МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

Кафедра компьютерной технологий и программной инженерии

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| асс. |  |  |  |  |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4 |
| «Работа с буфером глубины, прозрачностью» |
| по курсу: КОМПЬЮТЕРНАЯ ГРАФИКА |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4831 |  |  |  | К.А.Корюнщенков |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2020

**Цель работы**

Целью работы является получение практических навыков по работе с буфером глубины и прозрачностью в OpenGL.

**Задание на лабораторную работу**

Вывести несколько пересекающихся объектов, как минимум один из них должен быть прозрачным. Включить 3 источника света, задать свойства поверхностей и источников. Организовать раздельное вращение источников света вокруг неподвижной сцены, управляемое с клавиатуры или мышью.

**Ход работы**

**Листинг программы:**

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#include <cmath>

#include <GLUT/glut.h>

#include <cmath>

using namespace std;

int WinLenght = 800; // Ширина окна

int WinHight = 800; // Высота окна

// Глобальные переменные

int spiny = 0;

int spinx = 0;

int spinz = 0;

int Sun = 0;

int Sun1 = 0;

int Sun2 = 0;

bool L1 = false;

bool L2 = false;

bool L3 = false;

bool L4 = false;

void display();

void reshape(int w, int h){

glViewport(0, 0, (GLsizei)w, (GLsizei)h); // Создание окна отрисовки на весь экран

glMatrixMode(GL\_PROJECTION); // Работа в матрице проекций (с областью отображения)

glLoadIdentity(); // Очистка области отображения

// Задаем область видимости glOrtho-прямоуольник не меняюий пропрции. glFrustum - пирамида, чем предмет дальше, тем он меньше

glOrtho(-7.0, 7.0, -7.0, 7.0, -7, 7.0);

//glFrustum(-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);

glMatrixMode(GL\_MODELVIEW); // Работа с матрицей вида

glLoadIdentity(); // Очистка матрицы вида

}

// Функции которые надо выполнить 1 раз

void Init(){

glClearColor(0, 0, 0, 1); // При очистке экрана залить все черным(0,0,0) цветом

glShadeModel(GL\_SMOOTH); // Динамическая заливка

// Включение света

glLightModeli(GL\_LIGHT\_MODEL\_TWO\_SIDE, GL\_TRUE); // Освещение всех сторон полигона

glEnable(GL\_LIGHTING);

//glEnable(GL\_LIGHT0);

//glEnable(GL\_LIGHT1);

//glEnable(GL\_LIGHT2);

//glEnable(GL\_LIGHT3);

//glEnable(GL\_LIGHT4);

glColorMaterial(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE); // Цветом материала считается его диффузный цвет

glEnable(GL\_COLOR\_MATERIAL); // Разрешить материалам использовать цвет

glEnable(GL\_DEPTH\_TEST); // Тест глубины отвечает за удаление объектов загороженных другими

glEnable(GL\_NORMALIZE);

}

//Реакция на клавиатуру

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

switch (key) {

case 'd':

if (spiny == -360)

spiny = -1;

else

spiny--;

break;

case 'a':

if (spiny == 360)

spiny = 1;

else

spiny++;

glutPostRedisplay(); // Перерисовать дисплей

break;

case 's':

if (spinx == -360)

spinx = -1;

else

spinx--;

glutPostRedisplay(); // Перерисовать дисплей

break;

case 'w':

if (spinx == 360)

spinx = 1;

else

spinx++;

glutPostRedisplay(); // Перерисовать дисплей

break;

case 'q':

if (spinz == -360)

spinz = -1;

else

spinz--;

glutPostRedisplay(); // Перерисовать дисплей

break;

case 'e':

if (spinz == 360)

spinz = 1;

else

spinz++;

glutPostRedisplay(); // Перерисовать дисплей

break;

case 'z':

if (Sun == -360)

Sun = -1;

else

Sun--;

glutPostRedisplay(); // Перерисовать дисплей

break;

case 'x':

if (Sun == 360)

Sun = 1;

else

Sun++;

glutPostRedisplay(); // Перерисовать дисплей

break;

case 'c':

if (Sun1 == -360)

Sun1 = -1;

else

Sun1--;

glutPostRedisplay(); // Перерисовать дисплей

break;

case 'v':

if (Sun1 == 360)

Sun1 = 1;

else

Sun1++;

glutPostRedisplay(); // Перерисовать дисплей

break;

case 'b':

if (Sun2 == -360)

