ЛАБОРАТОРНАЯ РАБОТА №6

По дисциплине «Алгоритмические средства компьютерной графики»

ТЕМА. Пространственные формы и источники света

Цель работы: Попрактиковаться в задании и отображении пространственных форм и установки источников света в OpenGL

Порядок работы:

Задание 1.

1) В среде Visual Studio создайте пустое решение (solution) с именем LW6. В нем создайте консольный проект на языке C++ Win32 Console Application. Назовите его LW6_1.
2) В проект добавьте пустой файл с исходным кодом source.cpp
3) В качестве примера исходного кода программы возьмите код из лабораторной работы №5. Измените исходный код таким образом, что бы Ваши объекты (буквы из предыдущей лабораторной работы) были отображены с помощью полигонов (четырехугольников или треугольников) и представляли собой просранственные формы. Каждой стороне Вашего объекта задайте цвет таким образом, что бы цвета разных сторон отличались. Сохраните возможность управлять изображением с помощью клавиш (из предыдущей лабораторной работы).
4) Используйте функции OpenGL для включения буфера глубины и включения алгоритма удаления нелицевых граней.
5) Запустите приложение и проанализируйте его работу. Покрутите Ваши объекты. Если в отображении объектов была обнаружены ошибки, то исправьте их (обход вершин при задании граней в файле должен быть осуществлен против часовой стрелки для лицевых граней и по часовой стрелки для нелицевых граней).

Задание 2.

