**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

Факультет физико-математических и естественных наук

ОТЧЕТ

по лабораторной работе № 3

дисциплина: Компьютерная графика

Студент: Дорук Мерич Каймакджыоглу

Группа: НКНбд-01-20

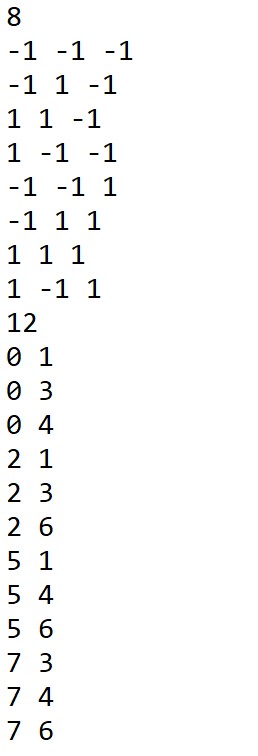
**МОСКВА**

2022 г.

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

**Ход работы:**

- 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.

*class* tck\_view

{

*public:*

*double* x, y, z;

        tck\_view()

        {

            x = 0; y = 0; z = 0;

        }

        tck\_view(*double* *xa*, *double* *yb*, *double* *zc*)

        {

            x = xa; y = yb; z = zc;

        }

};

*class* vertex

{

*public:*

        tck\_view worldcord;

        tck\_view vidovcord;

        vertex(){};

*void* classvertex(tck\_view *p*)

        {

            worldcord = p;

        }

        ~vertex(){};

*void* View\_Coord\_Fun(tck\_view *mircoord*)

        {

*double* roangle = mircoord.x, tetangle=mircoord.y, fiangle=mircoord.z;

            vidovcord.x = -worldcord.x \* sin(tetangle) + worldcord.y \* cos(tetangle);

            vidovcord.y = -worldcord.x \* cos(fiangle) \* cos(tetangle) - worldcord.y \* cos(fiangle) \* sin(tetangle) + worldcord.z \* sin(fiangle);

            vidovcord.z = -worldcord.x \* sin(fiangle) \* cos(tetangle) - worldcord.y \* sin(fiangle) \* sin(tetangle) - worldcord.z \* cos(fiangle) + roangle;

        }

};

*class* edge

{

*private:*

        vertex tmpvr, frvert;

*public:*

        edge(){};

*void* edgech (vertex *sv*, vertex *fv*)

        {

            tmpvr = sv;

            frvert = fv;

        }

        ~edge(){};

*void* draw\_edge\_fun(*double* *d*)

        {

            d = d/2;

*double* x1 = (*int*)((d\*tmpvr.vidovcord.x)/tmpvr.vidovcord.z);

*double* y1 = (*int*)((d\*tmpvr.vidovcord.y)/tmpvr.vidovcord.z);

*double* x2 = (*int*)((d\*frvert.vidovcord.x)/frvert.vidovcord.z);

*double* y2 = (*int*)((d\*frvert.vidovcord.y)/frvert.vidovcord.z);

            line(x1+(getmaxx()/2), y1+(getmaxy()/2), x2+(getmaxx()/2), y2+(getmaxy()/2));

        }

};

*struct* strk{

*public:*

*int* xa, yb;

};

*class* draw\_class

{

*private:*

        vertex \*verts;

        edge \*edges;

        strk \*strk1;

        tck\_view viewpoint;

*bool* prov = 0;

*public:*

*int* n, m;

        friend *class* edge;

        friend *class* vertex;

*void* draw()

        {

            for (*int* i=0; i<m; i++)

                edges[i].draw\_edge\_fun(viewpoint.x);

        }

*void* drawing()

        {

*int* xa, yb, zc;

            if(!prov)

            {

                FILE \*f;

                f = fopen("kub.txt", "r");

                if (f != NULL)

                {

                    fscanf(f, "%d", &n);

                    verts = new vertex[n];

                    for (*int* i=0; i<n; i++)

                    {

                        fscanf(f, "%d", &xa);

                        fscanf(f, "%d", &yb);

                        fscanf(f, "%d", &zc);

                        tck\_view p = tck\_view(xa\*100, yb\*100, zc\*100);

                        verts[i].classvertex(p);

                        verts[i].View\_Coord\_Fun(viewpoint);

                    }

                    fscanf(f, "%d", &m);

                    edges = new edge[m];

                    strk1 = new strk[m];

                    for (*int* i=0; i<m; i++)

                    {

                        fscanf(f, "%d", &xa);

                        fscanf(f, "%d", &yb);

                        strk1[i].xa = xa;

                        strk1[i].yb = yb;

                        edges[i].edgech (verts[strk1[i].xa], verts[strk1[i].yb]);

                    }

                    prov = 1;

                }

            }

            for (*int* i=0; i<n; i++)

                verts[i].View\_Coord\_Fun(viewpoint);

            for (*int* i=0; i<m; i++)

                edges[i].edgech (verts[strk1[i].xa], verts[strk1[i].yb]);

