**Уфимский Государственный Авиационный Технический Университет**

**Кафедра информатики**

Пояснительная записка к лабораторной работе № 2

Аффинные преобразования и аксонометрическое изображение точки” по дисциплине “ Математические и алгоритмические основы компьютерной графики

Выполнил:

студент группы СТС-109

Шлёнкин Дмитрий

Проверил: Верхотуров

Михаил Александрович

Уфа 2020

Оглавление

[1 Постановка задачи 5](#_Toc39947088)

[1.1 Содержательная постановка задачи 5](#_Toc39947089)

[1.2 Формальная постановка задачи 5](#_Toc39947090)

[2 Структура решения(1-й уровень детализации) 7](#_Toc39947091)

[3 Обзор и анализ методов (2-й уровень детализации) 8](#_Toc39947092)

[3.1 Ввод данных 8](#_Toc39947093)

[3.2 Обработка на аксонометрическом чертеже 8](#_Toc39947094)

[3.2.1 Вычисление координат точки Т при О.П. 8](#_Toc39947095)

[3.2.1 Вычисление координат точки Т при Ц.П. 17](#_Toc39947096)

[3.3 Для комплексного чертежа 20](#_Toc39947097)

[3.4 Вывод 20](#_Toc39947098)

[4 Описание применяемых методов (3-й уровень детализации) 21](#_Toc39947099)

[4.1 Укрупнённый алгоритм 22](#_Toc39947100)

[4.1.1 При Ортогональном проецировании 22](#_Toc39947101)

[4.1.2 При Центральном проецировании 24](#_Toc39947102)

[4.2 Для Комплексного чертежа 25](#_Toc39947103)

[4.3 Визуализация 25](#_Toc39947104)

[5 Деталезированный алгоритм (5 уровень детализации) 26](#_Toc39947105)

[5.1 Ввод 26](#_Toc39947106)

[5.2 Для аксонометрического чертежа 26](#_Toc39947107)

[5.2.1 При Ортогональном проецировании 26](#_Toc39947108)

[5.2.2 При Центральном проецировании 27](#_Toc39947109)

[5.3 Для комплексного чертежа 29](#_Toc39947110)

[5.4 Вывод 29](#_Toc39947111)

[6 Руководство программиста 30](#_Toc39947112)

[6.1 Из чего состоит программа 30](#_Toc39947113)

[6.1.1 Интерфейсная часть 30](#_Toc39947114)

[6.1.2 Прикладная часть 31](#_Toc39947115)

[6.2 Как работает программа 33](#_Toc39947116)

[7 Руководство пользователя 35](#_Toc39947117)

[Заключение 37](#_Toc39947118)

# Постановка задачи

## Содержательная постановка задачи

Необходимо написать программу, которая должна отображать на экране аксонометрический чертеж точки T, а также комплексный чертеж точек T и С, где T–точка трехмерного пространства, С – центр проецирования.

Реализовать динамическое изменение координат (x, y, z) точек Tи C.

Возможность выбора между ортогональным и центральным проецированиями

**Входные данные:**

* Пустой экран.

**Управляющие параметры:**

* Координаты точки T;
* Координаты точки C;
* ортогональным и центральным проецированиями.

**Выходные данные:**

* Изображение точки T на аксонометрическом чертеже;
* Изображение точек T и С на комплексном чертеже.

## Формальная постановка задачи

Свормируем формальную постановку задачи спомощью модели IDEF0.(Рисунок 1.1)



Рисунок 1.1 – Формальная постановка задачи.

# Структура решения(1-й уровень детализации)

Необходимо разбить данную задачу, на подзадачи. Для этого нужно разобраться, что идёт на вход, какие вычисления, что должен увидеть пользователь.

**Ввод**

Ввод трехмерных координат точки T;

Ввод трехмерных координат центра проектирования (точки C);

Выбор типа проецирования (ортогональное или центральное).

**Обработка**

2.1. Вычисление координат точки T и ее проекций на аксонометрическом чертеже

2.1.1. При ортогональном проецировании

2.1.2. При центральном проецировании

2.2. Вычисление координат точек T и С и их проекций на комплексном чертеже

**Вывод**

3.1. Визуализация изображения аксонометрического чертежа

3.2. Визуализация изображения комплексного чертежа

# Обзор и анализ методов (2-й уровень детализации)

## Ввод данных

Данная поздзадача является аналогичной задачей к первой лаб. Работе. Так же необходим ввод координат, необходима их динамика. Из нового – Это переключатель между Ортогональным проецированием и Центральным.

Способы могут быть различные, от передачи Bool’евой переменной до определенного элемента.

Данное переключение удобно сделать через элемент CheсkBox либо Radiobuton, помимо этого, еще соединив их элементом Panel. Элемент Panel дает возможность обьединить элементы (Если один элемент CheckBox или Radiobuton включен, то второй автоматически отключается.)

Удобнее всего будет сделать 2 элемента Radiobuton, соединенные Panel, отвечающие за разный тип проецирования.

**Обобщим:**

* Считываем координаты точки Т (Тx, Ty, Tz) и С( Tcx, Tcy, Tcz).
* Вибираем тип проецирования (Ортогональный - центральный)
* Записываем всё в нужные переменные

## Обработка на аксонометрическом чертеже

### Вычисление координат точки Т при О.П.

**Задание:**

Необходимо построить проекцию точки T на экран, с учетом того, что используется ортогональное проецирование. При условии, что наблюдатель находится в бесконечности на прямой «OC», то есть в произвольной точке с однородными координатами (xc, yc, zc, 0).

**Дано:** Трехмерные координаты точки T (x, y, z)

Трехмерные координаты точки C (xc, yc, zc)

**Найти:** Координаты ортогональной проекции Tэ точки T на плоскость экрана (Txэ, Tyэ).

**Решение:**

Рассмотрим (рисунок 3.1).

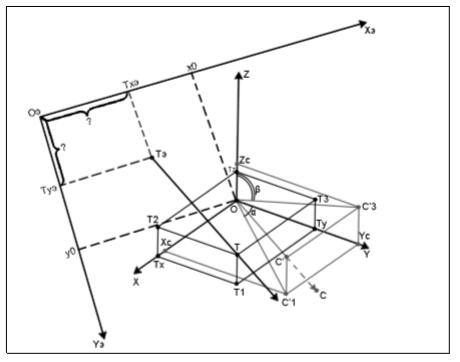


Рисунок 3.1

Дело в том, что для произвольного положения наблюдателя не существует преобразования. Однако среди базовых геометрических преобразований присутствуют 3 преобразования (матрицы) для решения задачи для наблюдателя, находящегося на одной из координатных осей. Вследствие этого возникает необходимость сместить одну оз осей и вектора ОС.

За смещенную ось примем ось OZ, тогда плоскость экрана будет задаваться осями Ox и Oy.

Данное преобразование возможно осуществить через выполнение 2-х поворотов.(Поворот оси OZ и после чего поворот оси Ox’).

