Федеральное государственное бюджетное образовательное учреждение высшего

образования

«Чувашский государственный университет им. И.Н. Ульянова»

Кафедра вычислительной техники

Лабораторная работа № 2

Использование библиотеки GLM.

Вариант 9

Выполнил: Иванов В.С.

студент группы ИВТ-41-22

Проверил: преподаватель

Васильев Никита Сергеевич

Чебоксары, 2024

Задание к лабораторной работе.

Лабораторная работа No2 строится на основе предыдущей работы. В рамках лабораторной работы

необходимо изучить модуль glm и принципы работы с контейнером std::vector. Кроме того, необходимо:

1. Реализовать автоматическое изменение цвета объекта. Цвет объекта меняется раз в секунду без каких-либо дополнительных действий пользователя в соответствии со следующим списком: белый, синий, красный, желтый и фиолетовый. При достижении последнего цвета из списка происходит возврат к первому цвету и смена цветов продолжается (для лучшего понимания, можно посмотреть пример к лабораторной работе).

2. Для задания списка цветов используется тип vec3 из библиотеки glm, а также контейнер std::vector. Иными словами, список цветов должен быть объявлен следующим образом:

// список цветов

vector<vec3> teapotColors;

3. Для автоматического изменения цвета необходимо доработать функцию simulation, которая вызывается по таймеру каждые 20 мс. При этом, частоту вызова функции simulation изменять не следует, поскольку чем выше частота симуляции, тем более плавным будет моделируемый процесс.

Кроме того, в функции simulation могут осуществляться и другие действия, которые происходят быстрее или медленнее. Для определения того, что прошла одна секунда и цвет необходимо изменить может потребоваться дополнительная переменная.

4. В процессе написания и отладки программы рекомендуется выводить на консоль вспомогательную информацию, чтобы контролировать состояние глобальных переменных, участвующих в симуляции или влияющих на вывод изображения на экран.

Текст программы с комментариями.

#include <windows.h>

#include <iostream>

#include <vector>

#include <GL/gl.h>

#include <GL/glu.h>

#include "GL/freeglut.h"

#include <glm/glm.hpp>

#include <glm/gtc/matrix\_transform.hpp>

#include <glm/gtc/type\_ptr.hpp>

// используем пространство имен стандартной библиотеки

using namespace std;

using namespace glm;

//вектор цветов

vector<vec3> colors{

{1.0f, 1.0f, 1.0f},

{0.0f, 0.0f, 1.0f},

{1.0f, 0.0f, 0.0f},

{1.0f, 1.0f, 0.0f},

{0.5f, 0.0f, 0.5f}

};

//переменные индекса, размера вектора цветов и счетчика времени

int index = 0;

int size = 5;

int timePassed = 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 и буфер глубины

glClearColor(0.22, 0.88, 0.1, 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);

// выводим объект - красный (1,0,0) чайник

//цвета используюся по индексу вектора цветов colors

glColor3f(colors[index].r,colors[index].g, colors[index].b);

glutWireTeapot(1.0);

// смена переднего и заднего буферов

glutSwapBuffers();

};

// функция вызывается каждые 20 мс

void simulation(int value)

{

//при вызове функции увеличиваем счетчик

timePassed += 20;

//при достижении 1000 мс сбрасываем счетчик и изменяем цвет

if (timePassed == 1000) {

index += 1;

//обнуляем счетчик

timePassed = 0;

//если дошди до конца вектора сбрасываем индекс

if (index == 5) index = 0;

cout << "Index of current color: " << index << endl;

}

// устанавливаем признак того, что окно нуждается в перерисовке

glutPostRedisplay();

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

glutTimerFunc(20, simulation, 0);

};

// Функция обработки нажатия клавиш

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

{

/\*if (key == 32) {

index += 1;

if (index == 5) index = 0;

}\*/

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

};

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

{

setlocale(LC\_ALL, "ru");

// 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(800, 600);

// 3. создаем окно

glutCreateWindow("Laba\_02");

// УСТАНОВКА ФУНКЦИЙ ОБРАТНОГО ВЫЗОВА

// 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);

// основной цикл обработки сообщений ОС

glutMainLoop();

};