Уфимский Государственный Авиационный Технический Университет

Кафедра информатики

Методические указания к лабораторной работе № 2 “Аффинные преобразования и аксонометрическое изображение точки” по дисциплине “ Математические и алгоритмические основы компьютерной графики ”

Уфа 2020

Краткая теоретическая справка

При решении многих задач вычислительной геометрии часто для задания точки используется следующая запись:

$$\boldsymbol{S} = \boldsymbol{XYZ1}$$ - для 3D-пространства

$$\boldsymbol{S} = \boldsymbol{XY1}$$ - для 2D-пространства

Запишем уравнение плоскости:

$$Ax + By + Cz + D = 0;$$ (общий вид)

$$XYZ1 \begin{matrix} A \\ B \\ C \\ D \end{matrix} = 0$$ (векторный вид)

Запись типа $\boldsymbol{XY1}$ **-** вектор-строка может рассматриваться, как частный случай записи $\boldsymbol{XYW}$ **,**где числа x, y, wназываются **однородными координатами**. Эти три числа однородных координат применяются для обозначения точки в двумерном пространстве.

Для определения термина “однородная” воспользуемся уравнением
aX + bY + c = 0, описывающем прямую линию в двумерном пространстве.

Заменяем X, Yна$\frac{x}{w}$и $\frac{y}{w}$и получим:

$$a*x + b*y + c*w = 0 \quad (1)$$

Уравнение (1) принято называть **однородным**, поскольку оно имеет одинаковую структуру в терминах a*x, b*y, c*w.
Соответственно, закономерно отсюда числа **x, y, w**назовем **однородными координатами** точки (x, y).

Переход из одной прямолинейной координатной системы в трёхмерном пространстве к другой описывается в общем случае следующим образом:

$$\begin{aligned} x' &= \alpha_1 x + \alpha_2 y + \alpha_3 z + \lambda \\ y' &= \beta_1 x + \beta_2 y + \beta_3 z + \mu \\ z' &= \gamma_1 x + \gamma_2 y + \gamma_3 z + \nu \end{aligned}$$

Или в матричном виде:

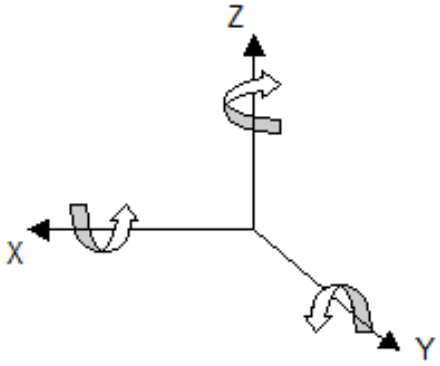
$$x'y'z'\,1 = x\,y\,z\,1 \quad A$$

$$A = \begin{matrix} \alpha_1 & \beta_1 & \gamma_1 & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & 0 \\ \alpha_3 & \beta_3 & \gamma_3 & 0 \\ \lambda & \mu & \upsilon & 1 \end{matrix}$$

Все преобразования системы координат являются либо базовыми, либо сложными (состоящими из последовательности базовых преобразований).

Перечислим все базовые преобразования точек и матрицы, соответствующие им:

1. **Повороты вокруг осей**



a) Вокруг оси Ox на угол φ против часовой стрелки (если смотреть из точки, расположенной в бесконечности на оси Ox, в направлении начала координат):

Матрица 1.a

$$[R_X] = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi & 0 \\ 0 & -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

b) Вокруг оси Oy на угол ψ против часовой стрелки (если смотреть из точки, расположенной в бесконечности на оси Oy, в направлении начала координат):

Матрица 1.b

$$[R_Y] = \begin{matrix} \cos\psi & 0 & -\sin\psi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\psi & 0 & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

c) Вокруг оси Oz на угол χ против часовой стрелки (если смотреть из точки, расположенной в бесконечности на оси Oz, в направлении начала координат):

Матрица 1.c

$$[R_Z] = \begin{matrix} \cos\chi & \sin\chi & 0 & 0 \\ -\sin\chi & \cos\chi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

2. **Растяжение (сжатие)**

Матрица 2

$$[D] = \begin{matrix} \alpha & 0 & 0 & 0 \\ 0 & \beta & 0 & 0 \\ 0 & 0 & \gamma & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

Если $\alpha, \beta, \gamma \in [0,1]$, то сжатие;

Если $\alpha, \beta, \gamma > 1$, то растяжение.

3. **Отражение (зеркалирование)**

a) Относительно плоскости YOZ:

Матрица 3.a

$$[M_X] = \begin{matrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

b) Относительно плоскости XOZ:

Матрица 3.b

$$[M_Y] = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

c) Относительно плоскости XOY:

Матрица 3.c

$$[M_Z] = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

4. **Перенос (сдвиг, перемещение) на вектор (**λ, μ, ν**)**

Матрица 4

$$[T] = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \lambda & \mu & \nu & 1 \end{matrix}$$

5. **Проецирование на плоскость**

   a) На плоскость YOZ (при расположении наблюдателя, смотрящего в направлении начала координат, в бесконечности на оси Ox (в точке (1,0,0,0))):

Матрица 5.a

$$[P_X] = \begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

   b) На плоскость XOZ (при расположении наблюдателя, смотрящего в направлении начала координат, в бесконечности на оси Oy (в точке (0,1,0,0))):

Матрица 5.b

$$[P_Y] = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

   c) На плоскость XOY (при расположении наблюдателя, смотрящего в направлении начала координат, в бесконечности на оси Oz (в точке (0,0,1,0))):

Матрица 5.c

$$[P_Z] = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

6. **Перспективное преобразование**

a) При расположении наблюдателяна оси Ox, в точке (c, 0, 0, 1):

Матрица 6.a

$$[C_X] = \begin{matrix} 1 & 0 & 0 & -1/c \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

b) При расположении наблюдателя на оси Oy в точке (0, c, 0, 1):

Матрица 6.b

$$[C_Y] = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1/c \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

c) При расположении наблюдателя на оси Oz в точке (0, 0, c, 1):

Матрица 6.c

$$[C_Z] = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1/c \\ 0 & 0 & 0 & 1 \end{matrix}$$

Рассмотренные выше геометрические преобразования относятся к базовым (элементарным) геометрическим преобразованиям, все остальные являются сложными, которые реализуются как комбинация базовых, осуществляемых в **строго определенном порядке**, так как в общем случае, геометрические преобразования некоммутативны.

В качестве сложных геометрических преобразований всегда рассматриваются какие-то аналоги базовых, но реализуемых относительно нестандартных расположений тех геометрических объектов, относительно которых производятся соответствующие действия.

Также стоит сказать про относительность базовых геометрических преобразований.
Базовое геометрическое преобразование точки (с сохранением расположения исходной системы координат) соответствует выполнению обратной операции по отношению к преобразованию системы координат.

Например, поворот точки вокруг оси Ox (Oy, Oz) на некоторый уголφ (ψ,χ)по часовой стрелке соответствует повороту системы координат вокруг оси Ox (Oy, Oz) на некоторый уголφ (ψ,χ) против часовой стрелки.

Постановка задачи

Содержательная постановка задачи

Задание на лабораторную работу № 2:

Написать программу, которая должна отображать на экране аксонометрический чертеж точки T, а также комплексный чертеж точек T и C, где T–некоторая точка трехмерного пространства, C – центр проецирования.
Также реализовать динамическое изменение координат (x, y, z) точек Ти Си возможность выбора между ортогональным и центральным проецированиями.

Содержание экрана:

- комплексный чертеж точки T (только первый октант) с изображением проекций T1, T2, T3 и линий связи;
- аксонометрический чертеж точки T (система координат, проекции, линии связи) с учетом расположения центра проецирования в точке C*(x,y,z,w)*;
- ползунковые переключатели (6 шт.) для интерактивного изменения координат *(x,y,z)* точек *T, C* (3 ползунковых переключателя для каждой точки);
- переключатель *w=0; w=1* для смены ортогонального проецирования на центральное.