1. **Поворот оси OZ по чесовой стрелке на угол Альфа.**

Поворот системы координат вокруг оси Oz на угол α по часовой стрелке Среди базовых преобразований точки присутствуетповорот точки вокруг оси Oz на угол αпротив часовой стрелки. Так как в данном случае поворот системы необходимо осуществить по часовой стрелке, то матрица не изменится.

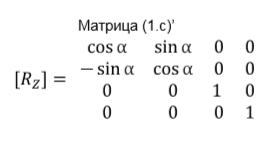
**

Рисунок 3.2 – Матрица поворот OZ

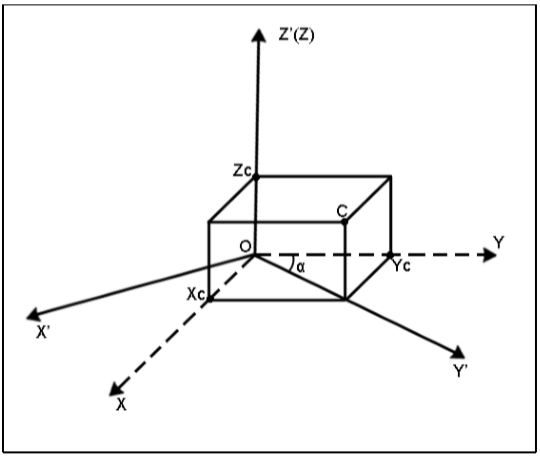
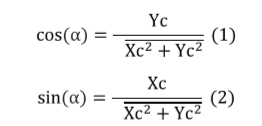


Рисунок 3.3 – Вид после поворота 1

Значения косинуса и синуса из (Рисунок 3.3) следовательно:

 (1)

1. **Поворот Ox’ по чаовой стрелке**

Среди базовых преобразований точки присутствуетповорот точки вокруг оси Ox на угол αпротив часовой стрелки. Так как в данном случае поворот системы необходимо осуществить по часовой стрелке, то матрица не изменится.

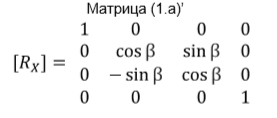


Рисунок 3.4 – Матрица поворот Ox’

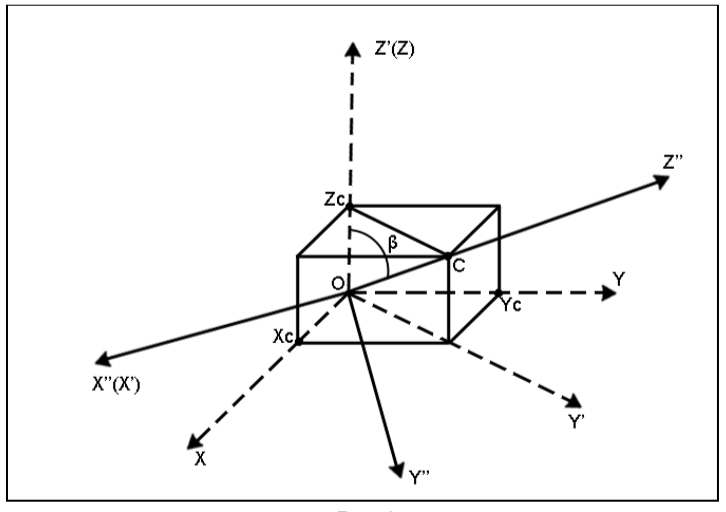
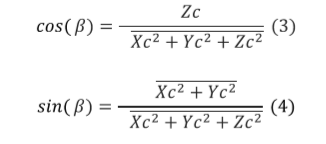


Рисунок 3.5 – Матрица поворот 2

Откуда формулы для Косинуса и Синуса;

 (2)

Частный случай Xc = Yc= Zc= 0 означает, что наблюдатель уже находится на оси Oz, следовательно, необходимости в повороте нет. В целях недопущения ошибки деления на ноль, заменим матрицу на единичную (т.е. cos(α) = 1, sin(α) = 0)

1. **Зеркальное отражение относительно плоскости yOz**

Из (Рисунок 3.5) видно, что ось Ox’’ направлена в противоположную сторону относительно оси OэXэ. В связи с этим применим данное базовое преобразование.

Отразить зеркально не составит труда по матрице:

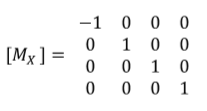


Рисунок 3.6 – Матрица зеркального поворота

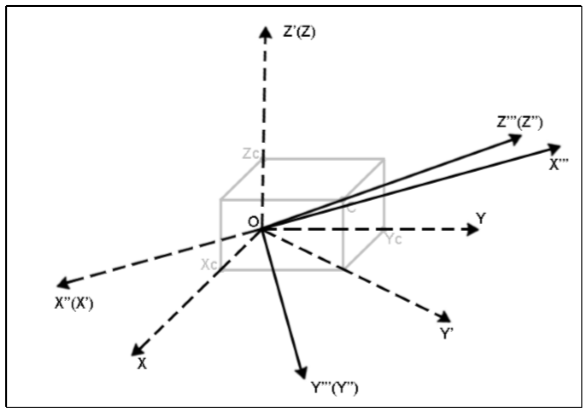


Рисунок 3.7 – Вид после зеркального отражения

1. **Проецирование на плоскость xOy.**

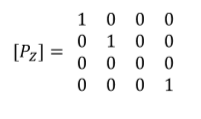


Рисунок 3.8 – Матрица проецирования на XOY

1. **Перенос системы координат**

λ = -x0; μ = -y0; ν = 0, где x0, y0 – экранные координатыточки O.

Среди базовых геометрических преобразований присутствует перенос точки на вектор(λ, μ, ν).Так как в нашем случае необходимо осуществить перенос системы координат, то координаты вектора переноса поменяют знаки (-λ, -μ, -ν). Подставим исходные значения для каждой координаты:

(-λ, -μ, -ν) = (-(-x0), -(-y0), 0) = (x0, y0, 0) (3)

Исходная матрица имеет вид:

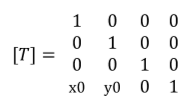


Рисунок 3.9 – Матрица переноса системы координат

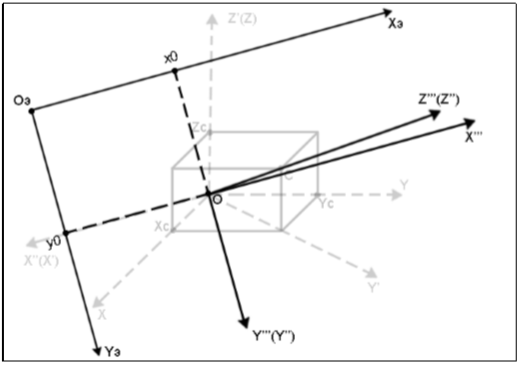


Рисунок 3.10 – После переноса системы координат

В конечном итоге, нужно выразить матрицу, на которую нужно будет умножить каждую вектор-строку. Итоговая матрица А будет равна поочередному произведению всех вышеперечисленных матриц.

 (4)

Эта матрица нужна для того, чтобы в конеччном итоге умножть вектор-стрроку каждой точки на данную матрицу,при чем на выходе получатся экранные координаты точки Т и ее проекций.

