**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ТВ**

**отчЁт**

**по лабораторной работе № 2**

**по дисциплине «Компьютерный синтез трехмерных изображений»**

**Тема: ИССЛЕДОВАНИЕ ГЕОМЕТРИЧЕСКИХ ПРЕОБРАЗОВАНИЙ В ТРЕХМЕРНОМ ПРОСТРАНСТВЕ (3D ПРЕОБРАЗОВАНИЙ) И МЕХАНИЗМОВ ПРОЕЦИРОВАНИЯ**

**Вариант 6**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9105 |  | Шаривзянов Д. Р. |
| Преподаватель |  | Сирый Р. С. |

Санкт-Петербург

2024

ИССЛЕДОВАНИЕ ГЕОМЕТРИЧЕСКИХ ПРЕОБРАЗОВАНИЙ В ТРЕХМЕРНОМ ПРОСТРАНСТВЕ (3D ПРЕОБРАЗОВАНИЙ) И МЕХАНИЗМОВ ПРОЕЦИРОВАНИЯ

**Цель работы:**

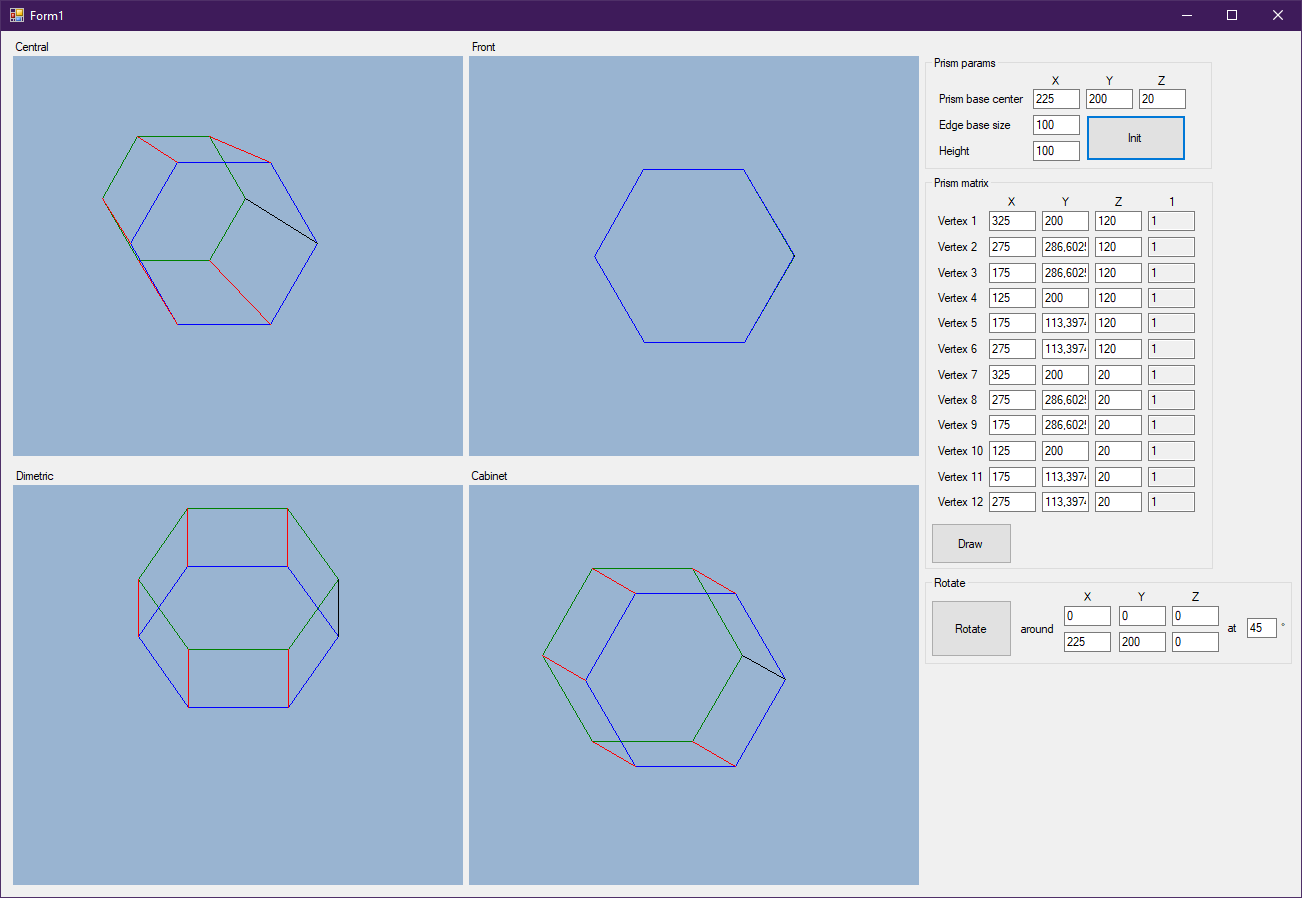
Целью лабораторной работы является знакомство с преобразованиями в 3D пространстве и механизмами проецирования.

