Дисциплина: Компьютерная графика

Лабораторная работа №3

Изображение в виде каркасной модели

Выполнила: Маслова Анастасия Сергеевна

№ Студ.билета: 1032216455

Группа: НКНбд-01-21

**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;

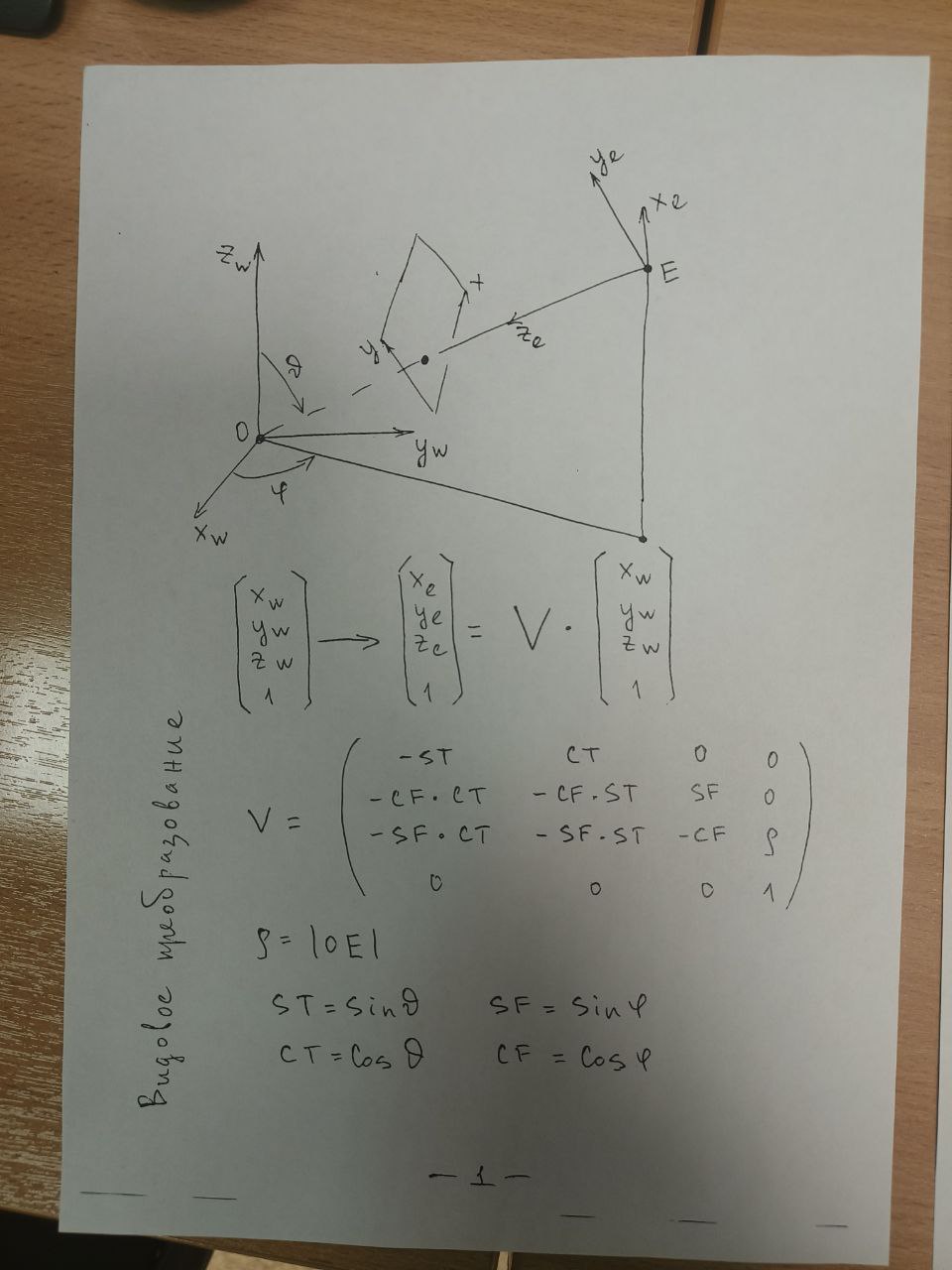
- Должен быть реализован объектно-ориентированный подход, моделирующий основные элементы объекта;