 (5)

*Ta Tb*

Где Тb это вектор-строка точки Т, Ta – это экранные координаты точки Т.

**Обобщим:**

1. Проверим, не совпадает ли точка С с началом координат (точкой O) (т.е. не имеет ли точка С однородные координаты (0, 0, 0, 0)). Если это так, то задача неразрешима, т.к. нет вектора СО, соответственно невозможно определить плоскость экрана(проецирования) и т.д.

2. Подготовка матрицы поворота системы координат вокруг оси Oz на угол α по часовой стрелке (матрица (Рисунок 3.2)) (подсчет cos(α) и sin(α) по формулам (1)). Также проверим, не равны ли координаты Xc, Yc нулю, для недопущения ошибки при делении на ноль.

3. Подготовка матрицы поворота системы координат вокруг оси Ox на угол β по часовой стрелке (матрица Рисунок 3.4 (подсчет cos(β) и sin(β) по формулам (2)). Также проверим, не равны ли координаты Xc, Yc, Zc нулю, для недопущения ошибки при делении на ноль.

4. Задание матрицы отражения оси Ox относительно плоскости YOZ (матрица Рисунок 3.6).

5. Подготовка матрицы проецирования на плоскость XOY (при расположении наблюдателя, смотрящего в направлении начала координат, в бесконечности на оси Oz (в точке (0,0,1,0))) (матрица Рисунок 3.8).

6. Осуществление сдвига осей координат на вектор (x0, y0, 0) , полученный по формуле (3)(матрица Рисунок 3.9).

7. Формирование матрицы итогового сложного геометрического преобразования: ортогонального проецирования для наблюдателя в точке C, формула (4)

8. Осуществление данного сложного преобразования к 11-ти точкам: To, T1o, T2o, T3o, Txo, Tyo, Tzo, Toxo, Toyo, Tozo, Too, а именно: умножение векторстроку точек на матрицу итогового сложного преобразования, формула (5) и получение новых точек после преобразования: T’o, T’1o, T’2o, T’3o, T’xo, T’yo, T’zo, T’oxo, T’oyo, T’ozo, T’oo соответственно.

Так как чертёж находится всегда в 1-ом октанте, то не сучествуют отрицательных концов осей, соответственно, не производим над ними действия.

### Вычисление координат точки Т при Ц.П.

**Задание:**

Необходимо построить проекцию точки Tна экран с учетом того, что используется центральное проецирование, при условии, что наблюдатель находится на конечном расстоянии от экрана.

**Дано:**

Трехмерные координаты точки T (x, y, z) Трехмерные координаты точки C (xc, yc, zc)

**Найти:** Координаты ортогональной проекции Tэ точки T на плоскость экрана (Txэ, Tyэ).

**Решение**:

Для того, чтобы построитьперспективное изображение, используется центральное проецирование, при котором центр проекции (точка С) находится на конечном расстоянии от экрана, в отличие от того, что было при ортогональном проецировании.

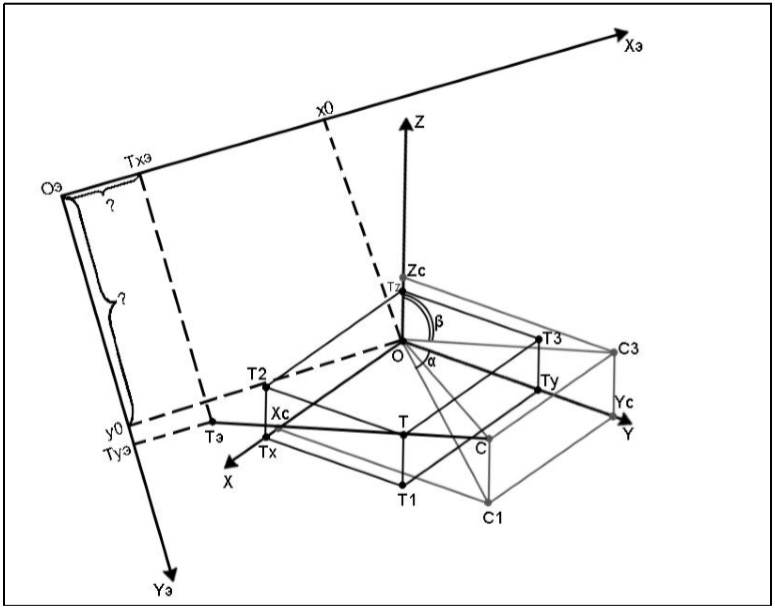
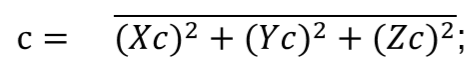


Рисунок 3.11 – Макет центрального проецирования.

Для центрального проецирования последовательность базовых преобразований выглядит почти так же, как для ортогонального проецирования, только к ней необходимо дополнительно добавить еще одно базовое геометрическое преобразование перед проецированием на плоскость XOY:

Перспективное преобразование (При расположении наблюдателя на оси Oz в точке (0, 0, c, 1)). Из рисунка можно без труда найти с (длина радиус-вектора центра проецирования):

 (6)

Рассмотрим базовое преобразование [Cz]. Данному геометрическому преобразованию соответствует матрица:

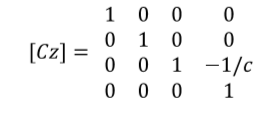


Рисунок 3.12 – Матрица Центрального проецирования

Так же «чтобы масштабный множитель был равен 1 – нужн привести координаты (Формула 7)

 (7)

То есть, сложное геометрическое преобразование будет выглядеть следующим образом:

 (8)

В конечном итоге, при помощи рассмотренной матрицы сможем найти центральную проекцию любой точки. Для осуществления этого необходимо умножить (справа) вектор-строку точки на матрицу , результатом чего будет вектор-строка искомой точки:

 (9)

*Ta Tb*

Нельзя не сказать о проверке для Центрального проецирования. Проекцию невозможно построить при:

1: Точка С совпадает с началом координат (точкой О).В данном случае не определен вектор направления проецирования OC.

2: Точка Tсовпадает с точкой С. В этом случае не определен вектор CT.

3: В случаях когда вектор CT не будет пересекать плоскость экрана.

Для этого воспользоваться нужно формулой для проверки:

 (10)

4: Выход за границы экрана. (Если какая либо точка вышла за пределы экрана)

То есть если: (XT< 0 или XT>width) или (YT< 0 или YT>height) То рисунок будет “вылезать” за границы экрана.

**Обобщенный алгоритм**:

1) Проверим, не совпадает ли точка С с началом координат (точкой O) (т.е. не имеет ли точка С однородные координаты (0, 0, 0, 0).Если это так, то нет вектора СО и задача неразрешима.

2) Проверим, не совпадает ли точка T с точкой С. Если это так, то не будет определен вектор CT и задача неразрешима.

3) Проверим, пересекает ли вектор CT плоскость экрана. Если это так, то задача неразрешима.

