МИНОБРНАУКИ РОССИИ
Санкт-Петербургский государственный электротехнический университет
« ЛЭТИ» им. В. И. Ульянова(Ленина)

Задание для лабораторной работы № 6
" Реализация трехмерного объекта с использованием библиотеки OpenGL "

Преподаватель: Герасимова Т.В.

Санкт-Петербург
2021 г.

Задание

Разработать программу, реализующую представление разработанного вами трехмерного рисунка, используя предложенные функции библиотеки OpenGL (матрицы видового преобразования, проецирование) и язык GLSL.

Разработанная программа должна быть пополнена возможностями остановки интерактивно различных атрибутов через вызов соответствующих элементов интерфейса пользователя, замена типа проекции, управление преобразованиями, как с помощью мыши, так и с помощью диалоговых элементов.

Выполнение работы

Основная задача, которую необходимо решить при выводе трехмерной графической информации, заключается в том, что объекты, описанные в мировых координатах, необходимо изобразить на плоской области вывода экрана, т.е. требуется преобразовать координаты точки из мировых координат (x,y,z) в оконные координаты (X,Y) ее центральной проекции. Это отображение выполняют в несколько этапов.

Первый этап – *видовое преобразование* – преобразование мировых координат в видовые (видовая матрица);

второй этап – *перспективное преобразование* – преобразование видовых координат в усеченные (матрица проекции).

Для того, чтобы активизировать какую-либо матрицу, надо установить текущий режим матрицы, для чего служит команда :

void glMatrixMode (GLenum mode).

Параметр mode определяет, с каким набором матриц будет выполняться последовательность операций, и может принимать одно из трех значений: GL_MODELVIEW, GL_PROJECTION, GL_TEXTURE. Для определения элементов матрицы используются следующие команды: glLoadMatrix, glLoadIdentity.

Преобразование объектов выполняется при помощи следующих операций над матрицами.

***void glRotate[f d]*(GLdouble angle, GLdouble x, GLdouble y, GLdouble z)**

Эта команда рассчитывает матрицу для выполнения вращения вектора против часовой стрелки на угол, определяемый параметром angle, осуществляемого относительно точки (x,y,z). После выполнения этой команды все объекты изображаются повернутыми.

***void glTranslate[f d]*(GLdouble x, GLdouble y, GLdouble z);**

При помощи этой команды осуществляется перенос объекта на расстояние x по оси X, на расстояние y по оси У и на z по оси Z.

Проекции.

Несоответствие между пространственными объектами и плоским изображением устраняется путем введения проекций, которые отображают объекты на двумерной проекционной картинной плоскости. Очень важное значение имеет расстояние между наблюдателем и объектом, поскольку "эффект перспективы" обратно пропорционален этому расстоянию. Таким образом, вид проекции зависит от расстояния между наблюдателем и картинной плоскостью, в зависимости от которой различают два основных класса проекций: *параллельные* и *центральные*. В работе использовалась центральная проекция, а именно перспективная проекция. Для задания проекции использовалась команда

gluPerspective(Gldouble angley,Gldouble aspect,Gldouble znear,Gldouble zfar).

Предполагается, что точка схода имеет координаты (0, 0, 0) в видовой системе координат. Параметр angley задает угол видимости (в градусах) в направлении оси y. В направлении оси x угол видимости задается через отношение сторон aspect, которое обычно определяется отношением сторон области вывода. Два других параметра задают расстояние от наблюдателя (точки схода) до ближней и дальней плоскости отсечения. Ориентация объекта задана при помощи команды

gluLookAp(Gldouble eyex,Gldouble eyey,Gldouble eyez,Gldouble eyex,Gldouble centerx,Gldouble centery,Gldouble centerz,Gldouble upx,Gldouble upy,Gldouble upz)

точка наблюдения задается группой параметров eye, центр сцены – center, верх сцены – up.

```
void CGLlab3View::OnSize(UINT nType, int cx, int cy)
{
        COpenGLView::OnSize(nType, cx, cy);
        // Установка параметров области вывода, проекции
        // и преобразования координат
        glViewport(0, 0, cx, cy);
        GLdouble gldAspect = (GLdouble) cx/ (GLdouble) cy;
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(10.0, gldAspect, 1.0, 100.0);
        // Точка наблюдения
        gluLookAt(6, 8, h, 0, 0, 0, 0, 0, 1);
        glMatrixMode(GL_MODELVIEW);
}
```