Динамика:

- при изменении координат точек T и C должны изменяться соответствующие чертежи.

Входные данные:

- Пустой экран.

Управляющие параметры:

- Координаты точки Т;
- Координаты точки С;
- ортогональным и центральным проецированиями.

Выходные данные:

- Изображение точки Тна аксонометрическом чертеже;
- Изображение точек Ти С на комплексном чертеже.

Формальная постановка задачи

Формализуем поставленную задачу при помощи модели в нотации IDEF0, рис. 1

Рис. 1 Контекстная модель в нотации IDEF0.

Структура решения (1-й уровень детализации)

На данном уровне детализации разделим задачу на следующие этапы, которые являются “классическими” при разработке программного обеспечения:

1. **Ввод**
    - Ввод трехмерных координат точки T;
    - Ввод трехмерных координат центра проектирования (точки C);
    - Выбор типа проецирования (ортогональное или центральное).

2. **Обработка**
    2.1. Вычисление координат точки Ти ее проекций на аксонометрическом чертеже
        2.1.1. При ортогональном проецировании
        2.1.2. При центральном проецировании
    2.2. Вычисление координат точек ТиС и их проекций на комплексном чертеже

3. **Вывод**
    3.1. Визуализация изображения аксонометрического чертежа
    3.2. Визуализация изображения комплексного чертежа

В итоге разделил задачу на 6 подзадач (“листьев”), которые подробно рассмотрим на следующем уровне детализации.

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

1. **Ввод**

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

   Стоит отметить единственное отличие: выбор типа проецирования между ортогональным и центральным. В данном случае необходимо выбрать одно из двух, для этого существует несколько элементов управления. В целях наибольшего удобства будем использовать элементы интерфейса – “радио кнопки”. Во многих средах разработки программного обеспечения для операционной системы *Windows* (например, *VisualStudio*) компонент “радио кнопок” представлен классом “RadioButton”.

Обобщенный алгоритм:

1. Считываем значения координат точки Т (X_T, Y_T, Z_T) и точки С(X_C, Y_C, Z_c);
2. Выбираем тип проецирования: ортогональное или центральное;
3. Записываем результаты в соответствующие переменные.

2. Обработка

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

2.1.1. При ортогональном проецировании

Задание:
Необходимо построить проекцию точки Тна экран с учетом того, что используется ортогональное проецирование, при условии, что наблюдатель находится в бесконечности на прямой ОС, то есть в произвольной точке с однородными координатами (x_c, y_c, z_c, 0)

Дано:

- Трехмерные координаты точки Т (x, y, z)
- Трехмерные координаты точки С (x_c, y_c, z_c)

Найти:

- Координаты ортогональной проекцииТэ точки Тна плоскость экрана (Тхэ, Туэ).

Решение:

Рассмотрим рисунок 2:

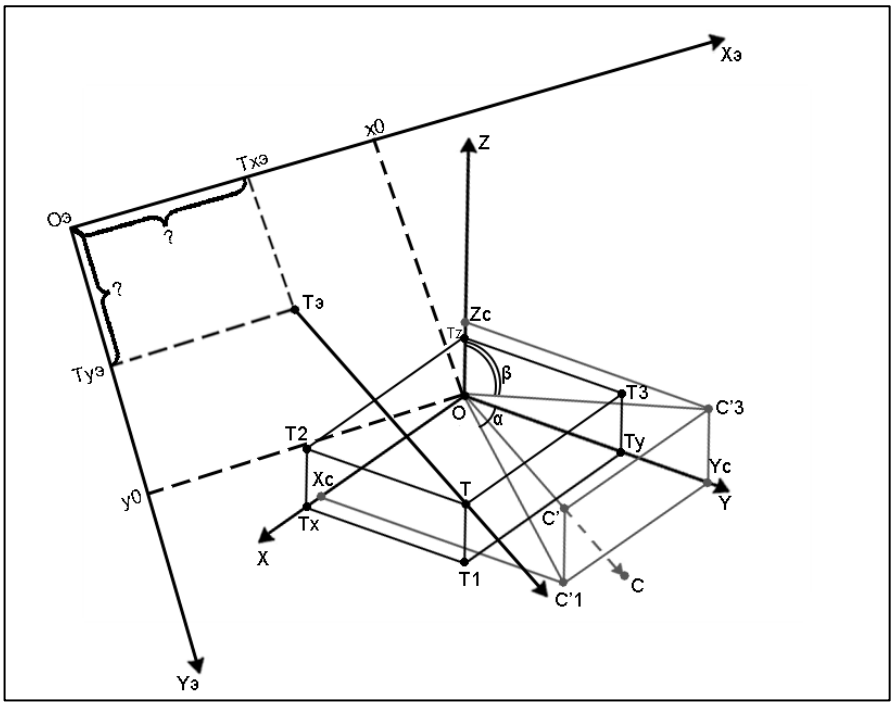


Рис. 2

Для произвольного расположения наблюдателя отсутствует базовое геометрическое преобразование, позволяющее найти проекцию точки T. Однако среди базовых геометрических преобразований присутствуют 3 преобразования (матрицы) для решения соответствующей задачи для наблюдателя, находящегося на одной из координатных осей (матрицы 5.a, 5.b, 5.c, если наблюдатель находится соответственно на оси Ox, Oy, Oz). Следовательно, возникает необходимость совмещения одной из координатных осей и вектора OC.

В качестве такой координатной оси возьмем, например, ось Oz. Соответственно, плоскость экрана будет задаваться осями Oxи Oy.

Данное преобразование возможно осуществить через выполнение 2-х поворотов

1. 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.
   Данному преобразованию соответствует матрица 1.c.

Матрица (1.c)’

$$[R_Z] = \begin{matrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

Рассмотрим иллюстрацию данного преобразования, рисунок 3:

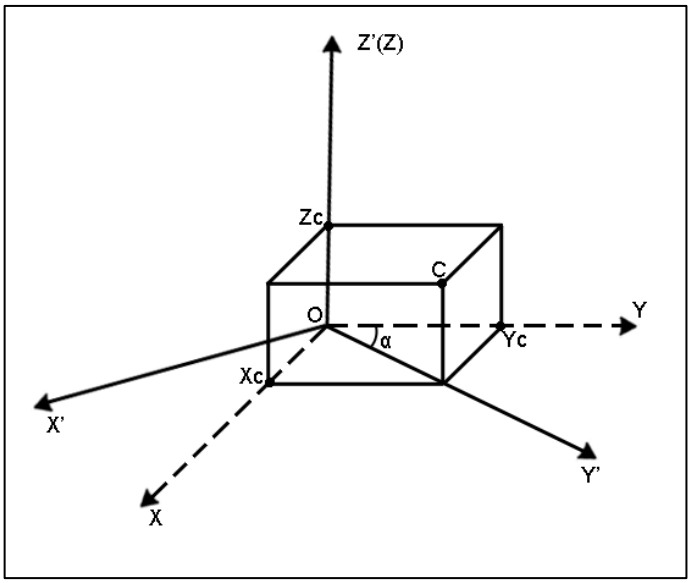


Рис.3

По данному рисунку можно без труда вычислить cos(α) и sin(α):

$$\cos(\alpha) = \frac{\mathrm{Yc}}{\sqrt{\mathrm{Xc}^2 + \mathrm{Yc}^2}} \quad (1)$$

$$\sin(\alpha) = \frac{\mathrm{Xc}}{\sqrt{\mathrm{Xc}^2 + \mathrm{Yc}^2}} \quad (2)$$

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

2. Поворот системы координат вокруг оси Ox' на угол β по часовой 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.
   Данному преобразованию соответствует матрица 1.a

Матрица (1.a)'

$$[R_X] = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & \sin\beta & 0 \\ 0 & -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

Рассмотрим иллюстрацию данного преобразования, рисунок 4:

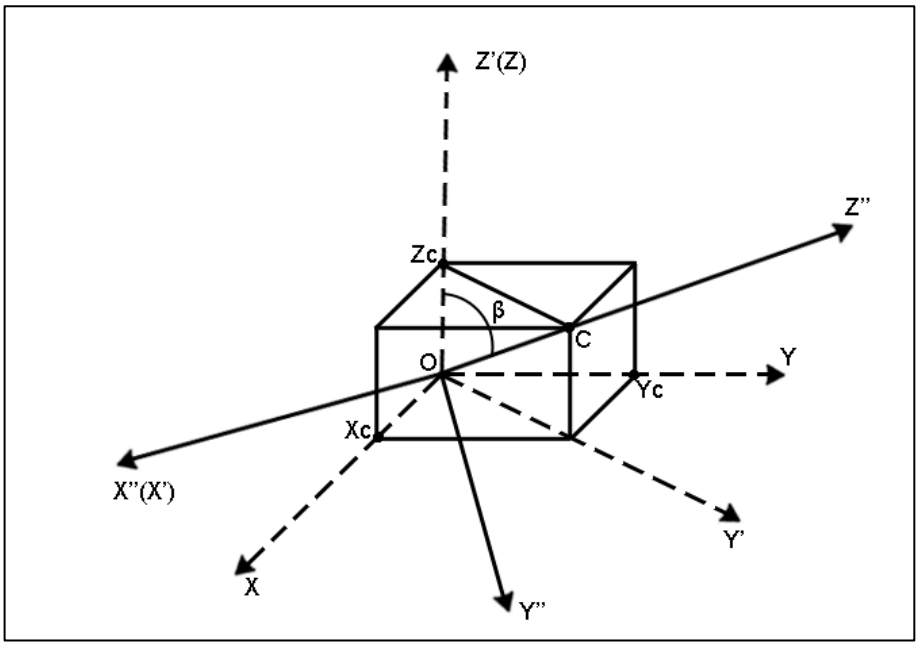


Рис.4

По данному рисунку можно без труда вычислить cos(β) и sin(β):

$$cos(\beta) = \frac{Zc}{\overline{Xc^2 + Yc^2 + Zc^2}} \quad (3)$$

$$sin(\beta) = \frac{\overline{Xc^2 + Yc^2}}{\overline{Xc^2 + Yc^2 + Zc^2}} \quad (4)$$

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 R_x на единичную (т.е. cos(α) = 1, sin(α) = 0)

3. <u>Отражение (зеркалирование) оси Охотносительно плоскости YOZ</u>
   Из рисунка 4 видно, что ось Ox'' направлена в противоположную сторону относительно оси $O_э X_э$. В связи с этим применим данное базовое преобразование.

Данному преобразованию соответствует матрица 3.а.

Матрица (3.а)'

$$[M_X] = \begin{matrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

Рассмотрим иллюстрацию данного преобразования, рисунок 5:

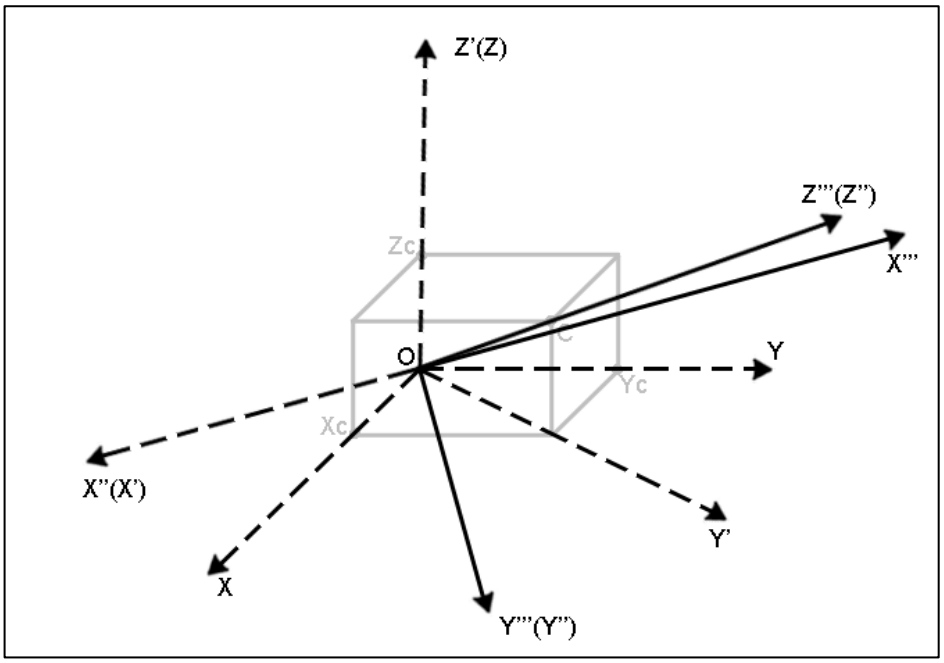


Рис.5

4. <u>Проецирование на плоскость XOY (при расположении наблюдателя, смотрящего в направлении начала координат, в бесконечности на оси Oz (в точке (0,0,1,0))).</u>

   Данному геометрическому преобразованию соответствует матрица 5.c

   Матрица (5.c)'

$$[P_Z] = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

5. <u>Перенос системы координат на вектор (λ, μ, ν)).</u>
   В нашем случае:
   λ = -x0; μ = -y0; ν = 0,
   где x0, y0 – экранные координатыточки O.

   Среди базовых геометрических преобразований присутствует <u>перенос точки</u> на вектор*(λ, μ, ν).*Так как в нашем случае необходимо осуществить <u>перенос системы координат</u>, то координаты вектора переноса поменяют знаки *(-λ, -μ, -ν).* Подставим исходные значения для каждой координаты: *(-λ, -μ, -ν)* = (-(-x0), -(-y0), 0) = (x0, y0, 0)**(5)**

   Итоговый вектор переноса системы координат: (x0, y0, 0)
   Данному геометрическому преобразованию соответствует матрица 4

Матрица (4)’

$$[T] = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \text{x0} & \text{y0} & 0 & 1 \end{matrix}$$

Рассмотрим иллюстрацию данного преобразования, рисунки 6, 7:

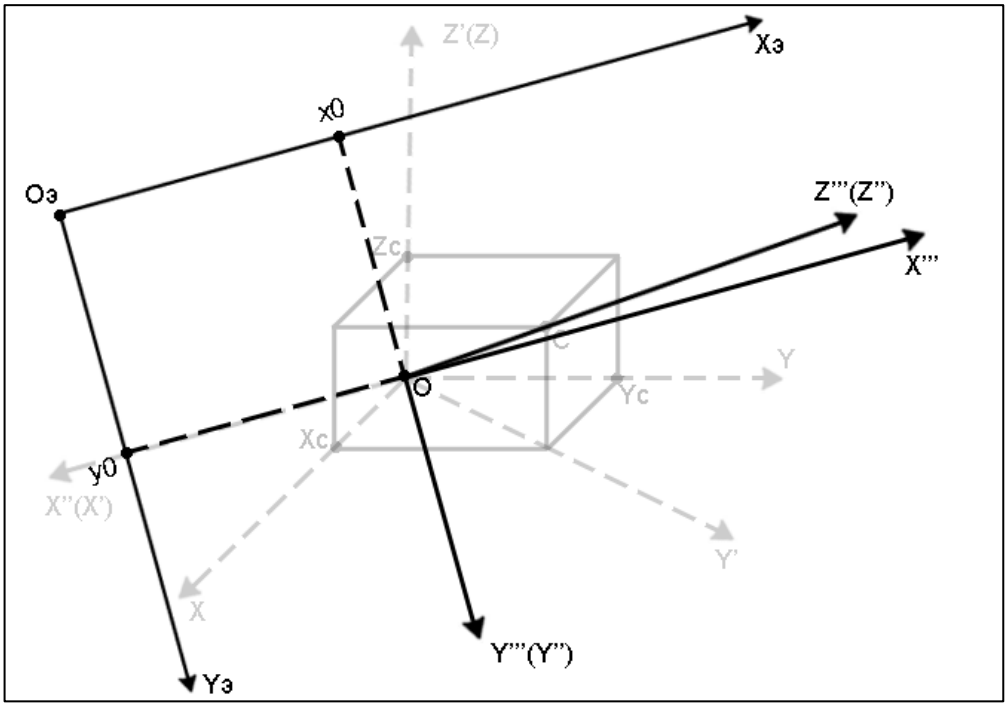


Рис. 6

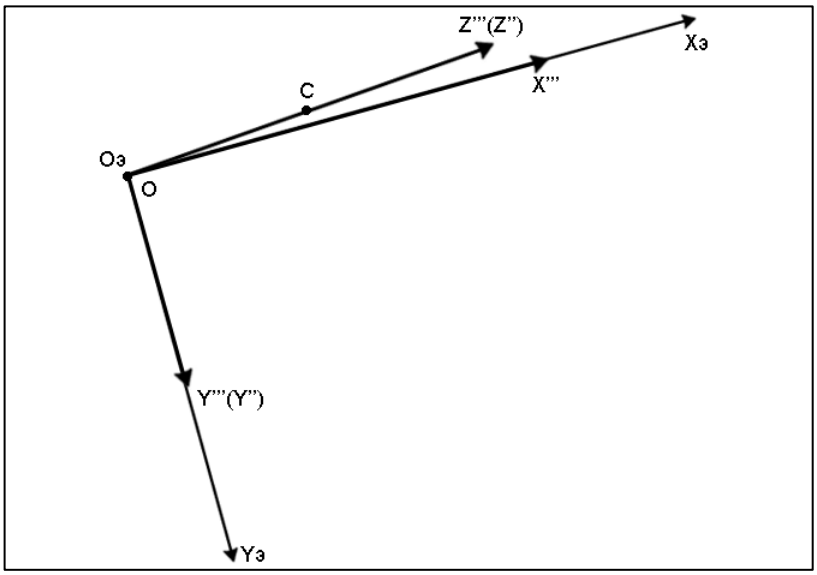


Рис. 7

Представим итоговое сложное геометрическое преобразование как совокупность 5последовательных базовых преобразований, рассмотренных выше конкретно для нашей задачи. Результатом будет следующая формула:

$$A = R_Z \cdot R_X \cdot M_X \cdot P_Z \cdot T \quad \textbf{(6)}$$

В конечном итоге, при помощи подробно рассмотренной матрицы A , сможем найти ортогональную проекцию любой точки, тем самым решив поставленную задачу. Для осуществления этого необходимо умножить (справа) вектор-строку точки на матрицу A , результатом чего будет вектор-строка искомой точки:

$$T' = T\ [A] \quad \textbf{(7)}$$

Также рассмотрим случай, при котором ортогональное проецирование невозможно:
Если точка Симеет однородные координаты: (0, 0, 0, 0), то не определен вектор направления проецирования (ОС). В данном случае, изображения аксонометрического чертежа не появится.

Обобщенный алгоритм:

1. Проверим, не совпадает ли точка С с началом координат (точкой О) (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. В противном случае переходим дальше по алгоритму.

2. Подготовка матрицы поворота системы координат вокруг оси Oz на угол α по часовой стрелке (матрица (1.c)') (подсчет cos(α) и sin(α) по формулам **(1)**, **(2)**). 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 (1.a)')(подсчет cos(β) и sin(β) по формулам **(3)**, **(4)**).Также проверим, не равны ли координаты Xc, Yc, Z снулю, для недопущения ошибки при делении на ноль.

4. Задание матрицы отражения оси Ox относительно плоскости YOZ(матрица (3.a)').

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 (5.c)').

6. Осуществление сдвига осей координат на вектор (x0, y0, 0) , полученный по формуле **(5)** (матрица (4)').

7. Формирование матрицы итогового сложного геометрического преобразования: ортогонального проецирования для наблюдателя в точке C, формула **(6).**

8. Осуществление данного сложного преобразования к 11-ти точкам: T^o, $T1^o$, $T2^o$, $T3^o$, Tx^o, Ty^o, Tz^o, Tox^o, Toy^o, Toz^o, To^o, а именно: умножение вектор-строку точек на матрицу итогового сложного преобразования, формула **(7)**и получение новых точек после преобразования: T'^o, $T'1^o$, $T'2^o$, $T'3^o$, $T'x^o$, $T'y^o$, $T'z^o$, $T'ox^o$, $T'oy^o$, $T'oz^o$, $T'o^o$ соответственно.

   *Замечание:*Как можно заметить, в данном случае вместо 14-ти точек остается 11. Это связано с тем, что в данной лабораторной работе чертеж всегда находится в первом октанте, поэтому для точек $–Tox^o$, $-Toy^o$, $-Toz^o$ не производим никаких действий.

2.1.2. **При центральном проецировании**

Задание:
Необходимо построить проекцию точки Тна 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 (x_c, y_c, z_c)

Найти:
Координаты ортогональной проекции Тэ точки Тна плоскость экрана (Тхэ, Туэ).

Решение:
Для того, чтобы построитьперспективное изображение, используется центральное проецирование, при котором центр проекции (точка C) находится на конечном расстоянии от экрана, в отличие от того, что было при ортогональном проецировании.
Рассмотрим рисунок 8:

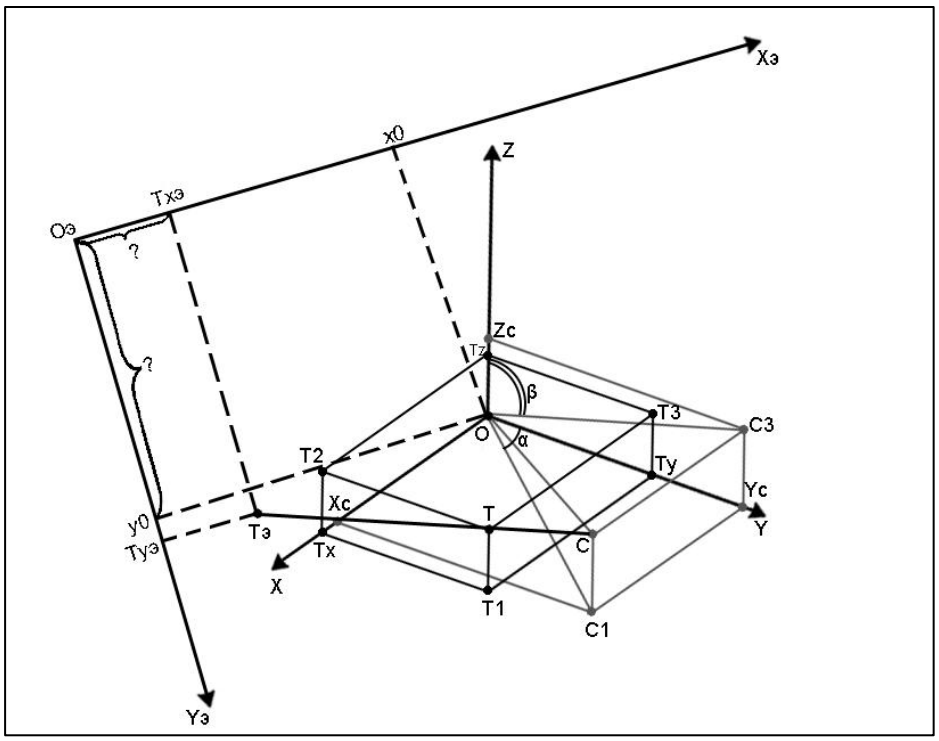


Рис. 8

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 c (длина радиус-вектора центра проецирования):

$$c = \sqrt{(Xc)^2 + (Yc)^2 + (Zc)^2}; \quad \textbf{(8)}$$

То есть, сложное геометрическое преобразование будет выглядеть следующим образом:

$$Ac = R_Z \cdot R_X \cdot M_X \cdot C_Z \cdot P_Z \cdot T \quad (\mathbf{9})$$

Базовые преобразования [R_z], [R_x], [M_x], [P_z], [T]рассматриваются так же, как и в пункте 2.1.1

Рассмотрим базовое преобразование [C_z]. Данному геометрическому преобразованию соответствует матрица 6.c:

Матрица (6.c)'

$$[Cz] = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1/c \\ 0 & 0 & 0 & 1 \end{matrix}$$

При центральном проецировании, в отличие от ортогонального, в однородных координатах (x, y, z, $1-\frac{z}{c}$) масштабный множитель wне будет равен 1. Необходимо учитывать, что соответствующие им “обычные” координаты будут равны:

$$\left(\frac{x}{1-\frac{z}{c}}, \frac{y}{1-\frac{z}{c}}, \frac{z}{1-\frac{z}{c}}, 1\right) \text{(10)}$$

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

$$T' = T\ [Ac] \textbf{(11)}$$

Рассмотрим случаи, при которых центральное проецирование невозможно.
Рассмотрим рисунок 9:

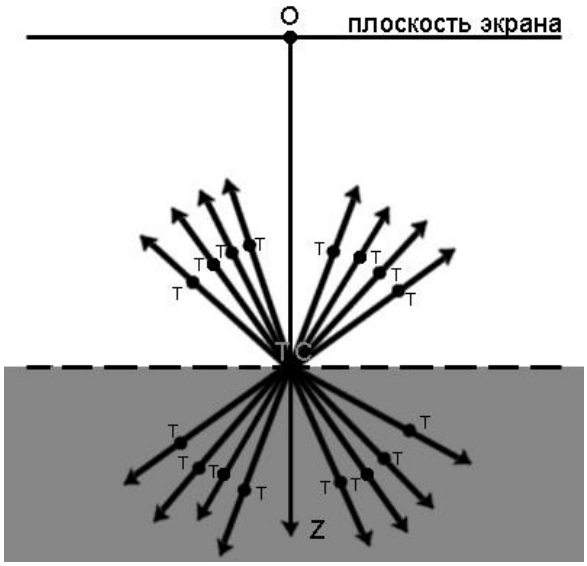


Рис. 9

1) ***Проверка № 1***: Точка C совпадает с началом координат (точкой O).В данном случае не определен вектор направления проецирования OC.
2) ***Проверка № 2:*** Точка Tсовпадает с точкой C. В этом случае не определен вектор CT.
3) ***Проверка № 3:***В случаях, которые закрашены на рисунке 9 серым цветом вектор CTне будет пересекать плоскость экрана.
   Реализуем проверку всех этих случаев:
   Для проверки, к какой области на рисунке 9 относится вектор CT можно использовать косинус угла между векторами CO и CT,

обозначим этот угол α(рисунок 10). Если косинус окажется неположительным, то построить центральную проекцию невозможно

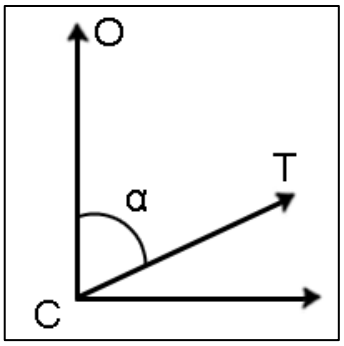


Рис. 10

Косинус можно найти при помощи скалярного произведения векторов:

$$\cos\ \alpha\ = \frac{(CO, CT)}{CO\ * |CT|}$$

CO (0-Xc, 0-Yc, 0-Zc) = (-Xc; -Yc; -Zc); CT (Xt – Xc, Yt – Yc, Zt – Zc);

$$CO, CT\ = -Xc * \ Xt - Xc\ \ - Yc * \ Yt - Yc\ \ - Zc * (Zt - Zc);$$

$$CO\ = \ \ \sqrt{Xc\ ^2 + \ \ Yc\ ^2 + \ \ Zc\ ^2}$$

$$CT\ = \ \ \sqrt{(Xt - Xc)^2 + \ (Yt - Yc)^2 + \ (Zt - Zc)^2}$$

Подставим и получим итоговую формулу:

$$\cos\ \alpha\ = \frac{-Xc * \ Xt - Xc\ \ - Yc * \ Yt - Yc\ \ - Zc * (Zt - Zc)}{\sqrt{Xc\ ^2 + \ \ Yc\ ^2 + \ \ Zc\ ^2} * \ \ \sqrt{(Xt - Xc)^2 + \ (Yt - Yc)^2 + \ (Zt - Zc)^2}}$$

По формуле видно, что выражение в знаменателе не может быть отрицательным, следовательно, знак косинуса угла между векторами COи CTопределяет выражение в числителе, а именно:
Если:

$$-Xc * \ Xt - Xc\ \ - Yc * \ Yt - Yc\ \ - Zc * \ Zt - Zc\ \ \leq 0$$ **(12)**

то построить центральную проекцию невозможно.

4) ***Проверка № 4*:** Выход за границы экрана.
При исходном расположении точек Ти Среализация проверки данного случая будет являться достаточно нетривиальной задачей, так как нам необходимо будет искать пересечение вектора проецирования с плоскостью экрана. Гораздо проще реализовать данную проверку в том случае, если нами уже были получены экранные координаты точек. Здесь все сведется к тому, что необходимо будет осуществить простой габаритный тест.
Рассмотрим рисунок 11:

Рис. 11

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

Обобщенный алгоритм:

1) Проверим, не совпадает ли точка C с началом координат (точкой O) (т.е. не имеет ли точка C однородные координаты (0, 0, 0, 0).) (проверка № 1).Если это так, то нет вектора CO и задача неразрешима. В противном случае переходим дальше по алгоритму.
2) Проверим, не совпадает ли точка T с точкой C (проверка № 2). Если это так, то не будет определен вектор CT и задача неразрешима. В противном случае переходим дальше по алгоритму.
3) Проверим, пересекает ли вектор CT плоскость экрана (проверка № 3). Если это так, то задача неразрешима. В противном случае переходим дальше по алгоритму.
4) Подготовка матрицы [R_z] (проверка деления на 0) (аналогично шагу 2) обобщенного алгоритма пункта 2.1.1)
5) Подготовка матрицы [R_x] (проверка деления на 0) (аналогично шагу 3) обобщенного алгоритма пункта 2.1.1)
6) Задание матрицы [M_x] (аналогично шагу 4) обобщенного алгоритма пункта 2.1.1)
7) Подготовка матрицы [C_z] перспективного преобразования (При расположении наблюдателя на оси Oz в точке (0, 0, c, 1) (Матрица (6.c)’)
8) Подготовка матрицы [P_z] (аналогично шагу 5) обобщенного алгоритма пункта 2.1.1)
9) Подготовка матрицы [T] (аналогично шагу 6) обобщенного алгоритма пункта 2.1.1)
10) Формирование матрицы итогового сложного геометрического преобразования: центрального проецирования для наблюдателя в точке C, формула **(8)**

11) Осуществление данного сложного преобразования к 11-ти точкам: T^o, $T1^o$, $T2^o$, $T3^o$, Tx^o, Ty^o, Tz^o, Tox^o, Toy^o, Toz^o, To^o, а именно: умножение вектор-строку точек на матрицу итогового сложного преобразования, формула **(10)**и получение новых точек после преобразования:
T'^o, $T'1^o$, $T'2^o$, $T'3^o$, $T'x^o$, $T'y^o$, $T'z^o$, $T'ox^o$, $T'oy^o$, $T'oz^o$, $T'o^o$соответственно.
12) Приведение однородных координат точки (с масштабным множителем wне равным 1) к “обычным” координатам, формула (9)
13) Проверка выхода точек за границы экрана (Проверка № 4). Если ни одна из точек не “вышла” за границы экрана, то переходим к следующему пункту. В противном случае выводим соответствующую ошибку.

2.2. **Вычисление координат проекций точекТ иСдля комплексного чертежа каждой из них.**
Данная подзадача является аналогичной той, что рассматривалась ранее в 1-й лабораторной работе.