4) Подготовка матрицы [Rz] (проверка деления на 0) (аналогично шагу 2) обобщенного алгоритма пункта 3.2.1)

5) Подготовка матрицы [Rx] (проверка деления на 0) (аналогично шагу 3) обобщенного алгоритма пункта 3.2.1)

6) Задание матрицы [Mx] (аналогично шагу 4) обобщенного алгоритма пункта 3.2.1)

7) Подготовка матрицы [Cz] перспективного преобразования (При расположении наблюдателя на оси Oz в точке (0, 0, c, 1) матрица Рисунок 3.12)

8) Подготовка матрицы [Pz] (аналогично шагу 5) обобщенного алгоритма пункта 3.2.1)

9) Подготовка матрицы [T] (аналогично шагу 6) обобщенного алгоритма пункта 3.2.1)

10) Формирование матрицы итогового сложного геометрического преобразования: центрального проецирования для наблюдателя в точке C, формула (7)

11) Осуществление данного сложного преобразования к 11-ти точкам: To, T1o, T2o, T3o, Txo, Tyo, Tzo, Toxo, Toyo, Tozo, Too, а именно: умножение вектор-строку точек на матрицу итогового сложного преобразования, формула (8) и получение новых точек после преобразования: TT’o, TT’1o, TT’2o, TT’3o, TT’xo, TT’yo, TT’zo, TT’oxo, TT’oyo, TT’ozo, TT’ooсоответственно.

12) Приведение однородных координат точек (TT’o, TT’1o, TT’2o, TT’3o, TT’xo, TT’yo, TT’zo, TT’oxo, TT’oyo, TT’ozo, TT’) (с масштабным множителем w не равным 1) к “обычным” координатам (MT’o, MT’1o, MT’2o, MT’3o, MT’xo, MT’yo, MT’zo, MT’oxo, MT’oyo, MT’ozo, MT’), формула (7)

13) Проверка выхода точек за границы экрана. Если ни одна из точек не “вышла” за границы экрана, то переходим к следующему пункту. В противном случае выводим соответствующую ошибку.

## Для комплексного чертежа

Данная подзадача является аналогичной той, что рассматривалась ранее в 1-й лабораторной работе.

## Вывод

Данная подзадача является аналогичной той, что рассматривалась ранее в 1-й лабораторной работе.

# Описание применяемых методов (3-й уровень детализации)

Введем необходимые типы данных:

* CPoint4D(для 4D-точки)
* CPoint3D(для 3D-точки)
* CPoint2D (для 2D-точки)

Типы структур:

structCPoint4D {

x; //Координата точки по Ox

y; //Координата точки по Oy

z; //Координата точки по Oz

w;//масштабный множитель

}

struct CPoint3D {

x; //Координата точки по Ox

y; //Координата точки по Oy

z; //Координата точки по Oz

}

struct CPoint2D {

x; //Координата точки по Ox

y; //Координата точки по Oy

}

Для хранения однородных координат каждой точки будем использовать массив “хранилищеОднородныхТочек”[]. После преобразования (умножения справа на матрицу сложного преобразования, полученной по формуле (5)) новые однородные координаты точек будем хранить в массиве “хранилищеПреобразованныхОднородныхТочек”[].

**Общие переменные:**

xT, yT, zT - установленные значения соответствующих ползунковых переключателей для координат точки T (x, y, zсоответственно).

xС, yС, zС - установленные значения соответствующих ползунковых переключателей для координат точки С (x, y, zсоответственно).

widht, height – ширина и высота экрана

хранилище3DТочекT[T, T1, T2, T3, Tx, Ty, Tz, Tox, -Tox, Toy, -Toy, Toz, -Toz, To]– массив начальных 3D-координат (для точки T).

хранилище3DТочекС[С, С1, С2, С3, Сx, Сy, Сz, Сox, -Сox, Сoy, -Сoy, Сoz, -Сoz, Сo]–массив начальных 3D-точек(для точки C).

хранилище4DТочекT[To, T1o, T2o, T3o, Txo, Tyo, Tzo, Toxo, Toyo, Tozo, Too] – массивначальныходнородных координат (для точки T).

**хранилище4DТочекС[Co, C1o, C2o, C3o, Cxo, Cyo, Czo, Coxo, Coyo, Cozo, Coo] – массивначальныходнородных координат (для точки C).**

**Для аксонометрического чертежа:**

Переменные для аксонометрического чертежа:

хранилищеОднородныхТочек[] –массив, для хранения однородных координат, полученных из введенных 3D-координат.

хранилищеПреобразованныхОднородныхТочек[] - массив, для хранения новых однородных координат, полученных после преобразования старых.

хранилище2DТочек[]- массив, для хранения 2D-точек

cosRz, sinRz–переменные, хранящие значения cos и sin для матрицы [Rz]

cosRx, sinRx–переменные, хранящие значения cos и sin для матрицы [Rx]

lenAxisX, lenAxisY–переменные, длины полуосей Oxи Oyсоотвественно

**Переменные комплексного чертежа:**

Данный пункт тому, что рассматривался ранее, в 1-й лабораторной работе.

## Укрупнённый алгоритм

### При Ортогональном проецировании

1) Проверим, совпадение точки С с началом координат (точкой O) (т.е. выполняется ли условие xC = 0 и yC=0 и zC=0)

Если это так, то задача неразрешима

2) Подготовка матрицы поворота системы координат вокруг оси Oz на угол α по часовой стрелке (cos(α) = cosRz, sin(α) = sinRz).

Матрица [Rz] = {cosRz sinRz 0 0

-sinRz cosRz 0 0

0 0 1 0

0 0 0 1}

Где: cosRz = 1; sinRz = 0, в случае, если координаты xC и yC одновременно равны 0. Если не выполняется данное условие, то:

cosRz = yC/ ; sinRz = xC/

3) Подготовка матрицы поворота системы координат вокруг оси Ox на угол β по часовой стрелке(cos(β) = cosRx, sin(β) = sinRx)

Матрица [Rx] = {1 0 0 0

0 cosRx sinRx 0

0 –sinRx cosRx 0

0 0 0 1}

Где: cosRx= zC/ ; sinRx = / ; во всех случаях (кроме того, когда .xC, yCи zC одновременно равны 0). Иначе Сos = 1, Sin = 0.

4) Задание матрицы отражения оси Ox относительно плоскости YOZ Матрица [Mx] = {-1 0 0 0

0 1 0 0

0 0 1 0

0 0 0 1}

5) Подготовка матрицы проецирования на плоскость XOY (при расположении наблюдателя, смотрящего в направлении начала координат, в бесконечности на оси Oz (в точке (0,0,1,0)))

Матрица [Pz] = {1 0 0 0

0 1 0 0

0 0 0 0

0 0 0 1}

6) Осуществление сдвига осей координат на вектор (-x0, -y0, 0). (x0 = -lenAxisX, y0 = -lenAxisY (формула (3)))

Матрица [T] = {1 0 0 0

0 1 0 0

0 0 1 0

lenAxisX lenAxisY 0 1}

7) Формирование матрицы итогового сложного геометрического преобразования: ортогонального проецирования для наблюдателя в точке C, формула (4). Матрица [A] = [Rz]\* [Rx]\* [Mx]\* [Pz]\* [T];

8) Осуществление данного сложного преобразования в цикле, к каждой из точек из массива хранилищеОднородныхТочек[To, T1o, T2o, T3o, Txo, Tyo, Tzo, Toxo, Toyo, Tozo, Too], а именно: умножение вектор-строку точек на матрицу итогового сложного преобразования, формула (5), записывая результаты в массив хранилищеПреобразованныхОднородныхТочек[T’o, T’1o, T’2o, T’3o, T’xo, T’yo, T’zo, T’oxo, T’oyo, T’ozo, T’oo].