**Исходные данные:**

|  |  |  |
| --- | --- | --- |
| Вариант | Вид многогранника | Типы проекции |
| 6 | Призма, в основании правильный шестиугольник | Центральная, спереди, диметрия, кабине |

|  |  |
| --- | --- |
| Вариант | Вид преобразования |
| 6 | Вращение многогранника вокруг произвольной прямой |

**Реализация формы программы:**



**Код программы:**

#include <math.h>

#define deg2rad (acos(-1))/180

// углы в радианах для проекций

const double psy = 45.0 \* deg2rad; // для димметрии

const double phi = atan(sin(psy)) \* deg2rad; // для димметрии

const double alpha = 30.0 \* deg2rad; // для Кавалье и Кабине

// координата для центральной проекции

const double d = 300;

const int vert\_num = 12; // количество вершин фигуры

const int dimension = 4; // число столбцов матрицы однородных координат //3-для двумерной графики, 4 для трехмерной

//матрицы общие

double hmg\_p[vert\_num][dimension] = { 0 }; // матрица однородных координат фигуры

int dek\_p[vert\_num][dimension - 1] = { 0 }; // матрица экранных координат фигуры

//матрицы проекций

double matrix\_Front[dimension][dimension] = // матрица вида спереди

{ 1, 0, 0, 0,

0, 1, 0, 0,

0, 0, 0, 0,

0, 0, 0, 1 };

double matrix\_Center[dimension][dimension] = //центральная

{ 1, 0, 0, 0,

0, 1, 0, 0,

0, 0, 0, 1 / d,

0, 0, 0, 1 };

double matrix\_Dimetry[dimension][dimension] =

{ cos(psy), sin(phi) \* sin(psy), 0, 0,

0, cos(phi), 0, 0,

sin(psy), -sin(phi) \* cos(psy), 0, 0,

0, 0, 0, 1 };

double matrix\_Cabinet[dimension][dimension] = {

1, 0, 0, 0,

0, 1, 0, 0,

-0.5\*cos(alpha), -0.5\*sin(alpha), 0, 0,

0, 0, 0, 1 };

private: System::Void update\_textBoxes(){

textBox\_x1->Text = Convert::ToString(hmg\_p[0][0]);

textBox\_y1->Text = Convert::ToString(hmg\_p[0][1]);

textBox\_z1->Text = Convert::ToString(hmg\_p[0][2]);

textBox\_w1->Text = Convert::ToString(hmg\_p[0][3]);

textBox\_x2->Text = Convert::ToString(hmg\_p[1][0]);

textBox\_y2->Text = Convert::ToString(hmg\_p[1][1]);

textBox\_z2->Text = Convert::ToString(hmg\_p[1][2]);

textBox\_w2->Text = Convert::ToString(hmg\_p[1][3]);

textBox\_x3->Text = Convert::ToString(hmg\_p[2][0]);

textBox\_y3->Text = Convert::ToString(hmg\_p[2][1]);

textBox\_z3->Text = Convert::ToString(hmg\_p[2][2]);

textBox\_w3->Text = Convert::ToString(hmg\_p[2][3]);

textBox\_x4->Text = Convert::ToString(hmg\_p[3][0]);

textBox\_y4->Text = Convert::ToString(hmg\_p[3][1]);