        }

*void* setviewpoint(*double* *roangle*, *double* *tetangle*, *double* *fiangle*)

        {

            viewpoint.x = roangle;

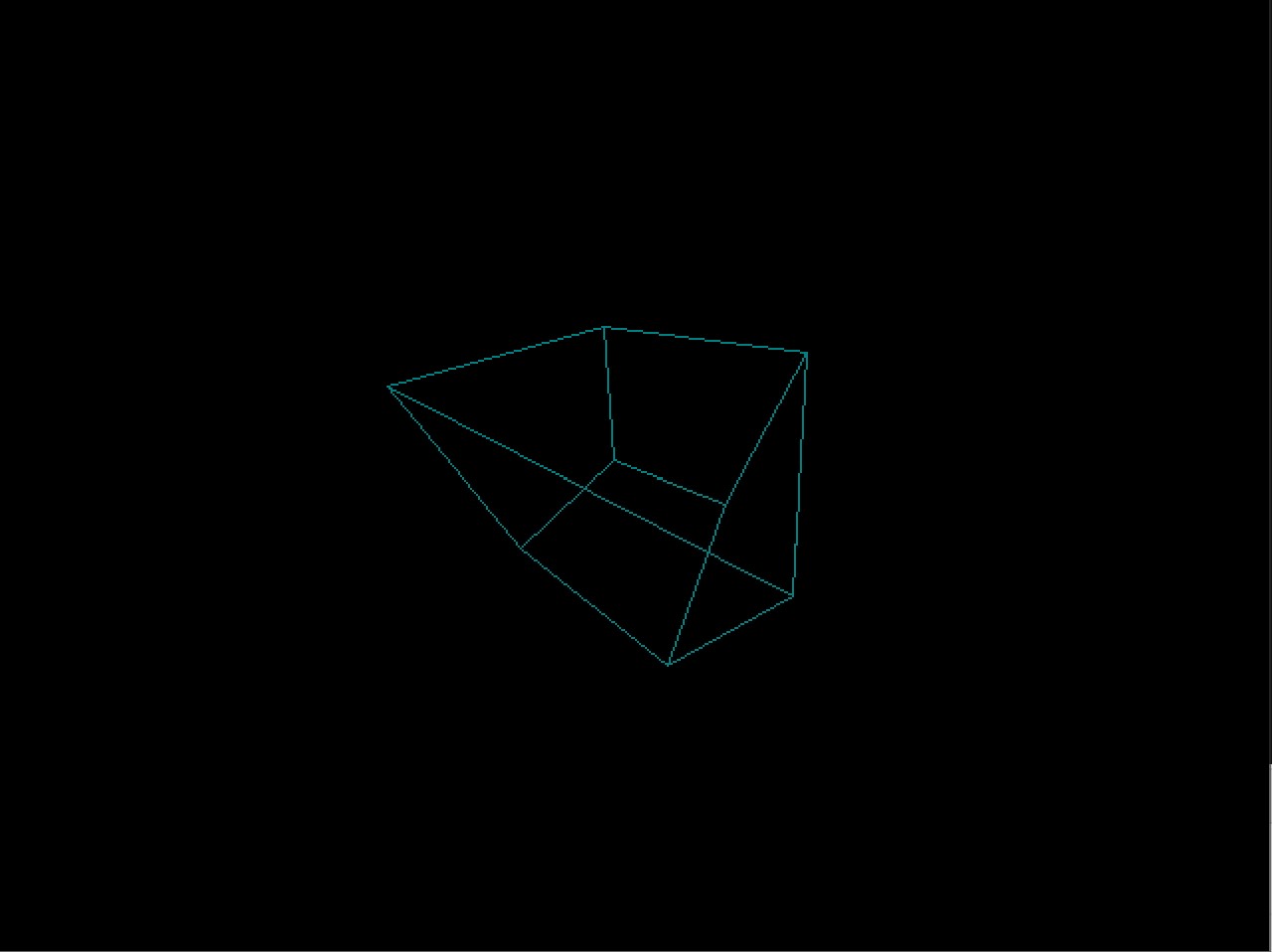
            viewpoint.y = tetangle;

            viewpoint.z = fiangle;

        }

};

**Вывод:**

****

**КОД:**

#include "graphics.h"

#include <stdlib.h>

#include <conio.h>

#include <math.h>

#include <direct.h>

#include <fstream>

#include <iostream>

#include <stdio.h>

using *namespace* std;

*class* tck\_view

{

*public:*

*double* x, y, z;

        tck\_view()

        {

            x = 0; y = 0; z = 0;

        }

        tck\_view(*double* *xa*, *double* *yb*, *double* *zc*)

        {

            x = xa; y = yb; z = zc;

        }

};

*class* vertex

{

*public:*

        tck\_view worldcord;

        tck\_view vidovcord;

        vertex(){};

