МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра ТВ

ОТЧЁТ

по лабораторной работе № 1

по дисциплине «Компьютерный синтез трехмерных изображений»

Тема: ИССЛЕДОВАНИЕ ГЕОМЕТРИЧЕСКИХ ПРЕОБРАЗОВАНИЙ НА ПЛОСКОСТИ (2D ПРЕОБРАЗОВАНИЙ)

Вариант 4

Студент гр. 9105 _____ Шаривзянов Д. Р.

Преподаватель _____ Сирый Р. С.

Санкт-Петербург

2024

ИССЛЕДОВАНИЕ ГЕОМЕТРИЧЕСКИХ ПРЕОБРАЗОВАНИЙ НА ПЛОСКОСТИ (2D ПРЕОБРАЗОВАНИЙ)

Цель работы:

Целью лабораторной работы является знакомство с аффинными преобразованиями в 2D пространстве и средствами отображения фигур с помощью C++ Window Forms.

Код программы:

```
#include<math.h>

const int vert_num = 4; // количество вершин фигуры
	const int dimension = 3;// число столбцов матрицы однородных координат
	//3-для двумерной графики, 4 для трехмерной
	double  hmg_p[vert_num][dimension] = { 0 }; // матрица однородных координат // фигуры
	int dek_p[vert_num][dimension - 1] = { 0 }; // матрица экранных координат //фигуры
	double matrix_T[dimension][dimension] = { 0 };// матрица преобразования

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 make_trans_mat(double x, double y, double result_mat[dimension][dimension]) {
		result_mat[0][0] = 1;
		result_mat[0][1] = 0;
		result_mat[0][2] = 0;

		result_mat[1][0] = 0;
		result_mat[1][1] = 1;
		result_mat[1][2] = 0;

		result_mat[2][0] = x;
```

```
                result_mat[2][1] = y;
                result_mat[2][2] = 1;
        }

private: System::Void btn_draw_Click(System::Object^ sender, System::EventArgs^ e) {
                // считывание введенных пользователем координат в матрицу. Использует //-ся функция Convert::ToInt32 для
преобразования типов данных

                hmg_p[0][0] = Convert::ToInt32(textBox_x1->Text);
                textBox_y1->Text = textBox_x1->Text;
                hmg_p[0][1] = Convert::ToInt32(textBox_y1->Text);
                hmg_p[0][2] = Convert::ToInt32(textBox_z1->Text);

                textBox_x2->Text = textBox_x4->Text;
                hmg_p[1][0] = Convert::ToInt32(textBox_x2->Text);
                textBox_y2->Text = textBox_y1->Text;
                hmg_p[1][1] = Convert::ToInt32(textBox_y2->Text);
                hmg_p[1][2] = Convert::ToInt32(textBox_z2->Text);

                textBox_y4->Text = textBox_x4->Text;
                hmg_p[3][0] = Convert::ToInt32(textBox_x4->Text);
                hmg_p[3][1] = Convert::ToInt32(textBox_y4->Text);
                hmg_p[3][2] = Convert::ToInt32(textBox_z4->Text);

                textBox_x3->Text = textBox_x1->Text;
                hmg_p[2][0] = Convert::ToInt32(textBox_x3->Text);
                textBox_y3->Text = textBox_y4->Text;
                hmg_p[2][1] = Convert::ToInt32(textBox_y3->Text);
                hmg_p[2][2] = Convert::ToInt32(textBox_z3->Text);

                hmg2dek(vert_num, hmg_p, dek_p); // вызов функции перевода координат из однородных в экранные
                pictureBox1->Refresh(); // вызов перерисовки элемента pictureBox.
        }

private: System::Void pictureBox1_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e) {
        //int side = x2 - x1;
        //
        //e->Graphics->DrawRectangle(System::Drawing::Pens::Red, x1, y1, side+10, side);

        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];

