РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

дисциплина: Компьютерная графика

Студент: Мухамедияр Адиль

Группа: НКНбд-01-20

МОСКВА 2022г.

Задание

Задание: написать компьютерную программу для построения изображения трехмерного тела в виде каркасной модели. Программа должна соответствовать следующим требованиям:

- Загружать данные из файла описания объекта типа .dat, состоящего из двух разделов – описание координат вершин и описание ребер. Файл может содержать данные о числе вершин и числе ребер;

- Должен быть реализован объектно-ориентированный подход, моделирующий основные элементы объекта;

- Должна быть возможность изменять положение точки наблюдения;

- Должны быть построены изображения куба (тестовое изображение) и объекта, спроектированного автором программы.

Теоретическая справка

Для выполнения данной лабораторной работы необходимо написать программу, которая будет читать из файла координаты в мировой системе координат, будет читать номера вершин координат, между которыми нужно будет провести грани, будет рисовать каркасную модель на экране с помощью полученных данных.

Для построения каркасной модели трехмерного объекта, заданного мировыми координатами[1], необходимо мировые координаты преобразовать в видовые координаты[2], сделать перспективное преобразование[3], затем, координаты, которые мы получили, преобразовать в экранные координаты[4].

[1] Мировая система координат – система координат, описывающая реальное положение объекта в пространстве.

[2] Для изображения объекта на экране его мировые координаты необходимо преобразовать (пересчитать) в другую систему координат, которая связана с точкой наблюдения. Эта система координат называется видовой системой координат и является левосторонней.

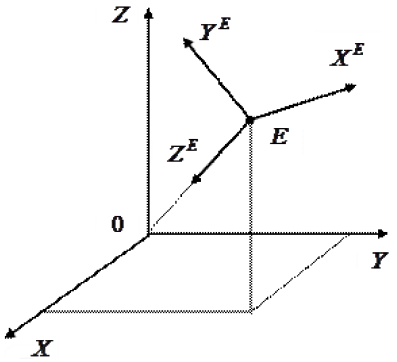

Рисунок1Система видовых координат

Ход решения

```
#include "graphics.h"
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <iostream>
#include <fstream>
using namespace std;

class Point // Точка
{
    private:
        double a;
        double b;
        double c;
    public:
        Point()
        {
            a = 0;
            b = 0;
            c = 0;
        }
        Point(double d, double e, double f)
        {
            a = d;
            b = e;
            c = f;
        }
        double getA(){return a;}
        double getB(){return b;}
        double getC(){return c;}
        void setA(double A){a=A;}
        void setB(double B){b=B;}
        void setC(double C){c=C;}
};


class Vertex // Вершины
{
    public:

        Point l; // Точка с мировыми координатами вершины
        Point k; // Точка с видовыми координатами вершины

    int n; // Номер точки
    // Конструктор
    Vertex(double ro, double teta, double fi, int n = 0)
    {
        Point p(ro, teta, fi);
        l = p;
    }
```

```
    Vertex()
    {
        Point p(0,0,0);
        l = p;
    }

    void setViewCoord(Point coor_spher)
    {
        double s1=sin(coor_spher.getB());
        double s2=sin(coor_spher.getC());
        double s3=cos(coor_spher.getB());
        double s4=cos(coor_spher.getC());
        k.setC(l.getA()*(-s1*s4)+l.getB()*(-s1*s2)+l.getC()*(-s3)+coor_spher.getA());
        k.setB(l.getA()*(s3*s4)+l.getB()*(-s3*s2)+l.getC()*s1);
        k.setA(l.getA()*(-s2)+l.getB()*s4);
    }

    Vertex *next;

};


class Edge {
    public:
        Vertex t;
        Vertex v;

    Edge(Vertex a, Vertex b) // Конструктор
    {
        t.k.setA(a.k.getA());
        v.k.setA(b.k.getA());
        t.k.setB(a.k.getB());
        v.k.setB(b.k.getB());
        t.k.setC(a.k.getC());
        v.k.setC(b.k.getC());
    }
```

```
    void drawEdge(double y_up, double y_down, double x_left, double x_right, double uu, double dd, double ll, double rr, double r2) // Соединяет точки линиями

        double x,y,x1,y1;
        int xx,yy,xx_1,yy_1;

        x=(r2/(2.0*(t.k.getC())))*t.k.getA();
        x1=(r2/(2.0*(v.k.getC())))*v.k.getA();
        xx=(int)(rr+((x-x_right)*(ll-rr))/(x_left-x_right));
        xx_1=(int) (rr+((x1-x_right)*(ll-rr))/(x_left-x_right));
        y=(r2/(2.0*(t.k.getC())))*t.k.getB();
        y1=(r2/(2.0*(v.k.getC())))*v.k.getB();
        yy=(int)(uu+((y-y_up)*(dd-uu))/(y_down-y_up));
        yy_1=(int) (uu+((y1-y_up)*(dd-uu))/(y_down-y_up));
        line(xx,yy,xx_1,yy_1);

    }
    Edge *next;

};
```

```
class Surface {
    public:
    friend class Edge;
        Vertex *vv;
        Edge *ee;
        Point vp;
        double y_up,y_down,x_left,x_right,uu,dd,ll,rr;
        int n;

    void file()// Считываем файл
    {
        int x,y,z,w,*m,*ml;
        ifstream f("coords.txt");
        f>>n;
        f>>w;
        vv=new Vertex[w];
        for(int i=0;i<w;i++)
        {
            f>>x;f>>y;f>>z;
            vv[i]=Vertex(x,y,z);
            vv[i].setViewCoord(vp);
        }
        m=new int[2*n];
        for(int i=0;i<2*n;i++)
            f>>m[i];

        ee=new Edge[n];
        for(int i=0;i<n;i++)
            ee[i]=Edge(vv[m[i*2]],vv[m[i*2+1]]);

        ml=new int[w*3];

        for(int i=0;i<w*3;i++)
            f>>ml[i];

        f.close();

    }
    // Сфера: ro,tetta, fi
    void setViewPoint(double ro, double tetta, double fi) // Считывает координаты точек
    {
        vp.setA(ro);
        vp.setB(tetta);
        vp.setC(fi);
    }
```

```
    void drawSurface() // Рисует по считанным координатам
    {
        for (int i=0;i<n;i++)
            for(int j=0;j<4;j++)
                if(t[j].flag==1)
                {
                    if((e[i].t==t[j].a||e[i].t==t[j].b||e[i].t==t[j].c)&&(e[i].v==t[j].a||e[i].v==t[j].b||e[i].v==t[j].c))
                    {
                        e[i].drawEdge(y_up,y_down,x_left,x_right,uu,dd,ll,rr,vp.getA());
                    }
                }


    }
};



int main()
{
    int gddriver = DETECT, gmode, errorcode;
    initgraph(&gddriver, &gmode, "");

    Surface sss;
    sss.file();
    sss.setViewPoint(10,M_PI/2,1.6);
    sss.drawSurface();
    getch();
    closegraph;
    return 0;
}
```

Исполнение программы