Sun2 = -1;

else

Sun2--;

glutPostRedisplay(); // Перерисовать дисплей

break;

case 'n':

if (Sun2 == 360)

Sun2 = 1;

else

Sun2++;

glutPostRedisplay(); // Перерисовать дисплей

break;

case '1':

if (L1){

glDisable(GL\_LIGHT4);

L1 = false;

}

else{

glEnable(GL\_LIGHT4);

L1 = true;

}

glutPostRedisplay(); // Перерисовать дисплей

break;

case '2':

if (L2){

glDisable(GL\_LIGHT1);

L2 = false;

}

else{

glEnable(GL\_LIGHT1);

L2 = true;

}

glutPostRedisplay(); // Перерисовать дисплей

break;

case '3':

if (L3){

glDisable(GL\_LIGHT2);

L3 = false;

}

else{

glEnable(GL\_LIGHT2);

L3 = true;

}

glutPostRedisplay(); // Перерисовать дисплей

break;

case '4':

if (L4){

glDisable(GL\_LIGHT3);

L4 = false;

}

else{

glEnable(GL\_LIGHT3);

L4 = true;

}

break;

default:

break;

}

glutPostRedisplay(); // Перерисовать дисплей

}

void MaterialChange(GLfloat diffuse\_red, GLfloat diffuse\_green, GLfloat diffuse\_blue, GLfloat specular\_red, GLfloat specular\_green, GLfloat specular\_blue, GLfloat shininess, GLfloat emission\_red, GLfloat emission\_green, GLfloat emission\_blue){

GLfloat mat\_diffuse[] = { diffuse\_red,diffuse\_green,diffuse\_blue,1.0 }; // Параметры основного цвета

GLfloat mat\_specular[] = { specular\_red,specular\_green,specular\_blue,1.0 }; // Параметры зеркального цвета

GLfloat mat\_shininess[] = { shininess }; // Процент зеркального отражения

GLfloat mat\_emission[] = { emission\_red,emission\_green,emission\_blue,1.0 }; // Параметры цвета излучаемого материалом

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE, mat\_diffuse);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SPECULAR, mat\_specular);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SHININESS, mat\_shininess);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, mat\_emission);

}

void DrawOrts(){

// Ось x

glBegin(GL\_LINES); // Начать рисовать линию

glColor3ub(0, 0, 255);// Установка цвета вершины 3 входных параметра R G B типа unsignit bait(0-255)

glVertex3f(0.0, 0.0, 0.0); //gl - использовать OpenGL; Vertex - нарисовать вершину; 3 - входных параметра(х,y,z); f - float тип входных параметров

glVertex3f(1.0, 0.0, 0.0);

glEnd();// Закончить рисовать линию

// Ось y

glBegin(GL\_LINES); // Начать рисовать линию

glColor3ub(0, 255, 0);// Установка цвета вершины 3 входных параметра R G B типа unsignit bait(0-255)

glVertex3f(0.0, 0.0, 0.0); //gl - использовать OpenGL; Vertex - нарисовать вершину; 3 - входных параметра(х,y,z); f - float тип входных параметров

glVertex3f(0.0, 1.0, 0.0);

glEnd();// Закончить рисовать линию

// Ось z

glBegin(GL\_LINES); // Начать рисовать линию

glColor3ub(255, 0, 0);// Установка цвета вершины 3 входных параметра R G B типа unsignit bait(0-255)

glVertex3f(0.0, 0.0, 0.0); //gl - использовать OpenGL; Vertex - нарисовать вершину; 3 - входных параметра(х,y,z); f - float тип входных параметров

glVertex3f(0.0, 0.0, 1.0);

glEnd();// Закончить рисовать линию

}

void DrawSq(float a, float b){

for (float i = 0; i < a; i = i + 0.1){

for (float j = 0; j < b; j = j + 0.1){

glBegin(GL\_POLYGON);

glVertex2f(0 + i, 0 + j);

glVertex2f(0.1 + i, 0 + j);

glVertex2f(0.1 + i, 0.1 + j);

glVertex2f(0 + i, 0.1 + j);

glEnd();

}

}

}

void Light1(){

glPushMatrix();

GLfloat pos[] = { 0.0,0.0,0,1.0 }; // Позиция света (х=1 y=1 z=1) 1-в конкретной точке, 0 солнце

GLfloat lightcolor[] = { 1.0,0.0,0.0,1.0 }; // Белый цвет (r = 1 g = 1 b = 1 a = 1)