Для получения реалистичного изображения используем освещение сцены. Число источников света может достигать восьми. Далее приводится фрагмент программы, обеспечивающий включение источника света и задание его параметров. В этом фрагмента показаны способы задания свойств материала объекта и включение необходимых тестов при отображении трехмерных объектов.

```
void CGLlab3View::OnInitialUpdate()
{
	COpenGLView::OnInitialUpdate();

	// Устанавливаем параметры источника света
	static float ambient[] = {0.0f, 0.0f, 0.0f, 1.0f};
	static float diffuse[] = {1.0, 1.0, 1.0, 1.0};
	static float specular[] = { 1.0, 1.0, 1.0, 1.0 };
	static float position[] = {100.0, 60.0, 150.0, 0.0};

	// Определяем свойства материала лицевой поверхности
	static float front_mat_shininess[] = {10.0};
	static float front_mat_specular[] = {0.5, 0.4, 0.4, 0.1};
	static float front_mat_diffuse[] = {0.0, 0.9, 0.3, 0.1};

	// Определяем свойства материала обратной поверхности
	static float back_mat_shininess[] = {3.2};
	static float back_mat_specular[] = {0.07568, 0.61424, 0.07568, 1.0};
	static float back_mat_diffuse[] = {0.633, 0.727811, 0.633, 1.0};

	static float lmodel_ambient[] = {1.0, 1.0, 0.0, 1.0};
	static float lmodel_twoside[] = {GL_TRUE};

	// Определяем цвет фона используемый по умолчанию
	glClearColor(1.0f, 0.96f, 0.866f, 1.0f);
	// Включаем различные тесты
	glDepthFunc(GL_LEQUAL);
	glEnable(GL_DEPTH_TEST);
	glEnable(GL_BLEND);
	glBlendFunc(GL_ONE, GL_SRC_COLOR);

	// Задаем источник света
	glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
	glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
	glLightfv(GL_LIGHT0, GL_POSITION, position);
	glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lmodel_ambient);
	glLightModelfv(GL_LIGHT_MODEL_TWO_SIDE, lmodel_twoside);

	// Разрешаем освещение и включаем источник света
	glEnable(GL_LIGHTING);
	glEnable(GL_LIGHT0);

	// Устанавливаем параметры материалов
	glMaterialfv(GL_FRONT, GL_SHININESS, front_mat_shininess);
	glMaterialfv(GL_FRONT, GL_SPECULAR, front_mat_specular);
	glMaterialfv(GL_FRONT, GL_DIFFUSE, front_mat_diffuse);
```

```
        glMaterialfv(GL_BACK, GL_SHININESS, back_mat_shininess);
        glMaterialfv(GL_BACK, GL_SPECULAR, back_mat_specular);
        glMaterialfv(GL_BACK, GL_DIFFUSE, back_mat_diffuse);

}
```

Задания

Каркасные объекты

Задание

Написать программу, рисующую проекцию трехмерного каркасного объекта.

Требования

1 Грани объекта рисуются с помощью доступных функций рисования отрезка в координатах окна. При этом использовать шейдеры GLSL и OpenGL
2 Вывод многогранника с удалением или прорисовкой невидимых граней;
3 Ортогональное и перспективное проецирование;
4 перемещения, повороты и масштабирование многогранника по каждой из осей независимо от остальных.
5 Генерация многогранника с заданной мелкостью разбиения.
6 Д.б. установлено изменение свойств источника света (интенсивность).
7 При запуске программы объект сразу должно быть хорошо виден.
8 Пользователь имеет возможность вращать фигуру (2 степени свободы) и изменять параметры фигуры.
9 Возможно изменять положение наблюдателя.
10 Нарисовать оси системы координат.
11 Все варианты требований могут быть выбраны интерактивно.

Пример задания.