textBox\_z4->Text = Convert::ToString(hmg\_p[3][2]);

textBox\_w4->Text = Convert::ToString(hmg\_p[3][3]);

textBox\_x5->Text = Convert::ToString(hmg\_p[4][0]);

textBox\_y5->Text = Convert::ToString(hmg\_p[4][1]);

textBox\_z5->Text = Convert::ToString(hmg\_p[4][2]);

textBox\_w5->Text = Convert::ToString(hmg\_p[4][3]);

textBox\_x6->Text = Convert::ToString(hmg\_p[5][0]);

textBox\_y6->Text = Convert::ToString(hmg\_p[5][1]);

textBox\_z6->Text = Convert::ToString(hmg\_p[5][2]);

textBox\_w6->Text = Convert::ToString(hmg\_p[5][3]);

textBox\_x7->Text = Convert::ToString(hmg\_p[6][0]);

textBox\_y7->Text = Convert::ToString(hmg\_p[6][1]);

textBox\_z7->Text = Convert::ToString(hmg\_p[6][2]);

textBox\_w7->Text = Convert::ToString(hmg\_p[6][3]);

textBox\_x8->Text = Convert::ToString(hmg\_p[7][0]);

textBox\_y8->Text = Convert::ToString(hmg\_p[7][1]);

textBox\_z8->Text = Convert::ToString(hmg\_p[7][2]);

textBox\_w8->Text = Convert::ToString(hmg\_p[7][3]);

textBox\_x9->Text = Convert::ToString(hmg\_p[8][0]);

textBox\_y9->Text = Convert::ToString(hmg\_p[8][1]);

textBox\_z9->Text = Convert::ToString(hmg\_p[8][2]);

textBox\_w9->Text = Convert::ToString(hmg\_p[8][3]);

textBox\_x10->Text = Convert::ToString(hmg\_p[9][0]);

textBox\_y10->Text = Convert::ToString(hmg\_p[9][1]);

textBox\_z10->Text = Convert::ToString(hmg\_p[9][2]);

textBox\_w10->Text = Convert::ToString(hmg\_p[9][3]);

textBox\_x11->Text = Convert::ToString(hmg\_p[10][0]);

textBox\_y11->Text = Convert::ToString(hmg\_p[10][1]);

textBox\_z11->Text = Convert::ToString(hmg\_p[10][2]);

textBox\_w11->Text = Convert::ToString(hmg\_p[10][3]);

textBox\_x12->Text = Convert::ToString(hmg\_p[11][0]);

textBox\_y12->Text = Convert::ToString(hmg\_p[11][1]);

textBox\_z12->Text = Convert::ToString(hmg\_p[11][2]);

textBox\_w12->Text = Convert::ToString(hmg\_p[11][3]);

}

private: System::Void matrix\_mult(int number\_of\_vertex\_A, double A[][dimension], double B[][dimension], double C[][dimension]) {

int i, j, k;

for (i = 0; i < number\_of\_vertex\_A; i++)

for (j = 0; j < dimension; j++) {

C[i][j] = 0;

for (k = 0; k < dimension; k++)

C[i][j] += (A[i][k] \* B[k][j]);

}

}

private: System::Void hmg2dek(int number\_of\_vertex\_A, double HMG[][dimension], int DEK[][dimension - 1]) {

for (int i = 0; i < number\_of\_vertex\_A; i++) {

double a = HMG[i][dimension - 1];

if (a != 0) {

DEK[i][0] = HMG[i][0] / a;

DEK[i][1] = HMG[i][1] / a;

}

else {

System::Windows::Forms::MessageBox::Show("Division by zero is impossible! Check the coordinates of the shape and the transformation matrix");

break;