### При Центральном проецировании

1) Проверим, не совпадает ли точка С с началом координат (точкой O) (т.е. не имеет ли точка С однородные координаты (0, 0, 0, 0).Если это так, то задача неразрешима.

2) Проверим, не совпадает ли точка T с точкой С. Если это так, то не будет определен вектор CT и задача неразрешима.

3) Проверим, пересекает ли вектор CT плоскость экрана. Если это так, то проекция отсутствует. В противном случае переходим дальше по алгоритму.

4) Подготовка матрицы [Rz] (аналогично шагу 2 укрупненного алгоритма пункта 3.2.1)

5) Подготовка матрицы [Rx] (аналогично шагу 3) укрупненного алгоритма пункта 3.2.1)

6) Задание матрицы [Mx] (аналогично шагу 4) укрупненного алгоритма пункта 3.2.1)

7) Подготовка матрицы [Cz] перспективного преобразования (При расположении наблюдателя на оси Oz в точке (0, 0, c, 1) (матрица Рисунок 3.12)

Матрица [Сz] = {1 0 0 0

0 1 0 0

0 0 1-1/c

0 0 0 1}

с = (формула (6))

8) Подготовка матрицы [Pz] (аналогично шагу 5) укрупненного алгоритма пункта 3.2.1)

9) Подготовка матрицы [T] (аналогично шагу 6) укрупненного алгоритма пункта 3.2.1)

10) Формирование матрицы итогового сложного геометрического преобразования: центрального проецирования для наблюдателя в точке C, формула (7) Матрица [Ac] = [Rz]\* [Rx]\* [Mx]\* [Cz]\* [Pz]\* [T];

11) Осуществление данного сложного преобразования в цикле, к каждой из точек из массива хранилищеОднородныхТочек[To, T1o, T2o, T3o, Txo, Tyo, Tzo, Toxo, Toyo, Tozo, Too], а именно: умножение вектор-строку точек на матрицу итогового сложного преобразования, формула (8), записывая результаты в массив хранилищеПреобразованныхОднородныхТочек[T’o, T’1o, T’2o, T’3o, T’xo, T’yo, T’zo, T’oxo, T’oyo, T’ozo, T’oo]

12) Приведение однородных координат (с масштабным множителем w не равным 1) каждой точки из массива хранилищеПреобразованныхОднородныхТочек[TT’o, TT’1o, TT’2o, TT’3o, TT’xo, TT’yo, TT’zo, TT’oxo, TT’oyo, TT’ozo, TT’oo] к “обычным” координатам в цикле, формула (9), записывая результаты в массив хранилище2DТочек[MT’o, MT’1o, MT’2o, MT’3o, MT’xo, MT’yo, MT’zo, MT’oxo, MT’oyo, MT’ozo,M T’oo]

13) Проверка выхода точек за границы экрана. Если ни одна из точек не “вышла” за границы экрана, то переходим к следующему пункту. В противном случае выводим соответствующую ошибку.

## Для Комплексного чертежа

Данная подзадача является аналогичной той, что рассматривалась ранее в 1-й лабораторной работе.

## Визуализация

Данная подзадача является аналогичной той, что рассматривалась ранее в 1-й лабораторной работе.

# Деталезированный алгоритм (5 уровень детализации)

**Подготовка функций**

*Функция\_проверки\_проецирования(RadioButton)* ***Вернуть***isOrhogonal

*{*

Если *RadioButton.активирована*

То isOrhogonal = 1

иначе

isOrhogonal = 0

*}*

Функция: *Проверка\_вектора\_ОС(* Xc, Yc, Zc*)*:

Если Xc == 0 и Yc == 0 и Zc == 0

{

Сообщение: "Ошибка: Не задан вектор С"

вернуть false;

}

иначе

{

вернуть true;

}

*Функция\_умножения\_матриц(Матрица\_1,матрица\_2)*

***Вернуть*** Public\_Matrix;

*{*

Цикл от 0 до Кол-ва строк в массиве\_1(row)

{

Цикл от 0 до кол-ва столбцов в массиве\_2(col)

{

Цикл от 0 до количества столбцов в массиве\_1(i)

{

Public\_Matrix[row, col] += Массив\_1[row, i] \* Массив\_2[i, col];

}

}

}

*}*

*Функция\_вычисления\_Орт\_проец\_(Point4D, Xc,Yc,Zc,CenterX, CenterY)* ***Вернуть*** *Point4D*

*{*

Если Yc == 0 и Xc == 0

{

Cos\_A\_Oz = 1;

Sin\_A\_Oz = 0;

}

Иначе

{

Cos\_A\_Oz = Yc / Sqrt(Pow(Xc, 2) + Pow(Yc, 2));

Sin\_A\_Oz = Xc / Sqrt(Pow(Xc, 2) + Pow(Yc , 2));

}

если Yc == 0 и Xc == 0 и Zc == 0

{

Cos\_B\_Ox = 1;

Sin\_B\_Ox = 0;

}

иначе

{

Cos\_B\_Ox = Zc / Sqrt(Pow(Xc, 2) + Pow(Yc, 2) + Pow(Zc, 2));

Sin\_B\_Ox = Sqrt(Pow(Xc, 2) + Pow(Yc , 2)) / Sqrt(Pow(Xc, 2) + Pow(Yc, 2) + Pow(Zc, 2));

}

matrixZ = (Cos\_A\_Oz, Sin\_A\_Oz, 0, 0,

-Sin\_A\_Oz, Cos\_A\_Oz, 0, 0,

0, 0, 1, 0,

0, 0, 0, 1)

matrixX = (1, 0, 0, 0,

0, Cos\_B\_Ox, Sin\_B\_Ox, 0,

0, -Sin\_B\_Ox, Cos\_B\_Ox, 0,

0, 0, 0, 1)

}

matrix\_Mirrors =(-1, 0, 0, 0,

0, 1, 0, 0,

0, 0, 1, 0,

0, 0, 0, 1)

matrix\_Pz =(1, 0, 0, 0,

0, 1, 0, 0,

0, 0, 0, 0,

0, 0, 0, 1)

matrix\_T = (1, 0, 0, 0,

0, 1, 0, 0,

0, 0, 1, 0,

CenterX, CenterY, 0, 1)

Point4D = *Функция\_умножения\_матриц* (*Point4D*, matrixZ, matrixX, matrix\_Mirrors, matrix\_Pz, matrix\_T)

***Вернуть*** *Point4D: Функция\_сложного\_преобразования\_для\_Ц\_П\_ (Point4D,Xc,Yc,Zc, CenterX, CenterY)*

{

C =Sqrt(Pow(Xc, 2) +Pow(Yc, 2) +Pow(Zc, 2));

Если Yc == 0 и Xc == 0

{

Cos\_A\_Oz = 1;