3. Вывод

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

Описание применяемых методов (3-й уровень детализации)

Для нахождения ортогональных и центральных проекций точки используется тип данных *Матрица,*описывающий матрицы базовых и сложных геометрических преобразований (матрица 4*4).Для описания 4D-точек используется тип *Точка4D*. Также для описания 3D и 2Dточек используются типы *Точка3D* и *Точка2D* соответственно.
Для хранения однородных координат каждой точки будем использовать массив “*хранилищеОднородныхТочек”*[].
После преобразования (умножения справа на матрицу сложного преобразования, полученной по формуле **(6)**) 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).
xC, yC, zC - установленные значения соответствующих ползунковых переключателей для координат точки C (x, y, zсоответственно).
w, h–ширина и высота экрана
хранилище3DТочекТ[T, T1, T2, T3, Tx, Ty, Tz, Tox, -Tox, Toy, -Toy, Toz, -Toz, To]–массив начальных 3D-координат (для точки T).
хранилище3DТочекС[C, C1, C2, C3, Cx, Cy, Cz, Cox, -Cox, Coy, -Coy, Coz, -Coz, Co]–массив начальных 3D-точек(для точки C).
хранилище4DТочекТ[T^o, $T1^o$, $T2^o$, $T3^o$, Tx^o, Ty^o, Tz^o, Tox^o, Toy^o, Toz^o, To^o] – массивначальныходнородных координат (дляточкиT).
хранилище4DТочекС[C^o, $C1^o$, $C2^o$, $C3^o$, Cx^o, Cy^o, Cz^o, Cox^o, Coy^o, Coz^o, Co^o] – массивначальныходнородных координат (дляточкиC).

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.
isOrthogonal – переменная, хранящая 1 в случае, если выбрано ортогональное проецирование и 0 в случае, если выбрано центральное проецирование.
cosRz, sinRz–переменные, хранящие значения cosи sinдля матрицы [Rz]
cosRx, sinRx–переменные, хранящие значения cosи sinдля матрицы [Rx]
c–переменная, хранящая значение сдля матрицы [Cz]
lenAxisX, lenAxisY–переменные, длины полуосей Охи Оусоотвественно

Переменные для комплексного чертежа:
Данный пункт тому, что рассматривался ранее, в 1-й лабораторной работе.

Укрупненный алгоритм

1. **Ввод**
    1) Вводим координаты точек T и C. Данный пункт аналогичен тому, что рассматривался ранее в 1-й лабораторной работе.
    2) Выбираем тип проецирования: ортогональное или центральное (isOrthogonal = 1 или = 0)

2. **Обработка**

2.1. **Для аксонометрического чертежа**

2.1.1. **При ортогональном проецировании**

1) Проверим, совпадение точки C с началом координат (точкой O) (т.е. выполняется ли условие xC = 0 и yC=0 и zC=0)
   Если это так, то задача неразрешима. В противном случае переходим дальше по алгоритму.

2) Подготовка матрицы поворота системы координат вокруг оси Oz на угол α по часовой стрелке (cos(α) = cosRz, sin(α) = sinRz).
   Матрица [R_z] = {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
   cosRz = yC/ $\sqrt{(xC)^2 + (yC)^2}$; sinRz = xC/ $\sqrt{(xC)^2 + (yC)^2}$во всех остальных случаях(формулы **(1),(2)**)

3) Подготовка матрицы поворота системы координат вокруг оси Ox на угол β по часовой стрелке(cos(β) = cosRx, sin(β) = sinRx)
   Матрица [R_x] = {1 0 0 0
   0 cosRx sinRx 0
   0 –sinRx cosRx 0
   0 0 0 1}

   Где:
   cosRx= zC/ $\sqrt{(xC)^2 + (yC)^2 + (zC)^2}$;
   sinRx = $\sqrt{(xC)^2 + (yC)^2}$/ $\sqrt{(xC)^2 + (yC)^2 + (zC)^2}$; во всех случаях (кроме того, когда .xC, yCи zC одновременно равны 0) (формулы **(3),(4)**)

4) Задание матрицы отражения оси Oxотносительно плоскости YOZ
   Матрица [M_x] = {-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 [P_z] = {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 **(5)**))

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 С, формула **(6).**
Матрица [A] = [R_z]* [R_x]* [M_x]* [P_z]* [T];

8) Осуществление данного сложного преобразования в цикле, к каждой из точек из массива *хранилищеОднородныхТочек*[T^o, $T1^o$, $T2^o$, $T3^o$, Tx^o, Ty^o, Tz^o, Tox^o, Toy^o, Toz^o, To^o], а именно: умножение вектор-строку точек на матрицу итогового сложного преобразования, формула**(7),** записывая результаты в массив *хранилищеПреобразованныхОднородныхТочек*[T'^o, $T'1^o$, $T'2^o$, $T'3^o$, $T'x^o$, $T'y^o$, $T'z^o$, $T'ox^o$, $T'oy^o$, $T'oz^o$, $T'o^o$]

2.1.2. **При центральном проецировании**

1) Проверим, не совпадает ли точка С с началом координат (точкой О) (т.е. не имеет ли точка С однородные координаты (0, 0, 0, 0).) (проверка № 1).Если это так, то задача неразрешима. В противном случае переходим дальше по алгоритму.
2) Проверим, не совпадает ли точка Т с точкой С (проверка № 2). Если это так, то не будет определен вектор СТ и задача неразрешима. В противном случае переходим дальше по алгоритму.
3) Проверим, пересекает ли вектор СТ плоскость экрана (проверка № 3). Если это так, то проекция отсутствует. В противном случае переходим дальше по алгоритму.
4) Подготовка матрицы [R_z] (аналогично шагу 2 укрупненного алгоритма пункта 2.1.1)
5) Подготовка матрицы [R_x] (аналогично шагу 3) укрупненного алгоритма пункта 2.1.1)
6) Задание матрицы [M_x] (аналогично шагу 4) укрупненного алгоритма пункта 2.1.1)
7) Подготовка матрицы [C_z] перспективного преобразования (При расположении наблюдателя на оси Oz в точке (0, 0, c, 1) (матрица (6.c)')
Матрица [C_z] = {1 0 0 0
0 1 0 0
0 0 1-1/c
0 0 0 1}
c = $\overline{(xC)^2 + (yC)^2 + (zC)^2}$ (формула **(8)**)
8) Подготовка матрицы [P_z] (аналогично шагу 5) укрупненного алгоритма пункта 2.1.1)

9) Подготовка матрицы [T] (аналогично шагу 6) укрупненного алгоритма пункта 2.1.1)
10) Формирование матрицы итогового сложного геометрического преобразования: центрального проецирования для наблюдателя в точке C, формула **(9)** *Матрица* [Ac] = $[R_z]^* [R_x]^* [M_x]^* [C_z]^* [P_z]^* [T]$;
11) Осуществление данного сложного преобразования в цикле, к каждой из точек из массива *хранилищеОднородныхТочек*[T^o, $T1^o$, $T2^o$, $T3^o$, Tx^o, Ty^o, Tz^o, Tox^o, Toy^o, Toz^o, To^o], а именно: умножение вектор-строку точек на матрицу итогового сложного преобразования, формула**(11),** записывая результаты в массив *хранилищеПреобразованныхОднородныхТочек*[T'^o, $T'1^o$, $T'2^o$, $T'3^o$, $T'x^o$, $T'y^o$, $T'z^o$, $T'ox^o$, $T'oy^o$, $T'oz^o$, $T'o^o$]
12) Приведение однородных координат (с масштабным множителем wне равным 1) каждой точки из массива *хранилищеПреобразованныхОднородныхТочек*[T'^o, $T'1^o$, $T'2^o$, $T'3^o$, $T'x^o$, $T'y^o$, $T'z^o$, $T'ox^o$, $T'oy^o$, $T'oz^o$, $T'o^o$] к “обычным” координатам в цикле, формула**(10),** записывая результаты в массив *хранилище2DТочек*[T''^o, $T''1^o$, $T''2^o$, $T''3^o$, $T''x^o$, $T''y^o$, $T''z^o$, $T''ox^o$, $T''oy^o$, $T''oz^o$, $T''o^o$]
13) Проверка выхода точек за границы экрана (Проверка № 4). Если ни одна из точек не “вышла” за границы экрана, то переходим к следующему пункту. В противном случае выводим соответствующую ошибку.