1) Создайте в решении LW6 новый консольный проект на языке C++ Win32 Console Application. Назовите его LW6_2.
2) В проект добавьте пустой файл с исходным кодом source.cpp
3) Наберите следующий текст программы:

```
#include <GL/glut.h>
#include <math.h>

#define PI 3.141592653

int light_sample = 1;

void init(void)
{
	glClearColor(0.3, 0.3, 0.3, 0.0);
	// рассчет освещения
	glEnable(GL_LIGHTING);
	// двухсторонний расчет освещения
	glLightModelf(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);
	// автоматическое приведение нормалей к
	// единичной длине
	glEnable(GL_NORMALIZE);
```

```
}

void reshape(int width, int height)
{
	glViewport(0, 0, width, height);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glOrtho(-1.2, 1.2, -1.2, 1.2, -1, 1);
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
}

void display(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	// свойства материала
	GLfloat material_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
	glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, material_diffuse);
	// установка источников света
	switch (light_sample)
	{
	case 1:// направленный источник света
	{GLfloat light0_diffuse[] = { 0.4, 0.7, 0.2 };
	GLfloat light0_direction[] = { 0.0, 0.0, 1.0, 0.0 };
	glEnable(GL_LIGHT0);
	glLightfv(GL_LIGHT0, GL_DIFFUSE, light0_diffuse);
	glLightfv(GL_LIGHT0, GL_POSITION, light0_direction);
	break; }
	case 2:		// точечный источник света
		// убывание интенсивности с расстоянием
		// отключено (по умолчанию)
	{GLfloat light1_diffuse[] = { 0.4, 0.7, 0.2 };
	GLfloat light1_position[] = { 0.0, 0.0, 1.0, 1.0 };
	glEnable(GL_LIGHT1);
	glLightfv(GL_LIGHT1, GL_DIFFUSE, light1_diffuse);
	glLightfv(GL_LIGHT1, GL_POSITION, light1_position);
	break; }
	case 3:		// точечный источник света
		// убывание интенсивности с расстоянием
		// задано функцией f(d) = 1.0 / (0.4 * d * d + 0.2 * d)
	{GLfloat light2_diffuse[] = { 0.4, 0.7, 0.2 };
	GLfloat light2_position[] = { 0.0, 0.0, 1.0, 1.0 };
	glEnable(GL_LIGHT2);
	glLightfv(GL_LIGHT2, GL_DIFFUSE, light2_diffuse);
	glLightfv(GL_LIGHT2, GL_POSITION, light2_position);
	glLightf(GL_LIGHT2, GL_CONSTANT_ATTENUATION, 0.0);
	glLightf(GL_LIGHT2, GL_LINEAR_ATTENUATION, 0.2);
	glLightf(GL_LIGHT2, GL_QUADRATIC_ATTENUATION, 0.4);
	break; }
	case 4:		// прожектор
		// убывание интенсивности с расстоянием
		// отключено (по умолчанию)
		// половина угла при вершине 30 градусов
		// направление на центр плоскости
	{GLfloat light3_diffuse[] = { 0.4, 0.7, 0.2 };
	GLfloat light3_position[] = { 0.0, 0.0, 1.0, 1.0 };
	GLfloat light3_spot_direction[] = { 0.0, 0.0, -1.0 };
	glEnable(GL_LIGHT3);
	glLightfv(GL_LIGHT3, GL_DIFFUSE, light3_diffuse);
	glLightfv(GL_LIGHT3, GL_POSITION, light3_position);
```

```
		glLightf(GL_LIGHT3, GL_SPOT_CUTOFF, 30);
		glLightfv(GL_LIGHT3, GL_SPOT_DIRECTION, light3_spot_direction);
		break; }
		case 5:		// прожектор
			// убывание интенсивности с расстоянием
			// отключено (по умолчанию)
			// половина угла при вершине 30 градусов
			// направление на центр плоскости
			// включен рассчет убывания интенсивности для прожектора
		{GLfloat light4_diffuse[] = { 0.4, 0.7, 0.2 };
		GLfloat light4_position[] = { 0.0, 0.0, 1.0, 1.0 };
		GLfloat light4_spot_direction[] = { 0.0, 0.0, -1.0 };
		glEnable(GL_LIGHT4);
		glLightfv(GL_LIGHT4, GL_DIFFUSE, light4_diffuse);
		glLightfv(GL_LIGHT4, GL_POSITION, light4_position);
		glLightf(GL_LIGHT4, GL_SPOT_CUTOFF, 30);
		glLightfv(GL_LIGHT4, GL_SPOT_DIRECTION, light4_spot_direction);
		glLightf(GL_LIGHT4, GL_SPOT_EXPONENT, 15.0);
		break; }
		case 6:		// несколько источников света
		{GLfloat light5_diffuse[] = { 1.0, 0.0, 0.0 };
		GLfloat light5_position[] = { 0.5 * cos(0.0), 0.5 * sin(0.0), 1.0, 1.0 };
		glEnable(GL_LIGHT5);
		glLightfv(GL_LIGHT5, GL_DIFFUSE, light5_diffuse);
		glLightfv(GL_LIGHT5, GL_POSITION, light5_position);
		glLightf(GL_LIGHT5, GL_CONSTANT_ATTENUATION, 0.0);
		glLightf(GL_LIGHT5, GL_LINEAR_ATTENUATION, 0.4);
		glLightf(GL_LIGHT5, GL_QUADRATIC_ATTENUATION, 0.8);
		GLfloat light6_diffuse[] = { 0.0, 1.0, 0.0 };
		GLfloat light6_position[] = { 0.5 * cos(2 * PI / 3), 0.5 * sin(2 * PI / 3),
1.0, 1.0 };
		glEnable(GL_LIGHT6);
		glLightfv(GL_LIGHT6, GL_DIFFUSE, light6_diffuse);
		glLightfv(GL_LIGHT6, GL_POSITION, light6_position);
		glLightf(GL_LIGHT6, GL_CONSTANT_ATTENUATION, 0.0);
		glLightf(GL_LIGHT6, GL_LINEAR_ATTENUATION, 0.4);
		glLightf(GL_LIGHT6, GL_QUADRATIC_ATTENUATION, 0.8);
		GLfloat light7_diffuse[] = { 0.0, 0.0, 1.0 };
		GLfloat light7_position[] = { 0.5 * cos(4 * PI / 3), 0.5 * sin(4 * PI / 3),
1.0, 1.0 };
		glEnable(GL_LIGHT7);
		glLightfv(GL_LIGHT7, GL_DIFFUSE, light7_diffuse);
		glLightfv(GL_LIGHT7, GL_POSITION, light7_position);
		glLightf(GL_LIGHT7, GL_CONSTANT_ATTENUATION, 0.0);
		glLightf(GL_LIGHT7, GL_LINEAR_ATTENUATION, 0.4);
		glLightf(GL_LIGHT7, GL_QUADRATIC_ATTENUATION, 0.8);
		break; }
		}
		GLfloat x, y;
		glBegin(GL_QUADS);
		glNormal3f(0.0, 0.0, -1.0);
		for (x = -1.0; x < 1.0; x += 0.005)
		{
			for (y = -1.0; y < 1.0; y += 0.005)
			{
				glVertex3f(x, y, 0.0);
				glVertex3f(x, y + 0.005, 0.0);
				glVertex3f(x + 0.005, y + 0.005, 0.0);
				glVertex3f(x + 0.005, y, 0.0);
			}
```

```
        }
        glEnd();
        // отключение источников света
        glDisable(GL_LIGHT0);
        glDisable(GL_LIGHT1);
        glDisable(GL_LIGHT2);
        glDisable(GL_LIGHT3);
        glDisable(GL_LIGHT4);
        glDisable(GL_LIGHT5);
        glDisable(GL_LIGHT6);
        glDisable(GL_LIGHT7);

        glutSwapBuffers();
}

void keyboard_function(unsigned char key, int x, int y)
{
        switch (key)
        {
        case '1':light_sample = 1;break;
        case '2':light_sample = 2; break;
        case '3':light_sample = 3; break;
        case '4':light_sample = 4; break;
        case '5':light_sample = 5; break;
        case '6':light_sample = 6; break;
        }

        glutPostRedisplay();
}

void main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        glutInitWindowPosition(50, 100);
        glutInitWindowSize(500, 500);
        glutCreateWindow("Пример установки источников света в OpenGL");
        init();
        glutDisplayFunc(display);
        glutReshapeFunc(reshape);
        glutKeyboardFunc(keyboard_function);
        glutMainLoop();
    }
```

4) Изучите текст программы.
5) Запустите программу и проанализируйте его работу.
6) Измените значения векторов `light4_diffuse[]`, `light4_position[]`, `light4_spot_direction[]`. Запустите приложение и проанализируйте его работу.

Задание 3.

В тексте программы проекта LW6_1 внесите изменения таким образом, что бы в сцене появился источник цвета. Проведите эксперименты с разными видами источников света и с их параметрами. Проанализируйте работу Вашей программы.