Sin\_A\_Oz = 0;

}

Иначе

{

Cos\_A\_Oz = Yc / Sqrt(Pow(Xc, 2) + Pow(Yc, 2));

Sin\_A\_Oz = Xc / Sqrt(Pow(Xc, 2) + Pow(Yc , 2));

}

если Yc == 0 и Xc == 0 и Zc == 0

{

Cos\_B\_Ox = 1;

Sin\_B\_Ox = 0;

}

иначе

{

Cos\_B\_Ox = Zc / Sqrt(Pow(Xc, 2) + Pow(Yc, 2) + Pow(Zc, 2));

Sin\_B\_Ox = Sqrt(Pow(Xc, 2) + Pow(Yc , 2)) / Sqrt(Pow(Xc, 2) + Pow(Yc, 2) + Pow(Zc, 2));

}

matrixZ = (Cos\_A\_Oz, Sin\_A\_Oz, 0, 0,

-Sin\_A\_Oz, Cos\_A\_Oz, 0, 0,

0, 0, 1, 0,

0, 0, 0, 1)

matrixX = (1, 0, 0, 0,

0, Cos\_B\_Ox, Sin\_B\_Ox, 0,

0, -Sin\_B\_Ox, Cos\_B\_Ox, 0,

0, 0, 0, 1)

}

matrix\_Mirrors =(-1, 0, 0, 0,

0, 1, 0, 0,

0, 0, 1, 0,

0, 0, 0, 1)

matrixCz = (1, 0, 0, 0,

0, 1, 0, 0,

0, 0, 1, -1 / C,

0, 0, 0, 1);

matrix\_Pz =(1, 0, 0, 0,

0, 1, 0, 0,

0, 0, 0, 0,

0, 0, 0, 1)

matrix\_T = (1, 0, 0, 0,

0, 1, 0, 0,

0, 0, 1, 0,

CenterX, CenterY, 0, 1)

MassCenterProection *= Функция\_умножения\_матриц* (*Point4D*, matrixZ, matrixX, matrix\_Mirrors, matrixCz, matrix\_Pz, matrix\_T)

Цикл от 0 до 11

{

Point2D [i, 0] = MassCenterProection[i, 0] / MassCenterProection[i, 3];

Point2D [i, 1] = MassCenterProection[i, 1] / MassCenterProection[i, 3];

}

*}*

## Ввод

Если *1 = Функция\_проверки\_проецирования(RadioButton), то*

Ортогональное проецирование,

Иначе

Центральное

Остальное в данном пункте аналогично тому, что рассматривалось ранее в 1-й лабораторной работе.

## Для аксонометрического чертежа

### При Ортогональном проецировании

Если *Проверка\_вектора\_ОС(*Xc, Yc, Zc*) = true, то выполняется следующий шаг*

Создание и умножение матриц и получение массива, с масштабным множителем 1 для Ортогонального чертежа:

Array2DPointsOrtogonal = *Функция\_вычисления\_Орт\_проец\_( Point4D, Xc,Yc,Zc,CenterX, CenterY)*

### При Центральном проецировании

Функция *Проверка\_вектора\_ОС(*Xc, Yc, Zc*) = true, то выполняется следующий шаг*

Подготовка матриц и умножение всех матриц, а так же умножение вектор строки на матрицу итогового пребразования: *Функция\_сложного\_преобразования\_для\_Ц\_П\_( Point4D,Xc,Yc,Zc, CenterX, CenterY).*

## Для комплексного чертежа

Данная подзадача является аналогичной той, что рассматривалась ранее в 1-й лабораторной работе.

## Вывод

Данная подзадача является аналогичной той, что рассматривалась ранее в 1-й лабораторной работе.

# Руководство программиста

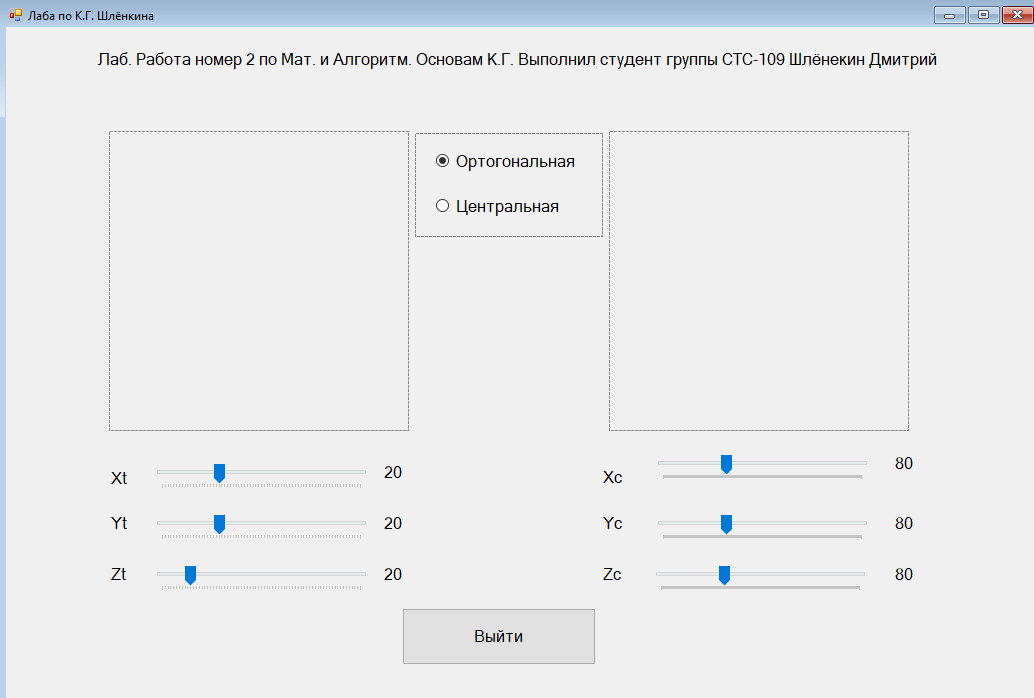
Рассмотрим реализацию данной лабораторной работы на платформе Microsoft .NETFramework, в среде разработки VisualStudio, на языке программирования C#.

