Отчет по лабораторной работе №4 по курсу «компьютерная графика»
Студент группы 08-306: Баскаков О.А.
Преподаватель: Измайлов А.А.
Работа выполнена: "25" ноября 2009 г.

Тема: Ознакомление с библиотеками трехмерной графики. Создание простой программы с трехмерной графикой.

Цель работы

Используя результаты л. р. №3, изобразить заданное тело (то же, что и в л. р. №3) с использованием средств библиотеки трехмерной графики (предпочтительно OpenGL) без применения готовых структур для хранения тел. Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей.

Задача

Вариант 1: Прямой усеченный элиптический конус.

Оборудование

Ноутбук с предустновленным QT Creator, видеокарта ATI.

Теоретическая часть

Принято решение писать данную лабораторную работу при помощи OpenGL.

OpenGL представляет из себя интерфейс программирования трехмерной графики. Используя OpenGL, можно с легкостью создать трехмерные поверхности, наложить на них текстуры, осветить источниками света, сделать эффект тумана, прозрачности, а также можно наложить трафарет, передвигать объекты сцены, лампы и камеры по заданным траекториям, сделав, тем самым, анимацию, не задумываясь о деталях реализации данных алгоритмов. OpenGL непосредственно не поддерживает работу с устройствами ввода, такими как мышь или клавиатура, т.к. эта библиотека является платформенно независимой, но эти сложности возможно избежать, используя Qt.

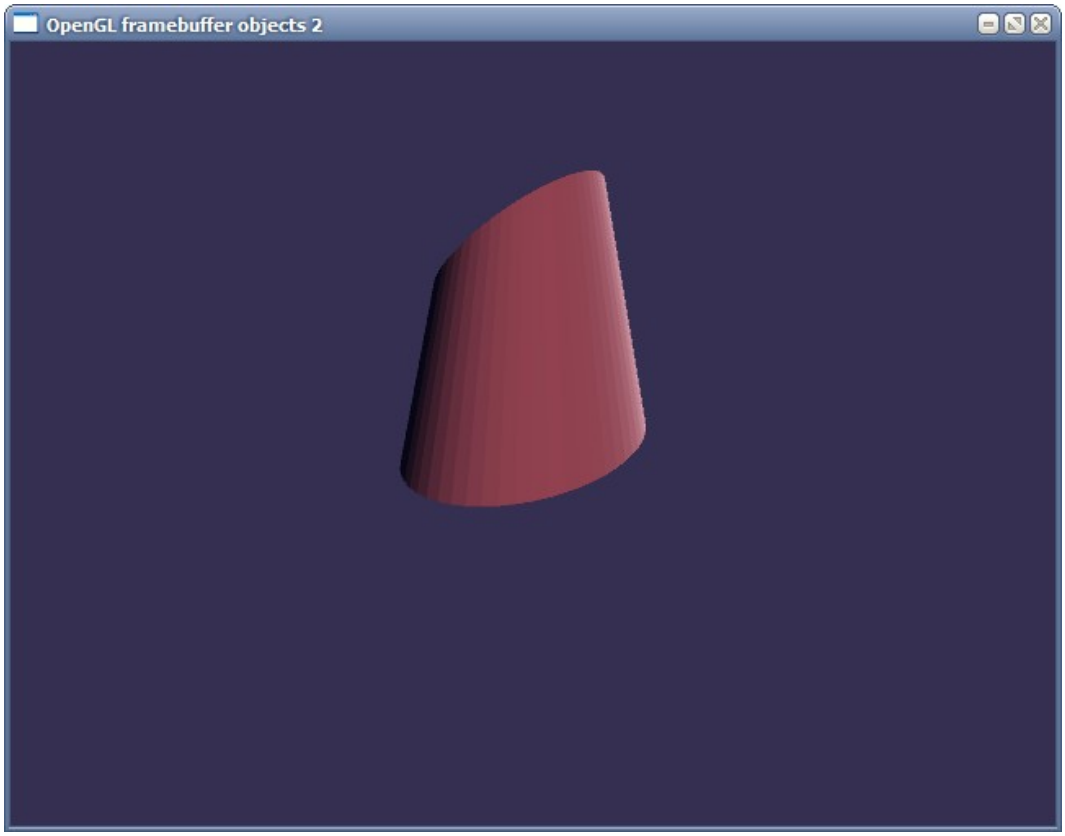
Ход решения

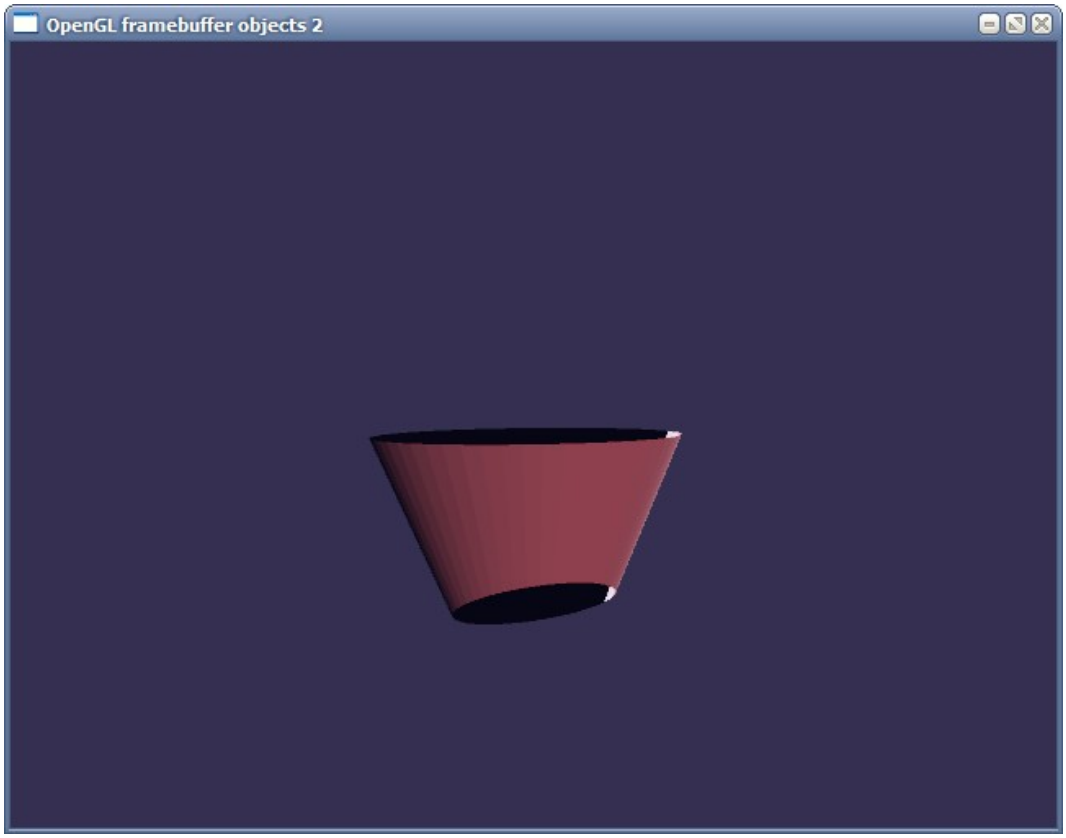
Определим систему координат, в которой будет отрисован объект. Для этого инициализируем матрицу преобразования и задаём объём отсечения при помощи ортографической проекции `glOrtho()`. Также инициализируем модельно-видовую матрицу. Для добавления фигуры на сцену используется два вспомогательных массива: вершин и соответствующих им нормалей. Испльзуются списки отображения, в добавляется тело процедурой из прошлой лабораторной раблты. При отрисовке фигуры необходимо произвести модельно-видовые преобразования, над соответствующей матрицей (повороты вокруг координатных осей на заданные углы).