        e->Graphics->DrawLine(System::Drawing::Pens::Red, x1, y1, x2, y2);
        e->Graphics->DrawLine(System::Drawing::Pens::Red, x1, y1, x3, y3);
        e->Graphics->DrawLine(System::Drawing::Pens::Red, x2, y2, x4, y4);
        e->Graphics->DrawLine(System::Drawing::Pens::Red, x3, y3, x4, y4);
}

private: System::Void button_transfer_Click(System::Object^ sender, System::EventArgs^ e) {
        double matrix_T_loc1[dimension][dimension] = { 0 };// матрица преобразования

        textBox_transx3->Text = textBox_transx->Text;
```

```
		textBox_transy3->Text = textBox_transy->Text;

		int cx = Convert::ToDouble(textBox_transx3->Text);
		int cy = Convert::ToDouble(textBox_transy3->Text);
		make_trans_mat(cx, cy, matrix_T_loc1);

		double result_mat1[vert_num][dimension] = { 0 };
		matrix_mult(vert_num, hmg_p, matrix_T_loc1, result_mat1);

		hmg2dek(vert_num, result_mat1, dek_p); // вызов функции перевода координат из однородных в экранные

		for (int i = 0; i < vert_num; i++)
			for (int j = 0; j < dimension; j++)
				hmg_p[i][j] = result_mat1[i][j];

		pictureBox1->Refresh(); // вызов перерисовки элемента pictureBox.
	}
	private: System::Void button_scale_Click(System::Object^ sender, System::EventArgs^ e) {
		double matrix_T_loc1[dimension][dimension] = { 0 };// матрица преобразования
		double matrix_T_loc2[dimension][dimension] = { 0 };// матрица преобразования
		double matrix_T_loc3[dimension][dimension] = { 0 };// матрица преобразования

		int cx = (dek_p[0][0] + dek_p[1][0] + dek_p[2][0] + dek_p[3][0]) / 4;
		int cy = (dek_p[0][1] + dek_p[1][1] + dek_p[2][1] + dek_p[3][1]) / 4;

		make_trans_mat(-cx, -cy, matrix_T_loc1);
		make_trans_mat(cx, cy, matrix_T_loc3);

		textBox_scalex1->Text = textBox_scalex->Text;
		matrix_T_loc2[0][0] = Convert::ToDouble(textBox_scalex1->Text);
		matrix_T_loc2[0][1] = Convert::ToDouble(textBox_scaley1->Text);
		matrix_T_loc2[0][2] = Convert::ToDouble(textBox_scalez1->Text);

		matrix_T_loc2[1][0] = Convert::ToDouble(textBox_scalex2->Text);
		textBox_scaley2->Text = textBox_scaley->Text;
		matrix_T_loc2[1][1] = Convert::ToDouble(textBox_scaley2->Text);
		matrix_T_loc2[1][2] = Convert::ToDouble(textBox_scalez2->Text);

		matrix_T_loc2[2][0] = Convert::ToDouble(textBox_scalex3->Text);
		matrix_T_loc2[2][1] = Convert::ToDouble(textBox_scaley3->Text);
		matrix_T_loc2[2][2] = Convert::ToDouble(textBox_scalez3->Text);

		double result_mat1[vert_num][dimension] = { 0 };
		double result_mat2[vert_num][dimension] = { 0 };

		matrix_mult(vert_num, hmg_p, matrix_T_loc1, result_mat1);
		matrix_mult(vert_num, result_mat1, matrix_T_loc2, result_mat2);
		matrix_mult(vert_num, result_mat2, matrix_T_loc3, result_mat1);

		hmg2dek(vert_num, result_mat1, dek_p); // вызов функции перевода координат из однородных в экранные

		for (int i = 0; i < vert_num; i++)
			for (int j = 0; j < dimension; j++)
				hmg_p[i][j] = result_mat1[i][j];

		pictureBox1->Refresh(); // вызов перерисовки элемента pictureBox.
	}
	private: System::Void button_rotate_Click(System::Object^ sender, System::EventArgs^ e) {
		double matrix_T_loc1[dimension][dimension] = { 0 };// матрица преобразования
		double matrix_T_loc2[dimension][dimension] = { 0 };// матрица преобразования
		double matrix_T_loc3[dimension][dimension] = { 0 };// матрица преобразования
```