2.2. **Для комплексного чертежа**
Данная подзадача является аналогичной той, что рассматривалась ранее в 1-й лабораторной работе.

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

Детализированный алгоритм

1. **Ввод**
   Если (выбраноОртогональноеПроецирование) isOrhogonal = 1;
   Иначе isOrhogonal = 0;

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

2. **Обработка**
   2.1. **Для аксонометрического чертежа**
      2.1.1. **При ортогональном проецировании**

```
Функция ПроверкаЦентраПроецирования(){
    Если (Xc== 0 и Yc== 0 и Zc== 0){
    ВывестиТекст (“Ошибка!Не определен вектор направления
    проецирования ОС”);
        Вернуть False;
    }
    Иначе
        Вернуть True;

    }
```

Замечание: двигаемся дальше по алгоритму только в том случае, если функция *ПроверкаЦентраПроецирования*() вернула значение *True*. В противном случае работа алгоритма завершается.

```
Функция ПодготовкаМатрицыПоворотаRz(cosRz,sinRz){
    МатрицаRz = {cosRzsinRz 0 0
                 -sinRzcosRz0 0
                  0      0   1 0
                  0      0   0 1};
    Вернуть (Rz);
    }

Функция ПодготовкаМатрицыПоворотаRx(cosRxи sinRx){
    МатрицаRx = {1   0       0      0
                 0 cosRx sinRx   0
                 0 –sinRx cosRx  0
                 0    0   0 1};
    Вернуть (Rx);
    }
```

```
Функция ПодготовкаМатрицыОтраженияOx(){
```

МатрицаM_x = {-1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1};
Вернуть (Mx);
}
Функция *ПодготовкаМатрицыПроецированияXOY*(){
МатрицаP_z = {1 0 0 0
0 1 0 0
0 0 0 0
0 0 0 1};
Вернуть (Pz);
}

Функция *ПодготовкаМатрицыСдвига*(-lenAxisX, -lenAxisY, 0){
*Матрица*T= {1 0 0 0
0 1 0 0
0 0 1 0
lenAxisX lenAxisY 0 1};
Вернуть (T);
}

Функция *ПодготовкаСложногоПреобразования*(Rz, Rx, Mx, Pz, T){
Матрица A = Rz * Rx * Mx * Pz * T;
Вернуть (A);
}

Замечание: В данном случае операция умножения “*” является перегруженной для матриц.

Если (*ПроверкаЦентраПроецирования*() == true){***//1)***
*Матрица*Rz = ПодготовкаМатрицыПоворотаRz*(*cosRz,sinRz*);//2)***
*Матрица*Rx = ПодготовкаМатрицыПоворотаRx*(*cosRxи sinRx*);//3)***
Матрица Mx = ПодготовкаМатрицыОтраженияOx();***//4)***
*Матрица*Pz = ПодготовкаМатрицыПроецированияXOY();***//5)***
*Матрица*T = ПодготовкаМатрицыСдвига(-lenAxisX, -lenAxisY, 0);***//6)***
*Матрица*A = ПодготовкаСложногоПреобразования(Rz, Rx, Mx, Pz,T);***//7)***

Цикл: для i = 1 до 11, с шагом 1:

хранилищеПреобразованныхОднородныхТочек[i] = *хранилищеОднородныхТочек*[i] * A;***//8)***

2.1.2. **При центральном проецировании**

Функция *ПроверкаЦентраПроецирования*() (аналогично шагу 1) детализированного алгоритма пункта 2.1.1)

Функция *ПроверкаСовпаденияСиТ*() {
    Если (isOrtoghonal == 0 и xC == xТи yC == yТи zC == zT){
        ВывестиТекст (“*Ошибка!Вектор СТ не определен!”);*
        Вернуть False
    Иначе
        Вернуть True
}

Функция *ПроверкаВектораСТ*() {
    Формула = -xC*(xT – xC) – yC*(yT – yC) – zC*(zT – zC);
    Если (isOrtoghonal == 0 и формула <= 0){
        ВывестиТекст (“*Ошибка!Вектор СТ не пересекает плоскость экрана”);*
        Вернуть False
    Иначе
        Вернуть True
}

Функция *ПодготовкаМатрицыПоворотаRz(*cosRz,sinRz) (аналогично шагу 2) детализированного алгоритма пункта 2.1.1)

Функция *ПодготовкаМатрицыПоворотаRx(*cosRxи sinRx)
(аналогично шагу 3) детализированного алгоритма пункта 2.1.1)

Функция *ПодготовкаМатрицыОтраженияOx*()
(аналогично шагу 4) детализированного алгоритма пункта 2.1.1)

Функция *ПодготовкаМатрицыПерспективногоПреобразованияOz*(c){
    МатрицаC_z = {1 0 0 0
                0 1 0 0
                0 0 1 -1/c
                0 0 0 1};
    Вернуть (Cz);
}

Функция *ПодготовкаМатрицыПроецированияXOY*()
(аналогично шагу 5) детализированного алгоритма пункта 2.1.1)

Функция *ПодготовкаМатрицыСдвига*(lenAxisX, lenAxisY, 0)
(аналогично шагу 6) детализированного алгоритма пункта 2.1.1)

Функция *ПодготовкаСложногоПреобразования*(Rz, Rx, Mx, Cz, Pz, T){
    *Матрица*A = Rz * Rx * Mx * Cz * Pz * T;
    Вернуть (A);
}
Замечание: В данном случае операция умножения “*” является перегруженной для матриц.

Функция *ПроверкаВыходаЗаГраницыЭкрана*(){

```
        Цикл: для i = 1 до 11, с шагом 1:
        Если (хранилище2DТочек[i] .x< 0 или хранилище2DТочек[i] .x>w      или
        хранилище2DТочек[i] .y< 0 или хранилище2DТочек[i] .y>h){

          ВывестиТекст (“Ошибка!Выход за границы экрана!”);
              Вернуть False
          }
          Иначе:
              Вернуть True

}

Если (ПроверкаЦентраПроецирования() == true и
ПроверкаСовпаденияСиТ()== trueи ПроверкаВектораСТ()== true){//1),2),3)

МатрицаRz = ПодготовкаМатрицыПоворотаRz(cosRz,sinRz); //4)
МатрицаRx = ПодготовкаМатрицыПоворотаRx(cosRxи sinRx);//5)
Матрица Mx = ПодготовкаМатрицыОтраженияOx();//6)
МатрицаCz = ПодготовкаМатрицыПерспективногоПреобразованияCz(c);//7)
МатрицаPz = ПодготовкаМатрицыПроецированияXOY();//8)
МатрицаT = ПодготовкаМатрицыСдвига(-lenAxisX, -lenAxisY, 0);//9)
МатрицаA = ПодготовкаСложногоПреобразования(Rz,RxMx,Cz,Pz,T);//10)

Цикл: для i = 1 до 11, с шагом 1: {

  хранилищеПреобразованныхОднородныхТочек[i] =
  хранилищеОднородныхТочек[i] * A;
  } //11)

  Цикл: для i = 1 до 11, с шагом 1:{
  хранилище2DТочек[i] .x=
  хранилищеПреобразованныхОднородныхТочек[i].x /
  хранилищеПреобразованныхОднородныхТочек[i].w;
  хранилище2DТочек[i] .y=
  хранилищеПреобразованныхОднородныхТочек[i].y /
  хранилищеПреобразованныхОднородныхТочек[i].w;
  }//12)

  ПроверкаВыходаЗаГраницыЭкрана(); //13)
  }
```

2.2. **Для комплексного чертежа**

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

3. **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#. Стиль программирования: объектно-ориентированный.

В данном пункте нам необходимо ответить на два основных вопроса:

1. **Из чего состоит программа?**
2. **Как работает программа?**

1. Из чего состоит программа?

Программа состоит из двух частей: **интерфейсная** и **прикладная**.

1.1. 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”.

