Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Чувашский государственный университет имени И.Н.Ульянова».

Факультет информатики и вычислительной техники

**Лабораторная работа №1**

**Инициализация OpenGL. Использование библиотеки freeglut.**

Выполнил: Мясников А.Н

Проверил: Степанов В. В

Чебоксары 2021

**Задание к лабораторной работе.**

В рамках данной лабораторной работы необходимо изучить функции GLUT для инициализации OpenGL и взаимодействия с операционной системой, разобраться с приведенным примером, а так же внести следующие изменения:

1. Используя функцию glutKeyboardFunc добавить функцию обработки события нажатия на клавишу;

2. Реализовать циклическое изменение цвета объекта по нажатию на клавишу, используя предварительно заданный массив цветов. В массиве цвета лежат в следующем порядке: черный, белый, синий и красный. Следует учесть, что во время защиты может потребоваться добавление новых цветов в массив;

3. Для защиты следует заранее подумать над тем, как организовать автоматическое изменение цветов с заданной скоростью.

**Таблица используемых функций.**

|  |  |
| --- | --- |
| glutInit | Используется для инициализации библиотеки GLUT  glutInit(int \*argcp, char \*\*argv)  Первый параметр представляет из себя указатель на количество аргументов в командной строке, а второй - указатель на массив аргументов. |
| glutInitDisplayMode | Устанавливает начальный режим отображения.  void glutInitDisplayMode(unsigned int mode);  Может быть один или несколько параметров. Каждая константа обладает своим значением, например, GLUT\_RGB - Для отображения графической информации используются 3 компоненты цвета RGB. |
| glutInitWindowPosition | Задаёт начальное положение окна относительно верхнего левого угла экрана  void glutInitWindowPosition(int x, int y); |
| glutInitWindowSize | Задаёт начальный размер окна  void glutInitWindowSize(int width, int height);  width – ширина в пикселях  height – высота в пикселях |
| glutCreateWindow | Создает окно с заголовком, который указывается в параметре  glutCreateWindow(“title”); |
| glutPostRedisplay | Устанавливает признак того, что окно нуждается в перерисовке |
| glutDisplayFunc | Устанавливает функцию, которая будет вызываться для перерисовки окна  void glutDisplayFunc(void (\*func)(void));  func – функция обратного вызова дисплея |
| glutReshapeFunc | Устанавливает функцию, которая будет вызываться при изменении размеров окна  void glutReshapeFunc(void (\*func)(int width, int height));  func – фукция обратного вызова Reshape |
| glutKeyboardFunc | Устанавливает функцию, которая будет вызвана при нажатии на клавишу  void glutKeyboardFunc(void (\*func)(unsigned char key int x, int y));  func – функция обратного вызова клавиатуры |
| glutTimerFunc | Устанавливает функцию, которая будет вызвана через N мс  void glutTimerFunc(без знака в мсек,  void (\*функция)(значение int), значение); |
| glutMainLoop | Входит в цикл обработки событий GLUT |
| glutSwapBuffers | Отвечает за смену переднего и заднего буферов |
| glutWireTeapot | Представляют собой каркасный чайник. Генерируются как нормали поверхности, так и координаты текстуры для чайника. |

**Текст программы.**

#include <stdio.h>

#include <windows.h>

#include <GL/GL.h>

#include <GL/GLU.h>

#include <GL/freeglut.h>

#include <iostream>

const int simulationTimerCount = 20;

bool teapotColorReplaceSwitch = false;

int Timer\_count; //для таймера

int teapotColorIterator = 0;

const double teapotColorReplaceTimer\_limit = 1000;

double teapotColorReplaceTimer\_countdown = 0;

//константы для расположения и размеров окна

const int x = 500;

const int y = 200;

const int width = 1000;

const int height = 800;

const int colorsCount = 4; //количество возможных цветов

struct colors {

float r;

float g;

float b;

};

colors teaPotColor;

colors defaultTeaPotColor;

colors colorsArray[colorsCount];

bool teaPotColorReplaceSwitch = false;

int teaPotColorIterator = 0;

//функция для установки цвета чайника

void setColors() {

//начальный цвет чайника

defaultTeaPotColor.r = 1.0;

defaultTeaPotColor.g = 0;

defaultTeaPotColor.b = 0;

colorsArray[0].r = 0.0; colorsArray[0].g = 0.0; colorsArray[0].b = 0.0; //чёрный

colorsArray[1].r = 1.0; colorsArray[1].g = 1.0; colorsArray[1].b = 1.0; // белый

colorsArray[2].r = 0.0; colorsArray[2].g = 0.0; colorsArray[2].b = 1.0; // синий

colorsArray[3].r = 1.0; colorsArray[3].g = 0.0; colorsArray[3].b = 0.0; // красный

}

void Simulation(int value) {

glutPostRedisplay();

if (teapotColorReplaceSwitch == true) {

Timer\_count += simulationTimerCount;

if (Timer\_count == 1000) {

Timer\_count = 0;

teapotColorIterator++;

teaPotColor.r = colorsArray[teapotColorIterator].r;

teaPotColor.g = colorsArray[teapotColorIterator].g;

teaPotColor.b = colorsArray[teapotColorIterator].b;

if (teapotColorIterator == 4) {

teapotColorIterator = 0;

}

}

}

glutTimerFunc(simulationTimerCount, 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, (double)w / h, 0.2, 70.0);

}

// функция вызывается при перерисовке окна

// в том числе и принудительно, по командам glutPostRedisplay

void Display() {

// отчищаем буфер цвета

glClearColor(0.0f, 0.9f, 0.25f, 1.0f);

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 (чайник)

glColor3f(teaPotColor.r, teaPotColor.g, teaPotColor.b);

glutWireTeapot(1.0);

// смена переднего и заднего буферов

glutSwapBuffers();

}

// Функция обработки нажатия клавиш

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

switch (key) {

case 'q': {

exit(0);

}

case 'f': {

glutFullScreen();

}

case '1': {

teaPotColor.r = colorsArray[teaPotColorIterator].r;

teaPotColor.g = colorsArray[teaPotColorIterator].g;

teaPotColor.b = colorsArray[teaPotColorIterator].b;

key == '1' ? teaPotColorIterator++ : 0;

teaPotColorIterator > colorsCount - 1 ?

teaPotColorIterator = 0 :

teaPotColorIterator < 0 ?

teaPotColorIterator = colorsCount - 1 : 0;

}; break;

case '2':{

teapotColorReplaceSwitch = true;

}; break;

case'3': {

teapotColorReplaceSwitch = false;

Timer\_count = 0;

}; break;

}

}

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

setlocale(LC\_ALL, "RUSSIAN");

setColors();

teaPotColor.r = defaultTeaPotColor.r;

teaPotColor.g = defaultTeaPotColor.g;

teaPotColor.b = defaultTeaPotColor.b;

// инициализация библиотеки GLUT

glutInit(&argc, argv);

// инициализация дисплея (формат вывода)

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

// создание окна:

// 1. устанавливаем положение окна

glutInitWindowPosition(x, y);

// 2. устанавливаем размер окна

glutInitWindowSize(width, height);

// 3. создаем окно

glutCreateWindow("lab1");

// 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);

glutMainLoop();

return;

}

**Результат работы программы.**