}

}

}

private: System::Void pictureBox\_front\_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e) {

int x1 = dek\_p[0][0];

int y1 = dek\_p[0][1];

int z1 = dek\_p[0][2];

int x2 = dek\_p[1][0];

int y2 = dek\_p[1][1];

int z2 = dek\_p[1][2];

int x3 = dek\_p[2][0];

int y3 = dek\_p[2][1];

int z3 = dek\_p[2][2];

int x4 = dek\_p[3][0];

int y4 = dek\_p[3][1];

int z4 = dek\_p[3][2];

int x5 = dek\_p[4][0];

int y5 = dek\_p[4][1];

int z5 = dek\_p[4][2];

int x6 = dek\_p[5][0];

int y6 = dek\_p[5][1];

int z6 = dek\_p[5][2];

int x7 = dek\_p[6][0];

int y7 = dek\_p[6][1];

int z7 = dek\_p[6][2];

int x8 = dek\_p[7][0];

int y8 = dek\_p[7][1];

int z8 = dek\_p[7][2];

int x9 = dek\_p[8][0];

int y9 = dek\_p[8][1];

int z9 = dek\_p[8][2];

int x10 = dek\_p[9][0];

int y10 = dek\_p[9][1];

int z10 = dek\_p[9][2];

int x11 = dek\_p[10][0];

int y11 = dek\_p[10][1];

int z11 = dek\_p[10][2];

int x12 = dek\_p[11][0];

int y12 = dek\_p[11][1];

int z12 = dek\_p[11][2];

//верхнее основание

e->Graphics->DrawLine(System::Drawing::Pens::Green, x1, y1, x2, y2);

e->Graphics->DrawLine(System::Drawing::Pens::Green, x2, y2, x3, y3);

e->Graphics->DrawLine(System::Drawing::Pens::Green, x3, y3, x4, y4);

e->Graphics->DrawLine(System::Drawing::Pens::Green, x4, y4, x5, y5);

e->Graphics->DrawLine(System::Drawing::Pens::Green, x5, y5, x6, y6);

e->Graphics->DrawLine(System::Drawing::Pens::Green, x6, y6, x1, y1);

//нижнее основание

e->Graphics->DrawLine(System::Drawing::Pens::Blue, x7, y7, x8, y8);

e->Graphics->DrawLine(System::Drawing::Pens::Blue, x8, y8, x9, y9);

e->Graphics->DrawLine(System::Drawing::Pens::Blue, x9, y9, x10, y10);

e->Graphics->DrawLine(System::Drawing::Pens::Blue, x10, y10, x11, y11);

e->Graphics->DrawLine(System::Drawing::Pens::Blue, x11, y11, x12, y12);

e->Graphics->DrawLine(System::Drawing::Pens::Blue, x12, y12, x7, y7);

//рёбры между основаниями (высоты)

e->Graphics->DrawLine(System::Drawing::Pens::Black, x1, y1, x7, y7);

e->Graphics->DrawLine(System::Drawing::Pens::Red, x2, y2, x8, y8);

e->Graphics->DrawLine(System::Drawing::Pens::Red, x3, y3, x9, y9);

e->Graphics->DrawLine(System::Drawing::Pens::Red, x4, y4, x10, y10);

e->Graphics->DrawLine(System::Drawing::Pens::Red, x5, y5, x11, y11);

e->Graphics->DrawLine(System::Drawing::Pens::Red, x6, y6, x12, y12);

}

Остальные 3 обработчика pictureBox\_x\_Paint идентичны этому

private: System::Void btn\_draw\_Click(System::Object^ sender, System::EventArgs^ e) {

hmg\_p[0][0] = Convert::ToDouble(textBox\_x1->Text);

hmg\_p[0][1] = Convert::ToDouble(textBox\_y1->Text);

hmg\_p[0][2] = Convert::ToDouble(textBox\_z1->Text);

hmg\_p[0][3] = Convert::ToDouble(textBox\_w1->Text);

hmg\_p[1][0] = Convert::ToDouble(textBox\_x2->Text);

hmg\_p[1][1] = Convert::ToDouble(textBox\_y2->Text);

hmg\_p[1][2] = Convert::ToDouble(textBox\_z2->Text);

hmg\_p[1][3] = Convert::ToDouble(textBox\_w2->Text);

hmg\_p[2][0] = Convert::ToDouble(textBox\_x3->Text);

hmg\_p[2][1] = Convert::ToDouble(textBox\_y3->Text);

hmg\_p[2][2] = Convert::ToDouble(textBox\_z3->Text);