## Из чего состоит программа

Программа состоит из 2-ух частей: интерфейсной и прикладной

### Интерфейсная часть

Для отображения необходимых изображений используются объекты класса “PictureBox”. Для ввода данных используются ползунковые переключатели, объекты класса “TrackBar”, и радиокнопки, объекты класса “RadioButton”. Для вывода кнопки “Выйти из программы” используется объект класса “Button”. Для вывода необходимого текста используются объекты класса “Label”.Рисунок 6.1



RadioButton1

RadioButton2

Button1

Label10

Label11

Label12

TrackBar6

TrackBar5

TrackBar4

TrackBar1

TrackBar2

TrackBar3

PictureBox1

PictureBox2

Label13

Label7

Label8

Label9

Label4

Label5

Label6

Label3

Label1

Label2

Рисунок 6.1 – Интерфейс

* Label

Label1 – подпись для координаты X

Label2 - подпись для координаты Y

Label3 - подпись для координаты Z

Label4- подпись Вывода для координаты X

Label5 - подпись Вывода для координаты Y

Label6 - подпись Вывода для координаты Z

Label7 - подпись угла между горизонтальным вектором, направленным влево и положительным направлением Ox

Label8 - подпись угла между горизонтальным вектором, направленным влево и положительным направлением Oy

Label9 - подпись угла между горизонтальным вектором, направленным влево и положительным направлением Oz

Label10 - подпись вывода угла между горизонтальным вектором, направленным влево и положительным направлением Ox

Label11 - подпись вывода угла между горизонтальным вектором, направленным влево и положительным направлением Oy

Label12 - подпись вывода угла ввод угла между горизонтальным вектором, направленным влево и положительным направлением Oz

Label13 – Подпись автора

* PictureBox

PictureBox1 - вывод «пространственного» чертежа

PictureBox2 вывод комплексного чертежа

* TrackBar

TrackBar1 - Ввод для координаты X

TrackBar2 - Ввод для координаты Y

TrackBar3 - Ввод для координаты Z

TrackBar4 - Ввод угла между горизонтальным вектором, направленным влево и положительным направлением Ox

TrackBar5 - Ввод угла между горизонтальным вектором, направленным влево и положительным направлением Oy

TrackBar6 - Ввод угла между горизонтальным вектором, направленным влево и положительным направлением Oz

* Button

Button1 – Закрытие программы

* RadioButton

RadioButton1 – Выбор Ортогонального проецирования

RadioButton2 – Выбор Центрального проецирования

### Прикладная часть

Программа состоит из одного внешнего класса

* Form1

И 9-ти внутренних классов

* CalculateMatrix
* Create\_Complex\_point
* Draw\_Class
* Matrix
* Calculate\_Hard\_Multiply
* Input\_Class
* CPoint4D
* CPoint3D
* CPoint2D

**Внешний класс Form1:**

Внешний класс приложения, служит для вызова процедур с внутреннего класса.

Содержит:

private void Form1\_Paint(object sender, PaintEventArgs e)

rivate void trackBar1\_Scroll(object sender, EventArgs e)

rivate void trackBar2\_Scroll(object sender, EventArgs e)

rivate void trackBar3\_Scroll(object sender, EventArgs e)

rivate void trackBar4\_Scroll(object sender, EventArgs e)

rivate void trackBar5\_Scroll(object sender, EventArgs e)

rivate void trackBar6\_Scroll(object sender, EventArgs e)

private void button1\_Click(object sender, EventArgs e)

**Внутренние классы:**

* CalculateMatrix – Содержит функцию умножения матриц

Функция умножения матриц:

public double[,] calculateMatrix(double[,] matrix\_A, double[,] matrix\_B)

* Create\_Complex\_point

Функция для пересчёта точек для КЧ

public (int, int)[] CreateStorageComplex2D((int, int, int)[] baseStorage3DPoints, double CenterX, double CenterY)

* Draw\_Class

Отрисовка осей комплексного чертежа:

public void DrawAxis\_complex(Graphics grr, Pen pen, (int, int)[] arr)

Отрисовка линий связи для комплексного чертежа

public void DrawLines\_complex(Graphics grr, Pen pen, (int, int)[] arr)

Отрисовка точек для К.Ч.

public void Paint\_Dots\_complex(Graphics grr, Pen pen, (int, int)[] arr)

Отрисовка дуги между точками

public void Paint\_Arc(Graphics grr, Pen pen, double dC, (int, int)[] arr,double CenterX,double CenterY)

Отрисовка осей для Аксонометрического чертежа

public void Paint\_Axis3D(Graphics grr, Pen Pen\_Axis,double[,] Array2DPointsOrtogonal)

Отрисовка линий связи Аксонометрического чертежа

public void Paint\_Lines\_Proection(Graphics grr, Pen Pen\_Lines, double[,] Array2DPointsOrtogonal)

Отрисовка точек Аксонометрического чертежа

public void Paint\_dots\_3D(Graphics grr, Pen Pen\_Axis, double[,] Array2DPointsOrtogonal)

* Matrix

Создание матрицы 4Х4

public Matrix(double a11, double a12, double a13, double a14, double a21, double a22, double a23, double a24, double a31, double a32, double a33, double a34, double a41,

double a42, double a43, double a44)

* Calculate\_Hard\_Multiply

Функция подсчета и вывода массива для Ортогонального проецирования:

public double[,] Calculate\_Ortogonal(double[,] Array4DPointsOrtogonal,double Xc,double Yc, double Zc,double CenterX,double CenterY)

Функция пересчёта и вывода массива для Центрального проецирования:

public double[,] Array\_C\_P(double[,] mass, double Xc, double Yc, double Zc, double CenterX, double CenterY)

* Input\_Class

Функция ввода для формирования массивов:

public void Input(PictureBox pc, int x, int y, int z, int Xc, int Yc, int Zc, RadioButton radioButton)

Функция отрисовки Ортогонального чертежа:

public void Paint\_Ortogonal(PictureBox pc)

Функция отрисовки для Центральной Проекции:

public void PaintComplex(PictureBox pc)

Функция проверки центра проецирования:

private bool Prove\_center(int Xc, int Yc, int Zc,int x,int y,int z, PictureBox pc)

* **CPoint4D** - Внутренний класс программы, описывающий 4Dточку и содержащий следующие поля и методы:

int x; поле, содержащее координату трехмерной точки по оси Ox;

int y; поле, содержащее координату трехмерной точки по оси Oу;

int z; поле, содержащее координату трехмерной точки по оси Oz;

double w; поле, содержащее масштабный множитель;

* **CPoint3D** - Внутренний класс программы, описывающий 3Dточку и содержащий следующие поля и методы:

int x; поле, содержащее координату трехмерной точки по оси Ox;

int y; поле, содержащее координату трехмерной точки по оси Oу;

int z; поле, содержащее координату трехмерной точки по оси Oz;

CPoint3D(int x, int y, int z)

Конструктор класса с 3-мя параметрами: координаты трехмерной точки по каждой из трех осей координат.

* **CPoint2D**

Внутренний класс программы, описывающий 2D точку и содержащий следующие поля и методы:

int x; поле, содержащее координату двумерной точки по оси Ox;

int y; поле, содержащее координату двумерной точки по оси Oу;

CPoint2D(intx, inty)

конструктор класса с 2-мя параметрами: координаты трехмерной точки по каждой из двух осей координат.

Покажем на функциональной схеме, как взаимодействуют между собой функции и методы, а также как происходит их взаимодействие с элементами интерфейса;

Trackbar6

Trackbar5

Trackbar4

Trackbar3

Trackbar2

Trackbar1

**Input()**

Xt = trackBar1.value

Xc = Trackbar4.value

….