GLfloat dir[] = { 0.0,0.0,5 }; // Направление света

GLfloat light\_specular[] = { 1.0,1.0,1.0,1.0 };

glRotatef(Sun, 0.0, 1.0, 1.0);

glTranslated(5, 0, 0);

glRotated(-90, 0, 1, 0);

glLightfv(GL\_LIGHT4, GL\_POSITION, pos); // Установка поззиции источника света

glLightfv(GL\_LIGHT4, GL\_DIFFUSE, lightcolor); // Установка основного света белым(r=1 g=1 b=1 a=1 )

//glLightfv(GL\_LIGHT4, GL\_SPECULAR, light\_specular);

glLightfv(GL\_LIGHT4, GL\_SPOT\_DIRECTION, dir);

glLightf(GL\_LIGHT4, GL\_SPOT\_CUTOFF, 30.0);

glDisable(GL\_LIGHTING);

glColor3ub(255, 0, 0);

glPushMatrix();

glTranslated(-0.1, -0.1, 0);

DrawSq(0.2, 0.2);

glTranslated(0.0, 0.0, 5.5);

DrawSq(0.2, 0.2);

glPopMatrix();

// Ось направления света

glBegin(GL\_LINES); // Начать рисовать линию

glColor3ub(255, 255, 255);

glVertex3f(0.0, 0.0, 0.0); //gl - использовать OpenGL; Vertex - нарисовать вершину; 3 - входных параметра(х,y,z); f - float тип входных параметров

glVertex3f(0.0, 0.0, 1.0);

glEnd();// Закончить рисовать линию

glEnable(GL\_LIGHTING);

glPopMatrix();

}

void Light2(){

glPushMatrix();

GLfloat pos[] = { 0.0,0.0,0,1.0 }; // Позиция света (х=1 y=1 z=1) 1-в конкретной точке, 0 солнце

GLfloat lightcolor[] = { 0.0,1.0,0.0,1.0 };

GLfloat dir[] = { 0.0,0.0,5 }; // Направление света

GLfloat light\_specular[] = { 0.0,1.0,0.0,1.0 };

glRotatef(Sun1, 0.0, 1.0, 0.0);

glTranslated(5, 0, 0);

glRotated(-90, 0, 1, 0);

glLightfv(GL\_LIGHT1, GL\_POSITION, pos); // Установка поззиции источника света

glLightfv(GL\_LIGHT1, GL\_DIFFUSE, lightcolor); // Установка основного света белым(r=1 g=1 b=1 a=1 )

glLightfv(GL\_LIGHT1, GL\_AMBIENT, lightcolor); // Установка основного света белым(r=1 g=1 b=1 a=1 )

// glLightfv(GL\_LIGHT1, GL\_SPECULAR, light\_specular);

glLightfv(GL\_LIGHT1, GL\_SPOT\_DIRECTION, dir);

glLightf(GL\_LIGHT1, GL\_SPOT\_CUTOFF, 100.0);

glDisable(GL\_LIGHTING);

glColor3ub(0, 255, 0);

glPushMatrix();

glTranslated(-0.1, -0.1, 0);

DrawSq(0.2, 0.2);

glTranslated(0.0, 0.0, 5.5);

DrawSq(0.2, 0.2);

glPopMatrix();

// Ось направления света

glBegin(GL\_LINES); // Начать рисовать линию

glColor3ub(255, 255, 255);

glVertex3f(0.0, 0.0, 0.0); //gl - использовать OpenGL; Vertex - нарисовать вершину; 3 - входных параметра(х,y,z); f - float тип входных параметров

glVertex3f(0.0, 0.0, 1.0);

glEnd();// Закончить рисовать линию

glEnable(GL\_LIGHTING);

glPopMatrix();

}

void Light3(){

glPushMatrix();

GLfloat pos[] = { 0.0,0.0,0,1.0 }; // Позиция света (х=1 y=1 z=1) 1-в конкретной точке, 0 солнце

GLfloat lightcolor[] = { 0.0,0.0,1.0,1.0 };