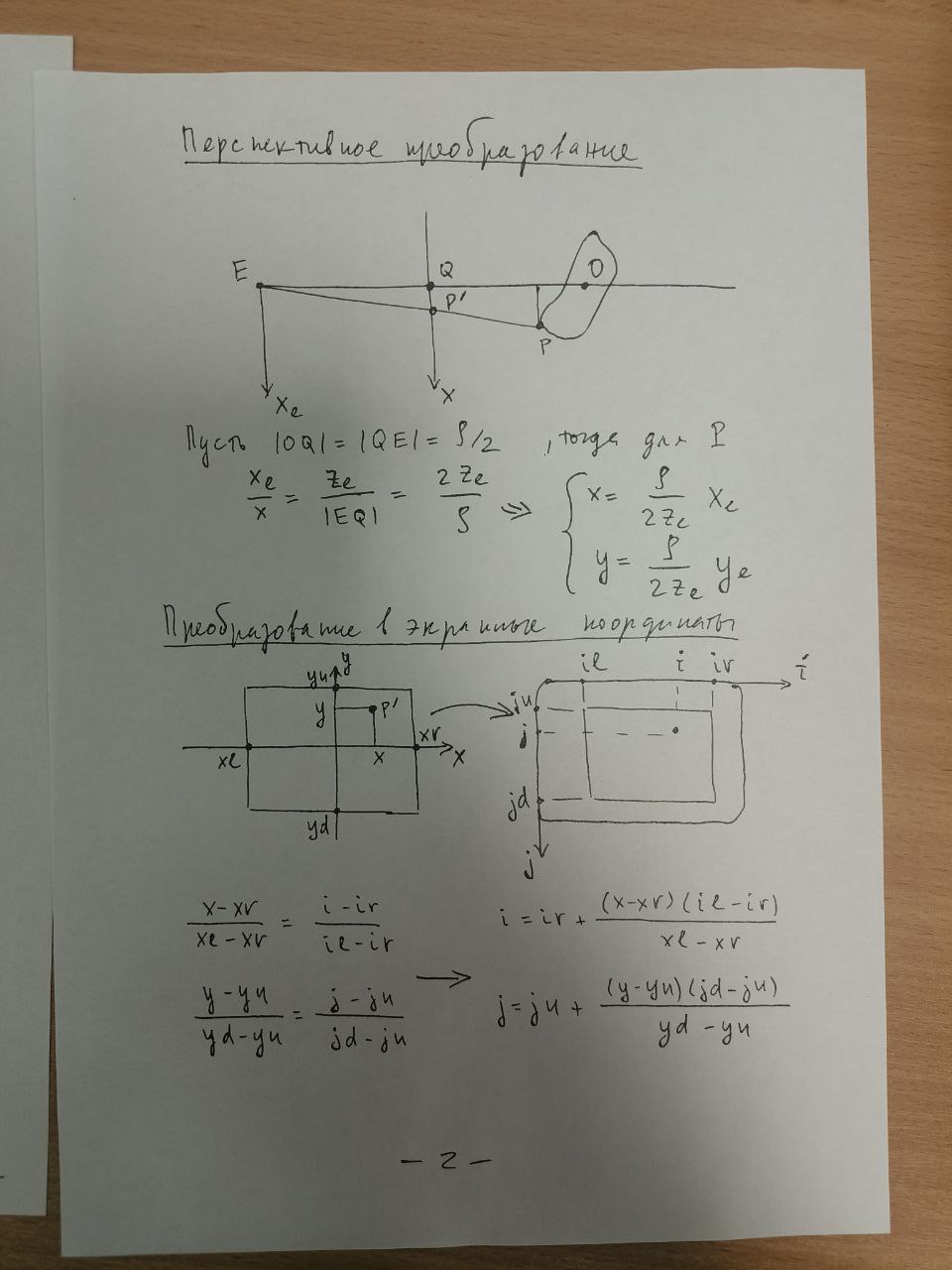
- Должна быть возможность изменять положение точки наблюдения;

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

**Ход работы**:

Для того, чтобы построить изображение трехмерного тела в виде каркасной модели, нужно провести перспективное преобразование получаемых на входе мировых координат. Необходимые расчеты приведены ниже:





Текст программы:

*#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** point**{**

public:

**double** x**,** y**,** z**;**

point**(){**

x **=** **0;** y **=** **0;** z **=** **0;**

**}**

point**(double** a**,** **double** b**,** **double** c**){**

x **=** a**;** y **=** b**;** z **=** c**;**

**}**

**};**

**class** vertex**{**

public:

point worldc**;**

point viewc**;**

vertex**(){**point p**(0,0,0);** worldc **=** p**;};**

**void** clvertex**(**point p**)**

**{**

worldc **=** p**;**

**}**

**~**vertex**(){};**

**void** view\_coord**(**point worldcoord**)**

**{**

**double** ro **=** worldcoord**.**x**,** teta**=**worldcoord**.**y**,** fi**=**worldcoord**.**z**;**

viewc**.**x **=** **-**worldc**.**x **\*** sin**(**teta**)** **+** worldc**.**y **\*** cos**(**teta**);**

viewc**.**y **=** **-**worldc**.**x **\*** cos**(**fi**)** **\*** cos**(**teta**)** **-** worldc**.**y **\*** cos**(**fi**)** **\*** sin**(**teta**)** **+** worldc**.**z **\*** sin**(**fi**);**

viewc**.**z **=** **-**worldc**.**x **\*** sin**(**fi**)** **\*** cos**(**teta**)** **-** worldc**.**y **\*** sin**(**fi**)** **\*** sin**(**teta**)** **-** worldc**.**z **\*** cos**(**fi**)** **+** ro**;**

