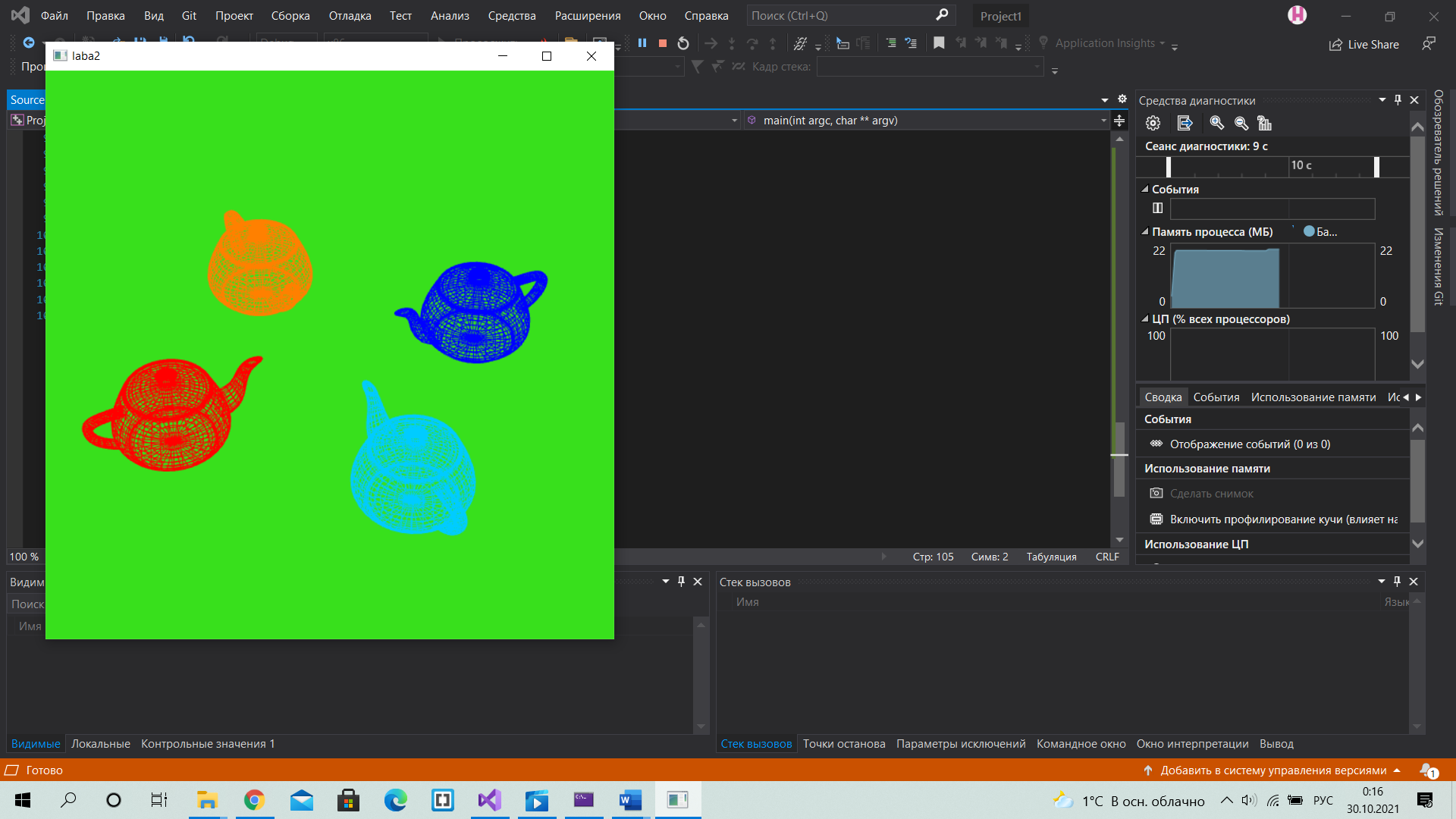
gObjs[3].setСolor(0.0,0.0,1.0);

// основной цикл обработки сообщений ОС

glutMainLoop();

}

Скриншот работы программы:



**Вывод:** вывела на экран несколько трехмерных объектов в разных местах сцены.