GLfloat dir[] = { 0.0,0.0,5 }; // Направление света

GLfloat light\_specular[] = { 0.0,0.0,1.0,1.0 };

glRotatef(Sun2, 0.0, 0.0, 1.0);

glTranslated(5, 0, 0);

glRotated(-90, 0, 1, 0);

glLightfv(GL\_LIGHT2, GL\_POSITION, pos); // Установка поззиции источника света

glLightfv(GL\_LIGHT2, GL\_DIFFUSE, lightcolor); // Установка основного света белым(r=1 g=1 b=1 a=1 )

glLightfv(GL\_LIGHT2, GL\_AMBIENT, lightcolor); // Установка основного света белым(r=1 g=1 b=1 a=1 )

//glLightfv(GL\_LIGHT2, GL\_SPECULAR, light\_specular);

glLightfv(GL\_LIGHT2, GL\_SPOT\_DIRECTION, dir);

glLightf(GL\_LIGHT2, GL\_SPOT\_CUTOFF, 50.0);

glDisable(GL\_LIGHTING);

glColor3ub(0, 0, 255);

glPushMatrix();

glTranslated(-0.1, -0.1, 0);

DrawSq(0.2, 0.2);

glTranslated(0.0, 0.0, 5.5);

DrawSq(0.2, 0.2);

glPopMatrix();

// Ось направления света

glBegin(GL\_LINES); // Начать рисовать линию

glColor3ub(255, 255, 255);

glVertex3f(0.0, 0.0, 0.0); //gl - использовать OpenGL; Vertex - нарисовать вершину; 3 - входных параметра(х,y,z); f - float тип входных параметров

glVertex3f(0.0, 0.0, 1.0);

glEnd();// Закончить рисовать линию

glEnable(GL\_LIGHTING);

glPopMatrix();

}

void Light4(){

glPushMatrix();

GLfloat pos[] = { 0.0,0.0,0,1.0 }; // Позиция света (х=1 y=1 z=1) 1-в конкретной точке, 0 солнце

GLfloat lightcolor[] = { 1.0,1.0,1.0,1.0 };

GLfloat dir[] = { 0.0,0.0,5 }; // Направление света

GLfloat light\_specular[] = { 0.0,0.0,0.0,1.0 };

glRotatef(Sun, 0.0, 1.0, -1.0);

glTranslated(5, 0, 0);

glRotated(-90, 0, 1, 0);

glLightfv(GL\_LIGHT3, GL\_POSITION, pos); // Установка поззиции источника света

glLightfv(GL\_LIGHT3, GL\_DIFFUSE, lightcolor); // Установка основного света белым(r=1 g=1 b=1 a=1 )

glLightfv(GL\_LIGHT3, GL\_AMBIENT, lightcolor); // Установка основного света белым(r=1 g=1 b=1 a=1 )

// glLightfv(GL\_LIGHT3, GL\_SPECULAR, light\_specular);

glLightfv(GL\_LIGHT3, GL\_SPOT\_DIRECTION, dir);

glLightf(GL\_LIGHT3, GL\_SPOT\_CUTOFF, 50.0);

glDisable(GL\_LIGHTING);

glColor3ub(255, 255, 255);

glPushMatrix();

glTranslated(-0.1, -0.1, 0);

DrawSq(0.2, 0.2);

glTranslated(0.0, 0.0, 5.5);

DrawSq(0.2, 0.2);

glPopMatrix();

// Ось направления света

glBegin(GL\_LINES); // Начать рисовать линию

glColor3ub(255, 255, 255);

glVertex3f(0.0, 0.0, 0.0); //gl - использовать OpenGL; Vertex - нарисовать вершину; 3 - входных параметра(х,y,z); f - float тип входных параметров

glVertex3f(0.0, 0.0, 1.0);

glEnd();// Закончить рисовать линию

glEnable(GL\_LIGHTING);

glPopMatrix();

}

// Функция отрисовки

void display(){

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT); // Очистка буеров цвета и глубины

glLoadIdentity(); // Очистка текущей

// Код отрисовки

glRotatef(45, 1.0, 0.0, 0.0);// Поворот сцены на 45 радусов по х

glRotatef(-45, 0.0, 1.0, 0.0);// Поворот сцены на -45 радусов по у

if (L1)

Light1();

if (L2)

Light2();

if (L3)

Light3();

if (L4)

Light4();

glRotated(spiny, 0, 1, 0);

glRotated(spinx, 1, 0, 0);

glRotated(spinz, 0, 0, 1);

MaterialChange(1, 0, 0, 1, 1, 1, 50, 0, 0, 0);

glColor4ub(106, 32, 99, 255);

glutSolidCube(1);

glEnable(GL\_BLEND); // Включить наложение цветов

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE); // Прозрачность приемника вычислять через альфа компонент. Прозрачность источника оставить без изменений

glDepthMask(GL\_FALSE); // Вырубить тест глубины

glPushMatrix();

glTranslated(-0.5, 0.5, 0.5);

