МИНОБРНАУКИ РОССИИ

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

Факультет информатики и вычислительной техники

Кафедра вычислительной техники

Отчет по лабораторной работе №2

по дисциплине: «Алгоритмические основы компьютерной графики»

«**Размещение объектов на сцене**»

Выполнила: студент гр.ИВТ-41-19

Семенов Александр Александрович

Проверил:

Степанов В. В.

Чебоксары 2021

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:

***1. Очистить буфер экрана (буферы цвета, глубины и т.д.)  
2. Установить матрицу камеры  
3. Для каждого объекта (в методе draw):  
a. сохранит матрицу наблюдения в стеке  
b. умножить матрицу наблюдения на матрицу модели  
c. Вывести модель (установив так же её цвет)  
d. Восстановить матрицу наблюдения из стека  
4. Поменять местами передний и задний буферы  
5. Перейти к шагу 1***

Текст программы с подробными комментариями:

//GraphicObject.h

#pragma once

#include <GL/freeglut.h>

#include<GL/gl.h>

#include<GL/GLU.h>

#include <stdio.h>

#include <windows.h>

using namespace std;

class GraphicObject

{

private:

// Позиция и угол поворота для объекта

GLfloat position[3];

GLfloat angle;

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

GLfloat modelMatrix[16];

// Цвет модели

GLfloat color[3];

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);

};

//GraphicObject.cpp

#include "GraphicObject.h"

GraphicObject::GraphicObject(void)

{}

// задать позицию объекта

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

{

position[0] = x;

position[1] = y;

position[2] = 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[0] = r;

color[1] = g;

color[2] = b;

}

// вывести объект

void GraphicObject::draw(void)

{

glPushMatrix();

glColor3f(color[0], color[1], color[2]);

glTranslatef(position[0], position[1], position[2]);

glRotatef(angle, 0.0, 1.0, 0.0);

glutWireTeapot(1.0);

glPopMatrix();

}

//Source.cpp

#include <GL/freeglut.h>

#include<GL/gl.h>

#include<GL/GLU.h>

#include "GraphicObject.h"

#include <stdio.h>

#include <windows.h>

bool count\_c = false;

double X1 = 0, Y1 = 0, Z1=0;

struct RGB{

double R;

double G;

double B;

};

int count\_t = 0;

RGB Colors[4] = { {0.0,0.0,0.0},{1,1,1},{0,0,1},{1,0,0} };

double c1=0, c2=0, c3 =0 ;

int i = 0;

double teapod\_s = 1.0;

GLfloat M0[16] = { -1,0,0,0, 0,1,0,0, 0,0,-1,0, 4,0,0,1 };

GLfloat M1[16] = { 1,0,0,0, 0,1,0,0, 0,0, 1,0, -4,0,0,1 };

const int graphicObjectCount = 4;

GraphicObject graphicObjects[graphicObjectCount];

// функция вызывается каждые 20 мс

void Simulation(int value)

{

// устанавливаем признак того, что окно нуждается в перерисовке

glutPostRedisplay();

if (count\_c == true)

{

count\_t += 20;

if (count\_t == 1000)

{

count\_t = 0;

i++;

c1 = Colors[i].R;

c2 = Colors[i].G;

c3 = Colors[i].B;

if (i == 4)

i = 0;

}

}

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

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

glColor3f(c1, c2, c3);

// glutWireTeapot(teapod\_s);

graphicObjects[0].draw();

graphicObjects[1].draw();

graphicObjects[2].draw();

graphicObjects[3].draw();

//glColor3f(0, 0, 1);

// glutWireTeacup(1.0);

// 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);

switch (key)

{

case 'p':

exit(0);

break;

case 'b':

i++;

c1 = Colors[i].R;

c2 = Colors[i].G;

c3 = Colors[i].B;

glutPostRedisplay();

if (i == 4)

i = 0;

break;

case 'n':

count\_c = true;

break;

case 'm':

count\_c = false;

count\_t = 0;

break;

case 'h':

teapod\_s += 1.0;

break;

case 'j':

teapod\_s -= 1.0;

break;

}

};

void 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("laba1");

// 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);

graphicObjects[0].setPosition(-2.5f, 0.0f, 0.0f);

graphicObjects[0].setAngle(0);

graphicObjects[0].setСolor(1.0, 1.0, 1.0);

graphicObjects[1].setPosition(0.0f, 0.0f, 2.5f);

graphicObjects[1].setAngle(90);

graphicObjects[1].setСolor(0.0, 0.0, 0.0);

graphicObjects[2].setPosition(0.0f, 0.0f, -2.5f);

graphicObjects[2].setAngle(-90);

graphicObjects[2].setСolor(0.0, 0.0, 1.0);

graphicObjects[3].setPosition(2.5f, 0.0f, 0.0f);

graphicObjects[3].setAngle(180);

graphicObjects[3].setСolor(1.0, 0.0, 0.0);

// основной цикл обработки сообщений ОС

glutMainLoop();

}

Скриншот работы программы:

