Отчет по лабораторной работе №2 по курсу «компьютерная графика»
Студент группы 08-306: Баскаков О.А.
Преподаватель: Измайлов А.А.
Работа выполнена: "14" октября 2009 г.

Тема: Каркасная визуализация выпуклого многоугольника.

Цель работы

Разработать формат представления многогранника и процедуру его каркасной отрисовки в ортографической и изометрических проекциях. Обеспечить удаление невидимых линий и возможность пространственных поворотов и масштабирования многоугольника. Обеспечить автоматическое центрирование и изменение размеров изображения при изменении размеров окна.

Задача

Вариант 1: Куб.

Оборудование

Ноутбук с предустановленным QT Creator, видеокарта ATI.

Теоретическая часть

Каркасная визуализация фигуры представляет собой аппроксимацию фигуры выпуклыми многоугольниками (чаще треугольниками или четырехугольниками). Чем выше аппроксимация, тем ближе к своей математической модели выглядит фигура. Но для того чтобы изображение фигуры выглядело более реалистично, необходимо удалить невидимые линии. Чтобы понять, нужно отрисовывать грань или нет, необходимо найти нормаль к ней, причем ту, которая направлена наружу от фигуры. Нормаль можно найти как векторное произведение двух сторон этой грани. Если координата z вектора нормали положительна, то грань видна, иначе нет.

Ход решения

Создадим класс 3-х мерной вершины Vertex, с поддержкой следующих операций: поворот вокруг координатных осей, масштабирование, перенос, сложение, скалярное и векторное произведения. В файле martrix.h объявлены функции для перемножения векторов и матриц, для задания тождественного, преобразования поворота и прочее.

Основной модуль занимается отправлением куба на «софтверный конвеер», те берет вершины тела из массива, собирает в полигоны, проверяет видимость, умножает на матрицу композиции преобразований и отрисовывает на экране определенными стилями пера видимые и невидимые грани. Видимость грани определяется при помощи нахождения нормали к ней и проверки её координаты глубины на отрицательность.

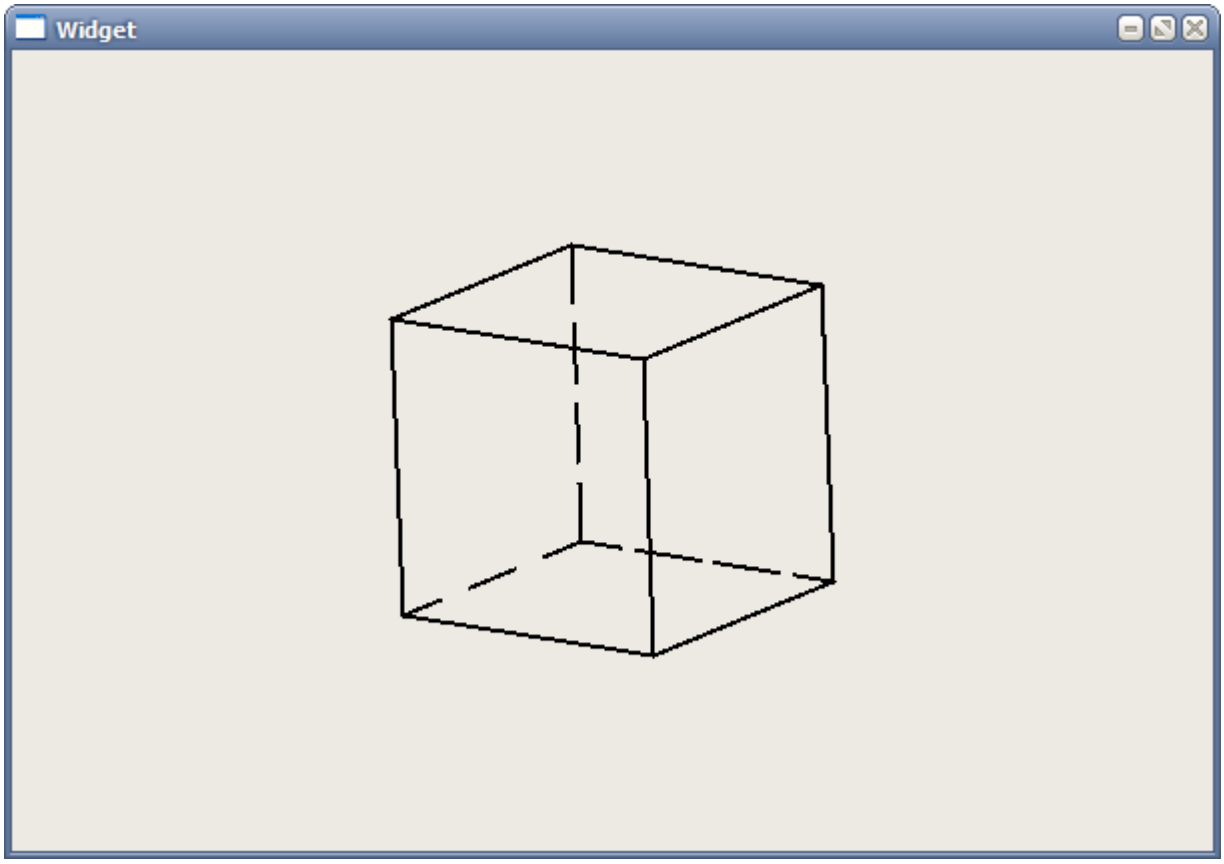
Вершины куба хранятся в константном массиве, где заданны их координаты.