```
		int cx = (dek_p[0][0] + dek_p[1][0] + dek_p[2][0] + dek_p[3][0]) / 4;
		int cy = (dek_p[0][1] + dek_p[1][1] + dek_p[2][1] + dek_p[3][1]) / 4;

		make_trans_mat(-cx, -cy, matrix_T_loc1);
		make_trans_mat(cx, cy, matrix_T_loc3);

		double alpha = (Convert::ToDouble(textBox_rotatealpha->Text) * (acos(-1))) / 180;

		textBox_rotatex1->Text = (cos(alpha)).ToString();
		matrix_T_loc2[0][0] = Convert::ToDouble(textBox_rotatex1->Text);
		textBox_rotatey1->Text = (-sin(alpha)).ToString();
		matrix_T_loc2[0][1] = Convert::ToDouble(textBox_rotatey1->Text);
		matrix_T_loc2[0][2] = Convert::ToDouble(textBox_rotatez1->Text);

		textBox_rotatex2->Text = (sin(alpha)).ToString();
		matrix_T_loc2[1][0] = Convert::ToDouble(textBox_rotatex2->Text);
		textBox_rotatey2->Text = (cos(alpha)).ToString();
		matrix_T_loc2[1][1] = Convert::ToDouble(textBox_rotatey2->Text);
		matrix_T_loc2[1][2] = Convert::ToDouble(textBox_rotatez2->Text);

		matrix_T_loc2[2][0] = Convert::ToDouble(textBox_rotatex3->Text);
		matrix_T_loc2[2][1] = Convert::ToDouble(textBox_rotatey3->Text);
		matrix_T_loc2[2][2] = Convert::ToDouble(textBox_rotatez3->Text);

		double result_mat1[vert_num][dimension] = { 0 };
		double result_mat2[vert_num][dimension] = { 0 };

		matrix_mult(vert_num, hmg_p, matrix_T_loc1, result_mat1);
		matrix_mult(vert_num, result_mat1, matrix_T_loc2, result_mat2);
		matrix_mult(vert_num, result_mat2, matrix_T_loc3, result_mat1);

		hmg2dek(vert_num, result_mat1, dek_p); // вызов функции перевода координат из однородных в экранные

		for (int i = 0; i < vert_num; i++)
			for (int j = 0; j < dimension; j++)
				hmg_p[i][j] = result_mat1[i][j];

		pictureBox1->Refresh(); // вызов перерисовки элемента pictureBox.
}
private: System::Void button_mirror_Click(System::Object^ sender, System::EventArgs^ e) {
		double matrix_T_loc1[dimension][dimension] = { 0 };// матрица преобразования
		double matrix_T_loc2[dimension][dimension] = { 0 };// матрица преобразования
		double matrix_T_loc3[dimension][dimension] = { 0 };// матрица преобразования

		int vertex = Convert::ToInt32(numericUpDown1->Value);

		int cx = dek_p[vertex - 1][0];
		int cy = dek_p[vertex - 1][1];

		make_trans_mat(-cx, -cy, matrix_T_loc1);
		make_trans_mat(cx, cy, matrix_T_loc3);

		double alpha = acos(-1);

		// Set up the rotation matrix
		matrix_T_loc2[0][0] = cos(alpha);
		matrix_T_loc2[0][1] = -sin(alpha);
		matrix_T_loc2[0][2] = 0;

		matrix_T_loc2[1][0] = sin(alpha);
		matrix_T_loc2[1][1] = cos(alpha);
		matrix_T_loc2[1][2] = 0;
```

```
        matrix_T_loc2[2][0] = 0;
        matrix_T_loc2[2][1] = 0;
        matrix_T_loc2[2][2] = 1;

        double result_mat1[vert_num][dimension] = { 0 };
        double result_mat2[vert_num][dimension] = { 0 };

        matrix_mult(vert_num, hmg_p, matrix_T_loc1, result_mat1);
        matrix_mult(vert_num, result_mat1, matrix_T_loc2, result_mat2);
        matrix_mult(vert_num, result_mat2, matrix_T_loc3, result_mat1);

        hmg2dek(vert_num, result_mat1, dek_p); // вызов функции перевода координат из однородных в экранные

        for (int i = 0; i < vert_num; i++)
                for (int j = 0; j < dimension; j++)
                        hmg_p[i][j] = result_mat1[i][j];

        pictureBox1->Refresh(); // вызов перерисовки элемента pictureBox.
}
```