hmg\_p[2][3] = Convert::ToDouble(textBox\_w3->Text);

hmg\_p[3][0] = Convert::ToDouble(textBox\_x4->Text);

hmg\_p[3][1] = Convert::ToDouble(textBox\_y4->Text);

hmg\_p[3][2] = Convert::ToDouble(textBox\_z4->Text);

hmg\_p[3][3] = Convert::ToDouble(textBox\_w4->Text);

hmg\_p[4][0] = Convert::ToDouble(textBox\_x5->Text);

hmg\_p[4][1] = Convert::ToDouble(textBox\_y5->Text);

hmg\_p[4][2] = Convert::ToDouble(textBox\_z5->Text);

hmg\_p[4][3] = Convert::ToDouble(textBox\_w5->Text);

hmg\_p[5][0] = Convert::ToDouble(textBox\_x6->Text);

hmg\_p[5][1] = Convert::ToDouble(textBox\_y6->Text);

hmg\_p[5][2] = Convert::ToDouble(textBox\_z6->Text);

hmg\_p[5][3] = Convert::ToDouble(textBox\_w6->Text);

hmg\_p[6][0] = Convert::ToDouble(textBox\_x7->Text);

hmg\_p[6][1] = Convert::ToDouble(textBox\_y7->Text);

hmg\_p[6][2] = Convert::ToDouble(textBox\_z7->Text);

hmg\_p[6][3] = Convert::ToDouble(textBox\_w7->Text);

hmg\_p[7][0] = Convert::ToDouble(textBox\_x8->Text);

hmg\_p[7][1] = Convert::ToDouble(textBox\_y8->Text);

hmg\_p[7][2] = Convert::ToDouble(textBox\_z8->Text);

hmg\_p[7][3] = Convert::ToDouble(textBox\_w8->Text);

hmg\_p[8][0] = Convert::ToDouble(textBox\_x9->Text);

hmg\_p[8][1] = Convert::ToDouble(textBox\_y9->Text);

hmg\_p[8][2] = Convert::ToDouble(textBox\_z9->Text);

hmg\_p[8][3] = Convert::ToDouble(textBox\_w9->Text);

hmg\_p[9][0] = Convert::ToDouble(textBox\_x10->Text);

hmg\_p[9][1] = Convert::ToDouble(textBox\_y10->Text);

hmg\_p[9][2] = Convert::ToDouble(textBox\_z10->Text);

hmg\_p[9][3] = Convert::ToDouble(textBox\_w10->Text);

hmg\_p[10][0] = Convert::ToDouble(textBox\_x11->Text);

hmg\_p[10][1] = Convert::ToDouble(textBox\_y11->Text);

hmg\_p[10][2] = Convert::ToDouble(textBox\_z11->Text);

hmg\_p[10][3] = Convert::ToDouble(textBox\_w11->Text);

hmg\_p[11][0] = Convert::ToDouble(textBox\_x12->Text);

hmg\_p[11][1] = Convert::ToDouble(textBox\_y12->Text);

hmg\_p[11][2] = Convert::ToDouble(textBox\_z12->Text);

hmg\_p[11][3] = Convert::ToDouble(textBox\_w12->Text);

double Result[vert\_num][dimension] = { 0 };

matrix\_mult(vert\_num, hmg\_p, matrix\_Center, Result);

hmg2dek(vert\_num, Result, dek\_p);

pictureBox\_central->Refresh();

matrix\_mult(vert\_num, hmg\_p, matrix\_Front, Result);

hmg2dek(vert\_num, Result, dek\_p);

pictureBox\_front->Refresh();

matrix\_mult(vert\_num, hmg\_p, matrix\_Dimetry, Result);

hmg2dek(vert\_num, Result, dek\_p);

pictureBox\_dimetric->Refresh();

matrix\_mult(vert\_num, hmg\_p, matrix\_Cabinet, Result);

hmg2dek(vert\_num, Result, dek\_p);

update\_textBoxes();

pictureBox\_cabinet->Refresh();