*void* classvertex(tck\_view *p*)

        {

            worldcord = p;

        }

        ~vertex(){};

*void* View\_Coord\_Fun(tck\_view *mircoord*)

        {

*double* roangle = mircoord.x, tetangle=mircoord.y, fiangle=mircoord.z;

            vidovcord.x = -worldcord.x \* sin(tetangle) + worldcord.y \* cos(tetangle);

            vidovcord.y = -worldcord.x \* cos(fiangle) \* cos(tetangle) - worldcord.y \* cos(fiangle) \* sin(tetangle) + worldcord.z \* sin(fiangle);

            vidovcord.z = -worldcord.x \* sin(fiangle) \* cos(tetangle) - worldcord.y \* sin(fiangle) \* sin(tetangle) - worldcord.z \* cos(fiangle) + roangle;

        }

};

*class* edge

{

*private:*

        vertex tmpvr, frvert;

*public:*

        edge(){};

*void* edgech (vertex *sv*, vertex *fv*)

        {

            tmpvr = sv;

            frvert = fv;

        }

        ~edge(){};

*void* draw\_edge\_fun(*double* *d*)

        {

            d = d/2;

*double* x1 = (*int*)((d\*tmpvr.vidovcord.x)/tmpvr.vidovcord.z);

*double* y1 = (*int*)((d\*tmpvr.vidovcord.y)/tmpvr.vidovcord.z);

*double* x2 = (*int*)((d\*frvert.vidovcord.x)/frvert.vidovcord.z);

*double* y2 = (*int*)((d\*frvert.vidovcord.y)/frvert.vidovcord.z);

            line(x1+(getmaxx()/2), y1+(getmaxy()/2), x2+(getmaxx()/2), y2+(getmaxy()/2));

        }

};

*struct* strk{

*public:*

*int* xa, yb;

};

*class* draw\_class

{

*private:*

        vertex \*verts;

        edge \*edges;

        strk \*strk1;

        tck\_view viewpoint;

*bool* prov = 0;

*public:*

*int* n, m;

        friend *class* edge;

        friend *class* vertex;

*void* draw()

        {

            for (*int* i=0; i<m; i++)

                edges[i].draw\_edge\_fun(viewpoint.x);

        }

*void* drawing()

        {

*int* xa, yb, zc;

            if(!prov)

            {

                FILE \*f;

                f = fopen("kub.txt", "r");

                if (f != NULL)

                {

                    fscanf(f, "%d", &n);

                    verts = new vertex[n];

                    for (*int* i=0; i<n; i++)

                    {

                        fscanf(f, "%d", &xa);

                        fscanf(f, "%d", &yb);

                        fscanf(f, "%d", &zc);

                        tck\_view p = tck\_view(xa\*100, yb\*100, zc\*100);

                        verts[i].classvertex(p);

                        verts[i].View\_Coord\_Fun(viewpoint);

                    }

                    fscanf(f, "%d", &m);

                    edges = new edge[m];

                    strk1 = new strk[m];

                    for (*int* i=0; i<m; i++)

                    {

                        fscanf(f, "%d", &xa);

                        fscanf(f, "%d", &yb);

                        strk1[i].xa = xa;

                        strk1[i].yb = yb;

                        edges[i].edgech (verts[strk1[i].xa], verts[strk1[i].yb]);

                    }

                    prov = 1;

                }

            }

            for (*int* i=0; i<n; i++)

                verts[i].View\_Coord\_Fun(viewpoint);

            for (*int* i=0; i<m; i++)

                edges[i].edgech (verts[strk1[i].xa], verts[strk1[i].yb]);

        }

*void* setviewpoint(*double* *roangle*, *double* *tetangle*, *double* *fiangle*)

        {

            viewpoint.x = roangle;

            viewpoint.y = tetangle;

            viewpoint.z = fiangle;

        }

};

*int* main()

{

*double* rott=210, tetangle=45, fiangle=150;

*int* d;

*int* gddriver = DETECT, gmode, errorcode;

    initgraph(&gddriver, &gmode, (*char*\*)"");

    draw\_class cube;

    cube.setviewpoint(rott, tetangle, fiangle);

    cube.drawing();

    cube.draw();

*int* per;

    while (!kbhit())

    {

        cleardevice();

        setbkcolor(8);

        setcolor(3);

        if(per=='a')

        tetangle += ((M\_PI/3));

        if(per=='d')

        tetangle -= ((M\_PI/3));

        if(per=='w')

        fiangle += ((M\_PI/3));

        if(per=='s')

        fiangle -= ((M\_PI/3));

        if(per=='e')

        rott +=3;

        if(per=='q')

        rott -=3;

        if(per==27)

        break;

        cube.setviewpoint(rott, tetangle, fiangle);

        cube.drawing();

        cube.draw();

        delay(250);

        per = getch();

    }

    getch();

    closegraph();

    return 0;

}