1. Основные преобразования

Вариант	Вид многоугольника	Вид преобразования
4	Квадрат	Зеркальное отображение многоугольника относительно одной из его вершин

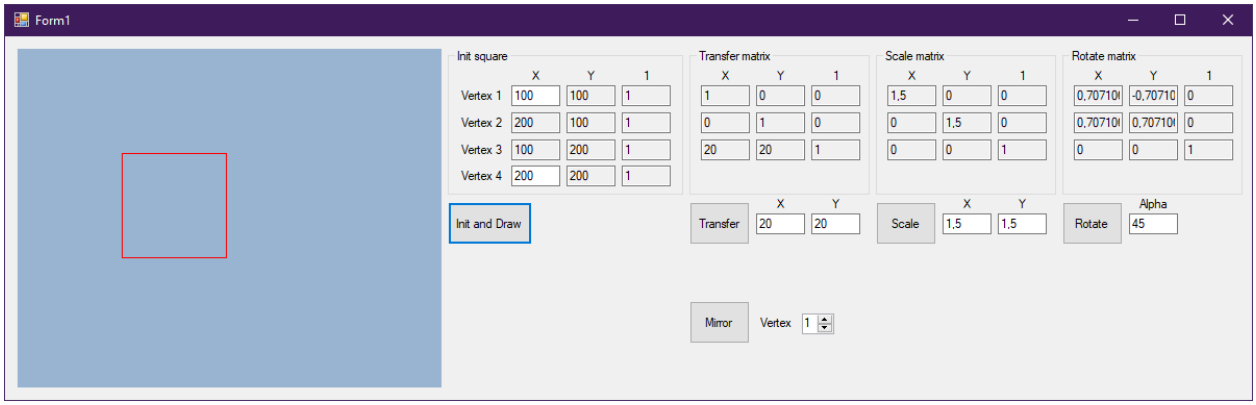

Рис 1. Основная форма программы.

Соответственно, матрицы основных $2D$ преобразований в однородных координатах имеют вид

матрица переноса

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ D_x & D_y & 1 \end{bmatrix},$$

матрица масштабирования

$$\mathbf{S} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

матрица поворота

$$\mathbf{R} = \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

2. **Специфичное преобразование**

Зеркальное отображение квадрата относительно одной из его вершин – это его поворот на 180 градусов относительно этой вершины.
Соответственно, для данного преобразования необходимо 3 основных:

1) Перемещение квадрата в положение, где одна из его вершин в центре координат: перемещение 4-х вершин на отрицательные координаты выбранной вершины
2) Поворот на 180 градусов
3) Обратное преобразование перемещения

Выводы:

Были изучены аффинные преобразования перемещения, поворота, масштабирования, а также преобразование отражения по одной из вершин. В ходе работы было замечено, что при повороте фигуры на 360 градусов (в исходное положение) происходит незначительное смещение её исходного положения из-за многократного округления чисел с плавающей точкой при переводе координат из однородных в экранные.