**}**

**};**

**class** edge**{**

private:

vertex tmpvertex**,** fervertex**;**

public:

edge**(){};**

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

**{**

tmpvertex **=** sv**;**

fervertex **=** fv**;**

**}**

**~**edge**(){};**

**void** drawedge**(double** d**)** *//perspective transformation*

**{**

d **=** d**/2;**

**double** x1 **=** **(int)((**d**\***tmpvertex**.**viewc**.**x**)/**tmpvertex**.**viewc**.**z**);**

**double** y1 **=** **(int)((**d**\***tmpvertex**.**viewc**.**y**)/**tmpvertex**.**viewc**.**z**);**

**double** x2 **=** **(int)((**d**\***fervertex**.**viewc**.**x**)/**fervertex**.**viewc**.**z**);**

**double** y2 **=** **(int)((**d**\***fervertex**.**viewc**.**y**)/**fervertex**.**viewc**.**z**);**

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

**}**

**};**

**struct** strk**{**

public:

**int** a**,** b**;**

**};**

**class** drawingmtd**{**

private:

vertex **\***verts**;**

edge **\***edges**;**

strk **\***strk1**;**

point viewpoint**;**

**bool** proff**;**

public:

**int** n**,** m**;**

**friend** **class** edge**;**

**friend** **class** vertex**;**

drawingmtd**()** **:** proff**(**false**)** **{}**

**void** draw**()**

**{**

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

edges**[**i**].**drawedge**(**viewpoint**.**x**);**

**}**

**void** drawing**()**

**{**

**int** c**,** a**,** b**;**

**if** **(!**proff**)**

**{**

**FILE** **\***f**;**

f **=** fopen**(**"C:\\Users\\maslo\\Desktop\\lab3\\data.txt"**,** "r"**);**

**if** **(**f **!=** NULL**)**

**{**

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

verts **=** **new** vertex**[**n**];**

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

**{**

fscanf**(**f**,** "%d"**,** **&**a**);**

fscanf**(**f**,** "%d"**,** **&**b**);**

fscanf**(**f**,** "%d"**,** **&**c**);**

point p **=** point**(**a**\*100,** b**\*100,** c**\*100);**

verts**[**i**].**clvertex**(**p**);**

verts**[**i**].**view\_coord**(**viewpoint**);**

cout**<<**"a"**<<**a**<<**" b"**<<**b**<<**" c"**<<**c**;**

**}**

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

edges **=** **new** edge**[**m**];**

strk1 **=** **new** strk**[**m**];**

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

**{**

fscanf**(**f**,** "%d"**,** **&**a**);**

fscanf**(**f**,** "%d"**,** **&**b**);**

strk1**[**i**].**a **=** a**;**

strk1**[**i**].**b **=** b**;**

edges**[**i**].**edgech**(**verts**[**strk1**[**i**].**a**],** verts**[**strk1**[**i**].**b**]);**

cout**<<**"a"**<<**a**<<**" b"**<<**b**;**

**}**

proff **=** true**;**

**}**

**}**

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

verts**[**i**].**view\_coord**(**viewpoint**);**

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

edges**[**i**].**edgech**(**verts**[**strk1**[**i**].**a**],** verts**[**strk1**[**i**].**b**]);**

**}**

**void** setviewpoint**(double** ro**,** **double** teta**,** **double** fi**)**

**{**

viewpoint**.**x **=** ro**;**

viewpoint**.**y **=** teta**;**

viewpoint**.**z **=** fi**;**

**}**

**};**

**int** main**()**

**{**

**double** ro**=200,** teta**=0,** fi**=0;**

**int** d**;**

**int** gddriver **=** DETECT**,** gmode**,** errorcode**;**

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

drawingmtd cube**;**

cube**.**setviewpoint**(**ro**,** teta**,** fi**);**

cube**.**drawing**();**

cube**.**draw**();**

**while** **(!**kbhit**())**

**{**

cleardevice**();**

setbkcolor**(0);**

setcolor**(**WHITE**);**

teta **+=** **((**M\_PI**/27));**

fi **+=** **((**M\_PI**/27));**

cube**.**setviewpoint**(**ro**,** teta**,** fi**);**

cube**.**drawing**();**

cube**.**draw**();**

delay**(250);**

**}**

getch**();**

closegraph**();**

**return** **0;**

**}**