}

private: System::Void btn\_init\_Click(System::Object^ sender, System::EventArgs^ e) {

double centerX = Convert::ToDouble(textBox\_centroidx->Text);

double centerY = Convert::ToDouble(textBox\_centroidy->Text);

double centerZ = Convert::ToDouble(textBox\_centroidz->Text);

double edge = Convert::ToDouble(textBox\_edge->Text);

double height = Convert::ToDouble(textBox\_height->Text);

double angle\_rad = 60 \* deg2rad;

hmg\_p[0][0] = centerX + edge;

hmg\_p[0][1] = centerY;

hmg\_p[0][2] = centerZ + height;

hmg\_p[0][3] = 1;

for (int i = 1; i < vert\_num / 2; i++) {

hmg\_p[i][0] = centerX + edge \* cos(angle\_rad \* i);

hmg\_p[i][1] = centerY + edge \* sin(angle\_rad \* i);

hmg\_p[i][2] = centerZ + height;

hmg\_p[i][3] = 1;

}

hmg\_p[6][0] = centerX + edge;

hmg\_p[6][1] = centerY;

hmg\_p[6][2] = centerZ;

hmg\_p[6][3] = 1;

for (int i = vert\_num / 2; i < vert\_num; i++) {

hmg\_p[i][0] = centerX + edge \* cos(angle\_rad \* i);

hmg\_p[i][1] = centerY + edge \* sin(angle\_rad \* i);

hmg\_p[i][2] = centerZ;

hmg\_p[i][3] = 1;

}

double Result[vert\_num][dimension] = { 0 };

matrix\_mult(vert\_num, hmg\_p, matrix\_Center, Result);

hmg2dek(vert\_num, Result, dek\_p);

pictureBox\_central->Refresh();

matrix\_mult(vert\_num, hmg\_p, matrix\_Front, Result);

hmg2dek(vert\_num, Result, dek\_p);

pictureBox\_front->Refresh();

matrix\_mult(vert\_num, hmg\_p, matrix\_Dimetry, Result);

hmg2dek(vert\_num, Result, dek\_p);

pictureBox\_dimetric->Refresh();

matrix\_mult(vert\_num, hmg\_p, matrix\_Cabinet, Result);

hmg2dek(vert\_num, Result, dek\_p);

update\_textBoxes();

pictureBox\_cabinet->Refresh();

}

private: System::Void btn\_rotate\_Click(System::Object^ sender, System::EventArgs^ e) {

double matrix\_buf\_1[vert\_num][dimension] = { 0 };// буфферная матрица преобразования

double matrix\_buf\_2[vert\_num][dimension] = { 0 };// буфферная матрица преобразования

double px = Convert::ToDouble(textBox\_rotx1->Text);

double py = Convert::ToDouble(textBox\_roty1->Text);

double pz = Convert::ToDouble(textBox\_rotz1->Text);

double qx = Convert::ToDouble(textBox\_rotx2->Text);

double qy = Convert::ToDouble(textBox\_roty2->Text);

double qz = Convert::ToDouble(textBox\_rotz2->Text);

double cx = hmg\_p[0][0];

double cy = hmg\_p[0][1];

double cz = hmg\_p[0][2];

double alpha = Convert::ToDouble(textBox\_degree->Text) \* deg2rad;

// Шаг 1: Перенос P в начало координат-------------------------------------------------------------

double T[dimension][dimension] = {

{1, 0, 0, -px},

{0, 1, 0, -py},

{0, 0, 1, -pz},

{0, 0, 0, 1}

};

// Шаг 2: Выравнивание вектора PQ с плоскостью XZ-------------------------------------------------------------

// Рассчитываем угол φ

double dx = qx - px;

double dy = qy - py;

double dz = qz - pz;

double phi = atan2(dy, dx);

// Матрица поворота вокруг Z для выравнивания с XZ

double Rz[dimension][dimension] = {

{cos(-phi), -sin(-phi), 0, 0},

{sin(-phi), cos(-phi), 0, 0},

{0, 0, 1, 0},

{0, 0, 0, 1}

};

// Шаг 3: Выравнивание вектора PQ с осью Z-------------------------------------------------------------

// Рассчитываем угол θ

double theta = atan2(sqrt(dx \* dx + dy \* dy), dz);