В качестве параметров в этот кланисс я передаю 4 вершины четырехугольника, а точнее номера ячеек, в которых о описаны.

В качестве параметров для генерации матриц поворота функции *mouseMoveEvent* передают относительное смещение курсора мыши по осям X и Y. Масштабирование привязано к колесику мышки.

Изометрическая проекция реализуется:

```
void Vector::izometr(int c, double iz){
if(a[c].z<0) {
a[c].x=a[c].x*iz/(iz-a[c].z);
a[c].y=a[c].y*iz/(iz-a[c].z);
a[c].z=a[c].z*iz/5;
}
}
```



Исходный код

Для подготовки работы были выполнены следующие исходные файлы:

```
typedef float Matrix[4][4];
typedef float Point[4];

typedef struct Vector {
            float x;
            float y;
            float z;
            Vector(){};
            Vector (float a, float b, float c);
            Vector(Point p1, Point p2);
        } Vector;
void Point_copy(float dst[4], const float src[4]);
void Point_mul(Point res,const Matrix M, const Point src);
float Point_scalar(Point p1, Point p2);
void Point_minus(Point res, Point p1, Point p2);
void Matrix4f_set_E(float M[4][4]);
void Matrix4f_set_scale(Matrix M, float scale);
void Matrix4f_set_rotX(Matrix M, float rotX);
void Matrix4f_set_rotY(Matrix M, float rotateY);
void Matrix4f_set_rotZ(Matrix M, float rotZ);
void Matrix4f_new(float M[4][4]);
void Matrix4f_mult(Matrix M1, Matrix M2);

#ifndef MATRIX_H
#define MATRIX_H
#include <math.h>
void Widget::paintEvent(QPaintEvent *event)
```

```
{
    QPainter p(this);
int i, j;
float dot[4];
    float ox, oy;
    ox=this->width()/2;
    oy=this->height()/2;

    for (i=0;i<8;i++) // Foreach vertex
    {
        Point_copy(Vert[i], Ver[i]);
        Point_mul(dot, Glob, Ver[i]);
        Point_copy(Vert[i], dot);
    }

    for(i=0;i<6;i++) // Foreach polygon
    {
        // Is visible?
        Point p1,p2;
        Point_minus( p1, Vert[Poly[i][1]],Vert[Poly[i][0]]);
        Point_minus( p2, Vert[Poly[i][2]],Vert[Poly[i][1]]);

        //if (Point_vec(p1,p2)<0)
        if ((p1[0]*p2[1]-p1[1]*p2[0])>0)
        {
        //setPen punktir
            p.setPen(dashPen);
        }
        else
        {
        //setPen sploshnoy
            p.setPen(solidPen);
        }

       for(j=0;j<4;j++) //Foreach point
        {
  p.drawLine(ox+Vert[Poly[i][j%4]][0], oy-Vert[Poly[i][j%4]][1],
  ox+Vert[Poly[i][(j+1)%4]][0], oy-Vert[Poly[i][(j+1)%4]][1]);
        }
    }
}
```

Замечания

Программа использует умное вращение через матрицы, а не глобальные переменные. Произведена интеграция с мышью для вращения и масштабирования тела. Изометрическая проекция не проработана.

Выводы

В ходе решения данной лабораторной работы был написан простейший софтверный графический движок, получены навыки построения и визуализации трехмерных объектов, что, несомненно, пригодится для выполнения следующих лабораторных работ.