*Функция\_проверки\_проецирования()*

*Функция\_вычисления\_Орт\_проец()*

*Проверка\_вектора\_ОС*

*Функция\_сложного\_преобразования\_для\_Ц\_П\_(Проверка\_центрального\_проец\_()*

**createstorageSpacial2D()**

DrawPoints**()**

**Paint\_Complex()**

DrawArc()

DrawLines**()**

DrawPoints**()**

*Calculate\_3D\_in\_2D\_Co\_1 ()*

*Calculate\_3D\_in\_2D\_Co\_2 ()*

*Calculate\_3D\_in\_2D\_Co\_2*

DrawLines**()**

**Paint\_Ortogonal\_and\_С\_P()**

PictureBox2

PictureBox1

Label12

Label11

Label10

Label4

Label5

Label6

## Как работает программа

При движении ползунка – меняются соответствующие переменные, происходит пересчёт по формулам, и происходит вывод нового рисунка – то есть – рисунок движется(Меняется).

В первую очередь из главной формы вызывается процедура Input(), куда передаются значения с трэкбаров (TrackBar1.Value, TrackBar2.Value, TrackBar3.Value, TrackBar4.Value, TrackBar5.Value, TrackBar6.Value, RadioButton1). В ней формируются переменные, необходимые для работы программы:

double CenterX,double CenterY, byte IsOrtogonal;

Далее идет проверка на выполнение условия о существовании задания: Если *1 = Функция\_проверки\_проецирования(RadioButton), то*

Ортогональное проецирование,

Иначе

Центральное.

\*Массивы

double[,]storageComplex2DPoint\_C;

double[,]storageComplex2DPoint\_T;

Формируются с помощью вызываемых в этой Функции присвоений:

storageComplex2DPoint\_C= create\_Complex\_Point.CreateStorageComplex2D (baseStorage3DPoints\_C, CenterX, CenterY);

storageComplex2DPoint\_T= create\_Complex\_Point.CreateStorageComplex2D (baseStorage3DPoints\_T, CenterX, CenterY);

Где Point.CreateStorageComplex2D – это функция, которая производит все математические действия и возвращает двумерный массив для комплексного чертежа(CPoint2D).

\*Массивы

double[,] Array2DPointsOrtogonal;

double[,] Array2DPointsCenterProection;

Формируются с помощью вызываемых в этой Функции присвоений:

Array2DPointsOrtogonal= calculate\_Hard\_Multiply.Calculate\_Ortogonal (Array4DPointsOrtogonal, Xc, Yc, Zc, CenterX, CenterY);

Array2DPointsCenterProection= calculate\_Hard\_Multiply.Array\_C\_P (Array4DPointsCenter, Xc, Yc, Zc, CenterX, CenterY);

Где Calculate\_Ortogonal и Array\_C\_P – это функции, которые производят все математические действия и возвращают двумерный массив для Ортогонального и центрального проецирования (CPoint4D).

Далее, из главной формы идет вызов процедур для отрисовки чертежей: Paint\_Ortogonal\_фтв\_С\_P(pictureBox1) и Paint\_Complex(pictureBox2). Они содержат методы для отрисовки точек линий связи и отрисовки координатных осей. (DrawPoints(),DrawLines()) – в Paint\_Ortogonal, и (DrawPoints(),DrawLines(), DrawArc()) – в Paint\_Complex.

Происходит отрисовка чертежей с двойной буферизацией. Если переменная IsOrtogonal = 1, то отрисовывается ортогональный чертеж, если переменная IsOrtogonal = 0, отрисовывается центральная проекция.

При изменении положения ползунковых переключателей (Trackbar1, Trackbar2, Trackbar3,Trackbar4, Trackbar5, Trackbar6) вызывается событие Scroll, в обработке которого фиксируются значения, и передаются в функцию Input() и производятся все вышеописанные действия.

Кроме того, событие Scroll вызывает функцию обновления (перерисовки) изображения Refresh() для изображений.

При нажатии на кнопку (Button) button1 (выход из программы), программа завершает свою работу.

# Руководство пользователя

Данная программа предназначена для визуализации аксонометрического чертежа точки T и комплексных чертежей точек T и С.

Рассмотрим Рисунок 7.1

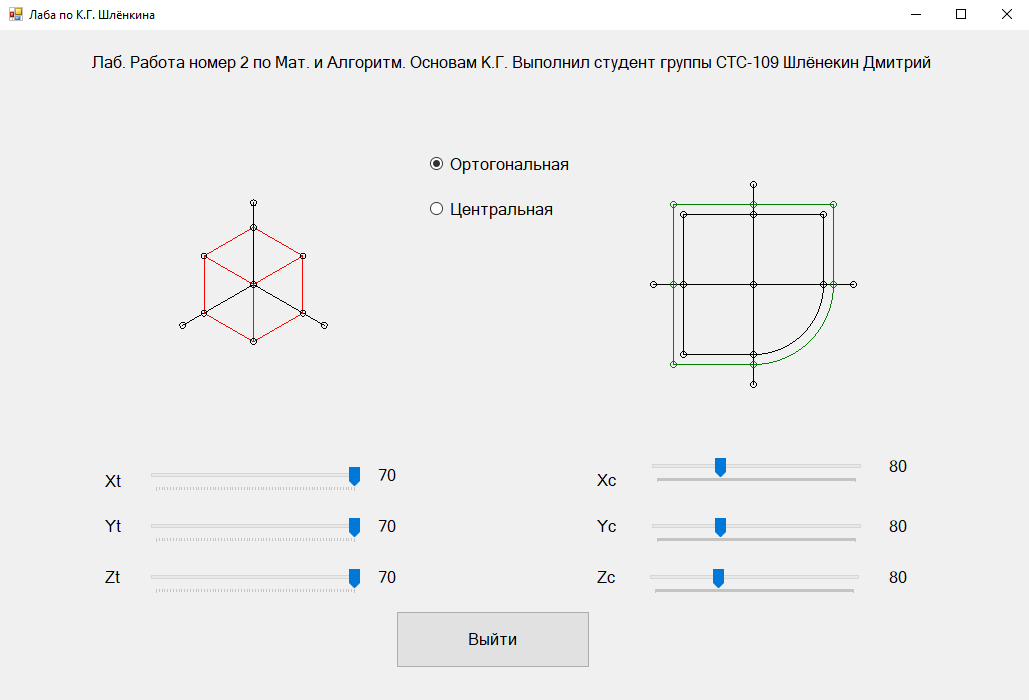


Рисунок 7.1 – Вид пользователя

Левое изображение является изображением аксонометрическогочертежа точки T при ортогональном или центральном проецировании (на рисунке выбрано ортогональное). Выбрать проецирования можно при помощи радиокнопок, расположенных сверху посередине экрана, с соответствующими подписями.

Правое изображение является изображением комплексных чертежей точек Tи С. Соответствующие наименования точек на обоих изображениях находятся справа от самих точек.

Для того, чтобы задать нужные значения координат точек T и С: x, y, z необходимо использовать соответствующие ползунковые переключатели в левой и правой нижних частях экрана соответственно.

Для выхода из программы необходимо нажать кнопку “Выйти из программы”, находящуюся снизу, посередине экрана.

# Заключение

В ходе данной лабораторной работы было рассмотрено создание программы визуализации аксонометрического чертеж (при ортогональном или центральном проецировании) точки T и комплексных чертежей точек T и С, позволяющей динамически менять значения координат точек T и С и выбирать тип проецирования между ортогональным и центральным.