// Матрица поворота вокруг Y для выравнивания с Z

double Ry[dimension][dimension] = {

{cos(theta), 0, sin(theta), 0},

{0, 1, 0, 0},

{-sin(theta), 0, cos(theta), 0},

{0, 0, 0, 1}

};

// Шаг 4: Поворот вокруг оси Z-------------------------------------------------------------

// Матрица поворота вокруг Z на угол alpha

double Rz\_alpha[dimension][dimension] = {

{cos(alpha), -sin(alpha), 0, 0},

{sin(alpha), cos(alpha), 0, 0},

{0, 0, 1, 0},

{0, 0, 0, 1}

};

// Шаг 5: Обратные преобразования-------------------------------------------------------------

// Обратный поворот вокруг Y

double Ry\_inv[dimension][dimension] = {

{cos(-theta), 0, -sin(-theta), 0},

{0, 1, 0, 0},

{sin(-theta), 0, cos(-theta), 0},

{0, 0, 0, 1}

};

// Обратный поворот вокруг Z

double Rz\_inv[dimension][dimension] = {

{cos(-phi), sin(-phi), 0, 0},

{-sin(-phi), cos(-phi), 0, 0},

{0, 0, 1, 0},

{0, 0, 0, 1}

};

// Обратный перенос

double T\_inv[dimension][dimension] = {

{1, 0, 0, px},

{0, 1, 0, py},

{0, 0, 1, pz},

{0, 0, 0, 1}

};

matrix\_mult(vert\_num, hmg\_p, T, matrix\_buf\_1);

matrix\_mult(vert\_num, matrix\_buf\_1, Rz, matrix\_buf\_2);

matrix\_mult(vert\_num, matrix\_buf\_2, Ry, matrix\_buf\_1);

matrix\_mult(vert\_num, matrix\_buf\_1, Rz\_alpha, matrix\_buf\_2);

matrix\_mult(vert\_num, matrix\_buf\_2, Ry\_inv, matrix\_buf\_1);

matrix\_mult(vert\_num, matrix\_buf\_1, Rz\_inv, matrix\_buf\_2);

matrix\_mult(vert\_num, matrix\_buf\_2, T\_inv, hmg\_p);

double Result[vert\_num][dimension] = { 0 };

matrix\_mult(vert\_num, hmg\_p, matrix\_Center, Result);

hmg2dek(vert\_num, Result, dek\_p);

pictureBox\_central->Refresh();

matrix\_mult(vert\_num, hmg\_p, matrix\_Front, Result);

hmg2dek(vert\_num, Result, dek\_p);

pictureBox\_front->Refresh();

matrix\_mult(vert\_num, hmg\_p, matrix\_Dimetry, Result);

hmg2dek(vert\_num, Result, dek\_p);

pictureBox\_dimetric->Refresh();

matrix\_mult(vert\_num, hmg\_p, matrix\_Cabinet, Result);

hmg2dek(vert\_num, Result, dek\_p);

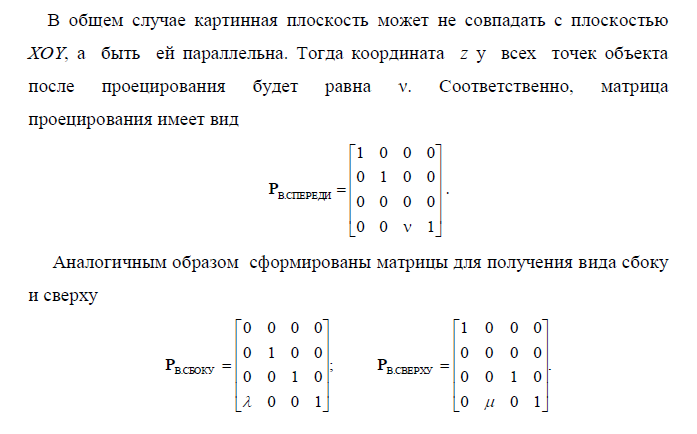
pictureBox\_cabinet->Refresh();

update\_textBoxes();