Для закраски фигуры необходимо добавить на сцену источник света, задав его характеристики (интенсивности фонового, рассеивающего и зеркального света), добавить свойства материала фигуры и включить режим освещения.

Перегружаем события мыши для управления положением фигурой и её масштабированием (колёсико). Для большей наглядности программы добавим в пользовательский интерфейс возможность изменять аппроксимацию фигуры, положение источника света, а также свойств материала фигуры.

Работа программы:





Исходный код

Для подготовки работы были выполнены следующие исходные файлы:

```
void GLWidget::initializeGL()
{

        glEnable(GL_DEPTH_TEST);

        glClearColor(0.21, 0.19, 0.32, 1.0);
        glShadeModel(GL_FLAT);

        GLfloat AmbientColor[] = { 30.0f/256.0, 30.0f/256.0,
100.0f/256.0, 1.0f };
        GLfloat DiffuseColor[] = { 0.7f, 0.3f, 0.3f, 1.0f };
        GLfloat SpecularColor[] = { 1.0f, 1.0f, 1.0f, 1.0f };
        GLfloat Shininess = 6.0f;


        glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, AmbientColor);
        glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, DiffuseColor);
        glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, SpecularColor);
        glMaterialf(GL_FRONT, GL_SHININESS, Shininess);
        GLfloat L_A = 1.0;

        GLfloat AmbientLight[] =  { 0.8f,0.8f, 0.8f, 1.0f };
        GLfloat DiffuseLight[] =  { 0.8f,0.8f, 0.8f, 1.0f };
        GLfloat SpecularLight[] = { 1.0f,1.0f, 1.0f, 1.0f };
        GLfloat LightPosition[] = { -300.0f, 0.0f, -600.0f, 0.0f };

        glLightfv(GL_LIGHT0, GL_SPECULAR, SpecularLight);
        glLightfv(GL_LIGHT0, GL_DIFFUSE, DiffuseLight);
        glLightfv(GL_LIGHT0, GL_POSITION, LightPosition);
        glLightfv(GL_LIGHT0, GL_LINEAR_ATTENUATION, &L_A );
        glLightfv(GL_LIGHT0, GL_QUADRATIC_ATTENUATION,&L_A);
        glLightfv(GL_LIGHT0, GL_CONSTANT_ATTENUATION, &L_A );
       glEnable(GL_LIGHTING);

       glEnable(GL_LIGHT0);



    glEnable(GL_LIGHTING);

    glDisable(GL_CULL_FACE);
        glFrontFace(GL_CCW);


    m_nPyramid = createPyramid( 1.2f);
}
```

```
void GLWidget::resizeGL(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
//    glOrtho(0, w, h, 0, -1, 1);
    glFrustum(-1.0, 1.0, -1.0, 1.0, 1.0, 10.0);

    //set Orto Matrix OLOLOLO!!!
}
vertex GLWidget::Normal(vertex p1,vertex p2,vertex p3)
{
  vertex n;
    double x1,y1,z1,x2,y2,z2,m;
  x1 = p1.x-p2.x;
  y1 = p1.y-p2.y;
  z1 = p1.z-p2.z;
  x2 = p1.x-p3.x;
  y2 = p1.y-p3.y;
  z2 = p1.z-p3.z;
 n.x = (y1*z2-y2*z1);
 n.y = -(x1*z2-z1*x2);
 n.z = x1*y2-y1*x2;
 m = sqrt(n.x*n.x + n.y*n.y + n.z*n.z);
 n.x /= -m;
 n.y /= -m;
 n.z /= -m;
 return n;
}

void GLWidget::paintGL()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0, 0.0, -5.0);
    glRotatef( rotX, 1.0, 0.0, 0.0);
    glRotatef( rotY, 0.0, 1.0, 0.0);
    glCallList(m_nPyramid);
    drawMe();
}
```

Замечания

Не реализовано текстурирование фигуры. Грамотная работа с мышью.

Выводы

Были изучены основные возможности библиотеки OpenGL, а именно использование примитивов, различные модельно-видовые и проекционные преобразования, использование освещения.