ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| асп. |  |  |  | Д.А. Смолиенко |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2 |
| РАБОТА С ИСТОЧНИКОМ СВЕТА И СВОЙСТВАМИ  МАТЕРИАЛА ПОВЕРХНОСТИ |
| по дисциплине:  КОМПЬЮТЕРНАЯ ГРАФИКА |
|  |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4631 |  |  |  | С.А. Гришин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2018

**Задание на лабораторную работу**

Включить источник света, задать отражающие свойства поверхностей: диффузное, зеркальное отражение, цвет поверхности. Вывести несколько объемных объектов. Каждый из них должен иметь различные свойства поверхностей (доминирующее зеркальное или диффузное отражение, различный цвет внешних поверхностей).

**Текст программы**

#include <GL/glut.h>

#include <stdlib.h>

double rotate\_y = 0;

double rotate\_x = 0;

float mat1\_dif[] = { 0.8, 0.8, 0.0 } float mat1\_amb[] = { 0.5, 0.5, 0.5 } float mat1\_spec[] = { 0.0, 0.0, 0.0 } float mat1\_shininess = 0.5 \* 128

float mat2\_dif[]

= { 0.0, 0.0, 0.8 };

float mat2\_amb[] = { 0.0, 0.0, 0.0 };

float mat2\_spec[] = { 1.0, 1.0, 1.0 };

float mat2\_shininess = 0.7 \* 128;

float mat3\_dif[] = { 0.9, 0.2, 0.0 };

float mat3\_amb[] = { 0.2, 0.2, 0.2 };

float mat3\_spec[] = { 0.8, 0.8, 0.8 };

float mat3\_shininess = 0.1 \* 128;

float mat4\_dif[] = { 2.0, 2.0, 2.0 };

float mat4\_amb[] = { 0.5, 0.5, 0.5 };

float mat4\_spec[] = { 0.0, 0.0, 0.0 };

float mat4\_shininess = { 0.0 };

void init(void)

{

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

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

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

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

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient)

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse)

glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular)

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_DEPTH\_TEST);

}

void display(void)

{

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

glPushMatrix();

glRotatef(20.0, 1.0, 0.0, 0.0);

glRotatef(rotate\_x, 1.0, 0.0, 0.0);

glRotatef(rotate\_y, 0.0, 1.0, 0.0);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_AMBIENT, mat1\_amb);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, mat1\_dif);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SPECULAR, mat1\_spec);

glMaterialf(GL\_FRONT\_AND\_BACK, GL\_SHININESS, mat1\_shininess);

glPushMatrix();

glTranslatef(-0.75, 0.5, 0.0);

glRotatef(90.0, 1.0, 0.0, 0.0);

glutSolidTorus(0.275, 0.85, 50, 50);

glPopMatrix();

glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat2\_amb);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat2\_dif);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat2\_spec);

glMaterialf(GL\_FRONT, GL\_SHININESS, mat2\_shininess);

glPushMatrix();

glTranslatef(-0.75, -0.5, 0.0);

glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat3\_amb);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat3\_dif);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat3\_spec);

glMaterialf(GL\_FRONT, GL\_SHININESS, mat3\_shininess);

glPushMatrix();

glTranslatef(1.5, 0.0, -1.0);

glutSolidSphere(1.0, 50, 50);

glPopMatrix();

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_AMBIENT, mat4\_amb);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, mat4\_dif);

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SPECULAR, mat4\_spec);

glMaterialf(GL\_FRONT\_AND\_BACK, GL\_SHININESS, mat4\_shininess);

glPushMatrix();

glTranslatef(1.0, 2.0, -1.0);

glRotatef(45.0, 1.0, 1.0, 0.0);

glutSolidCube(1.0);

glPopMatrix();

glPopMatrix();

glFlush();

}

void reshape(int w, int h)

{

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

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(

40.0,

(GLfloat)w / h,

1, 100.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(

0.0f, 0.0f, 9.0f,

0.0f, 0.0f, 0.0f,

0.0f, 100.0f, 0.0f);

}

void specialKeys(int key, int x, int y)

{

if (key == GLUT\_KEY\_RIGHT)

rotate\_y += 3;

else if (key == GLUT\_KEY\_LEFT)

rotate\_y -= 3;

else if (key == GLUT\_KEY\_UP)

rotate\_x -= 3;

else if (key == GLUT\_KEY\_DOWN)

rotate\_x += 3;

glutPostRedisplay();

}

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

{

glutInit(&argc, argv);

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

glutInitWindowSize(500, 500);

glutCreateWindow("CG LR2");

init();

glutReshapeFunc(reshape);

glutDisplayFunc(display);

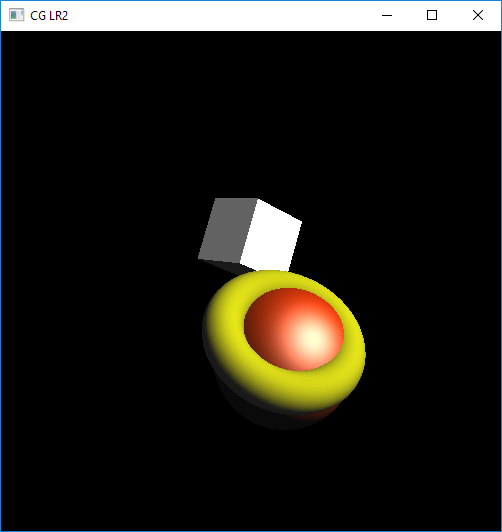
glutSpecialFunc(specialKeys);

glutMainLoop();

return 0;

}

**Результат работы программы**



**Выводы**

В ходе лабораторной работы была разработана программа, выводящая объёмную сцену. Каждому объёмному объекту были заданы свойства поверхностей (доминирующее зеркальное или диффузное отражение, различный цвет внешних поверхностей). Включен источник света.