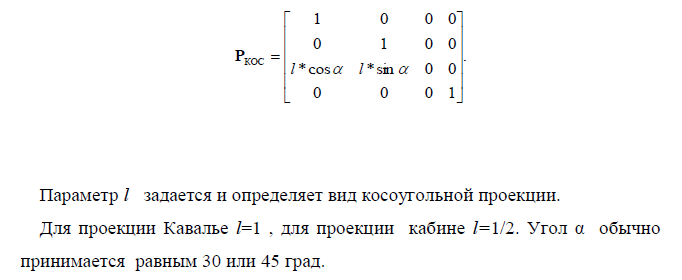
}

1. Преобразования проецирований

**Топологическое - спереди:**

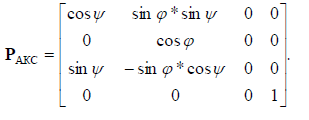


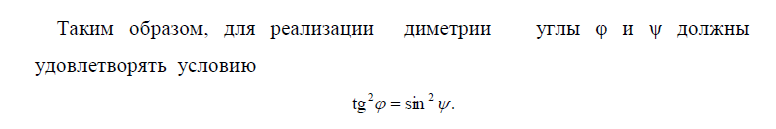
**Косоугольное - кабине:**



**Аксонометрическое - диметрия:**

В изометрии углы между осями XYZ одинаковые и равны 120°, в диметрии равны два из трех углов, в триметрии величина всех углов между осями разная.

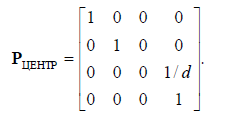




Возьмём угол , тогда

**Центральное - одноточечное:**

Возьмём центр проекции d = 300



1. Специфичное преобразование

Вращение многогранника вокруг произвольной прямой состоит из следующих шагов:

Обозначим произвольную прямую вектором

1. Перенос P в начало координат

{1, 0, 0, -px},

{0, 1, 0, -py},

{0, 0, 1, -pz},

{0, 0, 0, 1}

1. Выравнивание вектора PQ с плоскостью XZ

Этот шаг требует вычисления угла φ между проекцией вектора на плоскость YZ и осью Y. Этот угол используется для поворота вокруг оси Z.

// Рассчитываем угол φ

double dx = qx - px;

double dy = qy - py;

double dz = qz - pz;

double phi = atan2(dy, dx);

// Матрица поворота вокруг Z для выравнивания с XZ

{cos(-phi), -sin(-phi), 0, 0},

{sin(-phi), cos(-phi), 0, 0},

{0, 0, 1, 0},

{0, 0, 0, 1}

1. Выравнивание вектора PQ с осью Z

После того как вектор будет выровнен с плоскостью XZ, следующий шаг — выравнивание с осью Z, что требует поворота вокруг оси Y.

// Рассчитываем угол θ

double theta = atan2(sqrt(dx \* dx + dy \* dy), dz);

// Матрица поворота вокруг Y для выравнивания с Z

{cos(theta), 0, sin(theta), 0},

{0, 1, 0, 0},

{-sin(theta), 0, cos(theta), 0},

{0, 0, 0, 1}

1. Поворот вокруг оси Z

Теперь, когда ось вращения выровнена с осью Z, можно выполнить поворот на заданный угол 𝛼.

// Матрица поворота вокруг Z на угол alpha

{cos(alpha), -sin(alpha), 0, 0},

{sin(alpha), cos(alpha), 0, 0},

{0, 0, 1, 0},

{0, 0, 0, 1}

1. Обратные преобразования

После вращения нужно выполнить обратные преобразования для возврата фигуры в исходное положение.

// Обратный поворот вокруг Y

{cos(-theta), 0, -sin(-theta), 0},

{0, 1, 0, 0},

{sin(-theta), 0, cos(-theta), 0},

{0, 0, 0, 1}

// Обратный поворот вокруг Z

{cos(-phi), sin(-phi), 0, 0},

{-sin(-phi), cos(-phi), 0, 0},

{0, 0, 1, 0},

{0, 0, 0, 1}

// Обратный перенос

{1, 0, 0, px},

{0, 1, 0, py},

{0, 0, 1, pz},

{0, 0, 0, 1}

**Выводы:**

Были изучены механизмы проецирований, а также разработано преобразование вращения многогранника вокруг произвольной прямой в 3D пространстве.