MaterialChange(0, 0, 1, 0, 0, 0, 50, 0, 0, 0);

glColor4ub(255, 0, 0,100);

glutSolidCube(1);

glPopMatrix();

glDepthMask(GL\_TRUE); // Врубить тест глубины

glDisable(GL\_BLEND); // Выключить наложение цветов

glutSwapBuffers(); //Более плавная анимация

glFlush(); // Отрисовка

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DEPTH | GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize(WinLenght, WinHight); // Размер создаваемого окна

glutCreateWindow("Свет"); // Создание окна с именем белый квадрат

Init();

glutDisplayFunc(display); // Использовать функцию display для отрисовки изображения

glutReshapeFunc(reshape); // Использовать функцию reshape при изменении размеров окна

glutKeyboardFunc(keyboard); // Использовать функцию keyboard при использовании клавиатуры

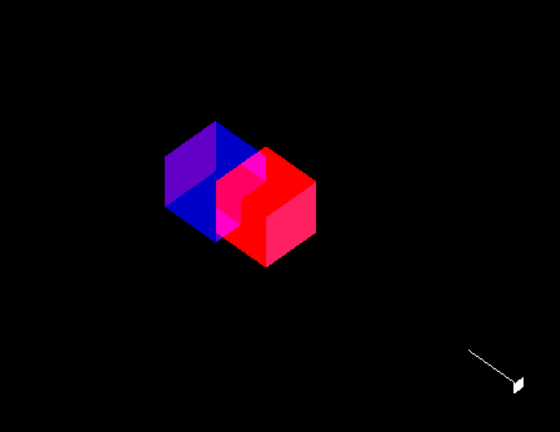
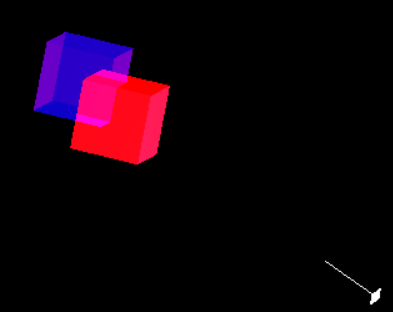
glutMainLoop(); // Зациклить main()

return 0;

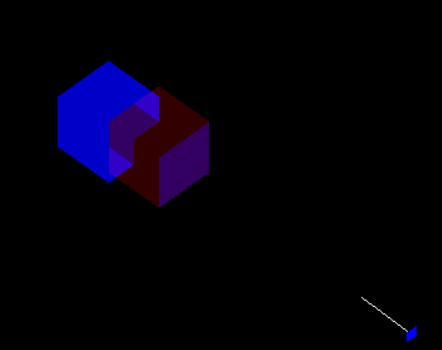
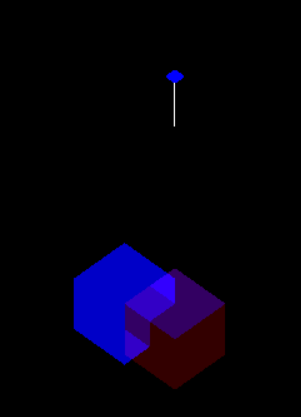
}

**Пример выполнения программы:**

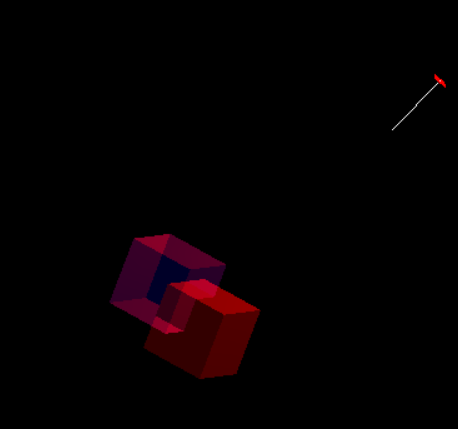
Синий прозрачный куб пересечен с красным не прозрачным. Под белым светом:

Синий прозрачный куб пересечен с красным не прозрачным. Под синим светом:

Синий прозрачный куб пересечен с красным не прозрачным. Под красным светом:

**Выводы:**

В данной лабораторной работе были получены практические навыки по работе с буфером глубины и прозрачностью в OpenGL.