Все компоненты, представленные выше и составляющие пользовательский интерфейс данной лабораторной работы, отображаются в окне (объекте класса “Form”). Это представлено на рисунке 12:

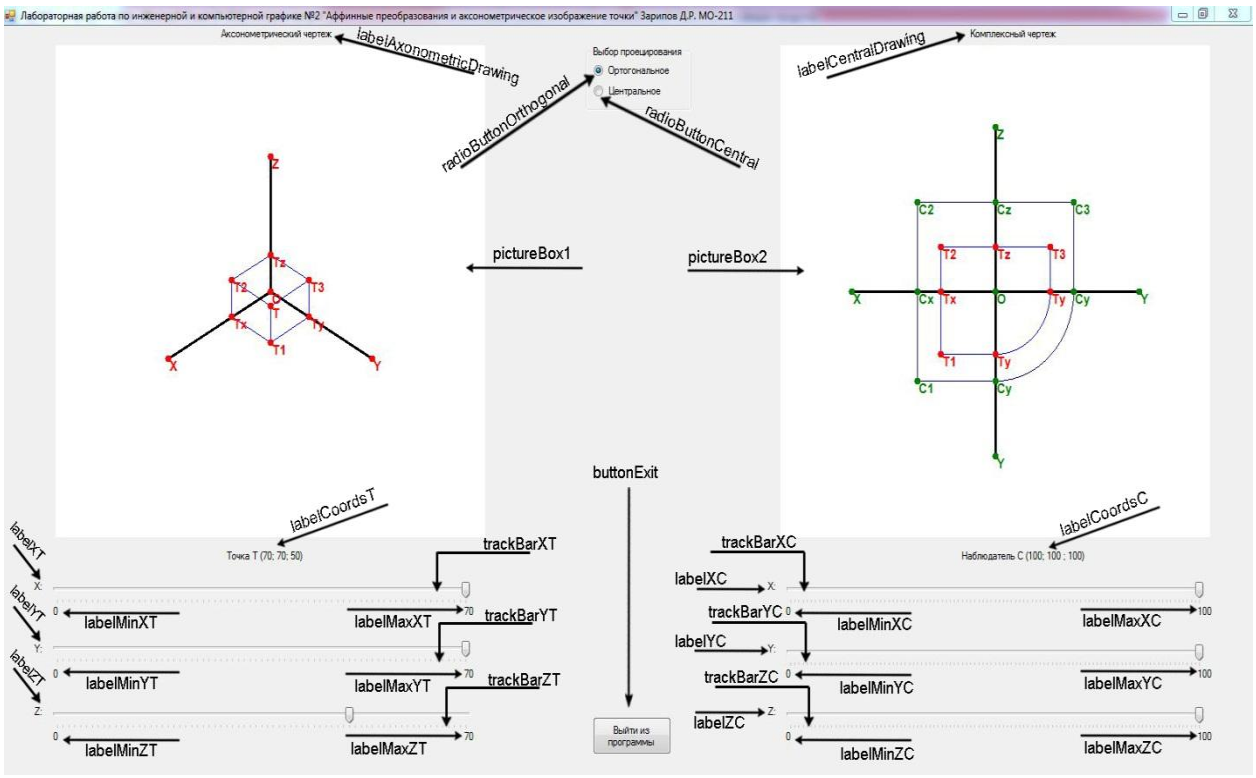


Рис.12 Пользовательский интерфейс

Представим информацию об основных объектах интерфейса:

- PictureBox
    1. PictureBox1 – вывод аксонометрического чертежа;
    2. PictureBox2 – вывод комплексного чертежа.

- TrackBar
    1. trackBarXT – ввод координаты x точки T;
    2. trackBarYT – ввод координаты y точки T;
    3. trackBarZT – ввод координаты z точки T;
    4. trackBarXC – ввод координаты x точки C;
    5. trackBarYC– ввод координаты y точки C;
    6. trackBarZC – ввод координаты z точки C;

- RadioButton
    1. radioButtonOrthogonal–выбор ортогонального проецирования;
    2. radioButtonCentral–выбор центрального проецирования;

    Замечание: две данные радикнопки связаны так, что выбрать можно только одну из них.

- Label
    1. labelAxonometricDrawing – отображение надписи “Аксонометрический чертеж” около соответствующего “пикчербокса”;

2. labelComplexDrawing – отображение надписи “Комплексный чертеж” около соответствующего “пикчербокса”;;
3. labelCoordsT – показание текущих введенных значений координатx, y, z точки T;
4. labelXT – отображение надписи “X” около соответствующего ползунка;
5. labelYT – отображение надписи “Y” около соответствующего ползунка;
6. labelZT – отображение надписи “Z” около соответствующего ползунка;
7. labelMinXT – отображение мин. значения координаты x точки T
8. labelMinYT – отображение мин. значения координаты y точки T
9. labelMinZT – отображение мин. значения координаты z точки T
10. labelMaxXT – отображение макс. значения координаты x точки T
11. labelMaxYT – отображение макс. значения координаты y точки T
12. labelMaxZT – отображение макс. значения координаты z точки T
13. labelCoordsC – показание текущих введенных значений координат x, y, z точки C;
14. labelXC – отображение надписи “X” около соответствующего ползунка;
15. labelYC– отображение надписи “Y” около соответствующего ползунка;
16. labelZC – отображение надписи “Z” около соответствующего ползунка;
17. labelMinXC – отображение мин. значения координаты x точки C
18. labelMinYC – отображение мин. значения координаты y точки C
19. labelMinZC – отображение мин. значения координаты z точки C
20. labelMaxXC – отображение макс. значения координаты x точки C
21. labelMaxYC – отображение макс. значения координаты y точки C
22. labelMaxZC – отображение макс. значения координаты z точки C

- Button
    1. buttonExit – выход из программы.

1.2. Прикладная часть

Программа состоит из одного внешнего класса:

- Form1

И 7-ми внутренних классов и 3-х интерфейсов:

- Axis
- CPoint3D
- CPoint2D
- CPoint4D
- AxonometricDrawing
- ComplexDrawing
- Matrix
- BaseMatrixTrasformations (интерфейс)
- ComplexMatrixTrasformations (интерфейс)
- FinalTrasformation (интерфейс)
- MatrixTrasformations

Рассмотрим отдельно каждый класс и интерфейс:

- **Form1**
  Внешний класс программы, содержащий следующие поля и методы:

  CPoint3D[] **storage3DPointT**[14]
  поле(массив), содержащее 14 вводимых пользователем 3D-точек (для точки T)
  CPoint3D[] **storage3DPointC**[14]
  поле(массив), содержащее 14 вводимых пользователем 3D-точек (для точки C)
  CPoint4D[] **storage4DPointT**[11]
  поле(массив), содержащее 11 точек с начальными однородными координатами (точки T, ее проекций и точек, задающих оси координат)
  CPoint4D[] **storage4DPointC**[11]
  поле(массив), содержащее 11 точек с начальными однородными координатами (точки C, ее проекций и точек, задающих оси координат)
  int **xT, yT, zT**
  поля, содержащие координаты точки Tx, y, zсоответственно.
  int **xC, yC, zC**
  поля, содержащие координаты точки Cx, y, zсоответственно.
  bool **isOrthogonal**
  поле, содержащее значение true, если выбрано ортогональное проецирование и falseв противном случае
  AxonometricDrawing **axonometricDrawing**
  поле, представляющее собой объект класса аксонометрического чертежа

ComplexDrawing **complexDrawing**
поле, представляющее собой объект класса комплексного чертежа

void **input()**
метод, обрабатывающий вводимые пользователем данные о координатах точек Ти Си о выборе типа проецирования;

void **processAxonometricOrth()**
метод, выполняющие все необходимые действия для пересчета однородных координат точек Ти С в случае, если выбрано ортогональное проецирование.

void **processAxonometricCentral()**
метод, выполняющие все необходимые действия для пересчета однородных координат точек Ти С в случае, если выбрано центральное проецирование.

void **processComplex()**
метод, пересчитывающий вводимые пользователем координаты 3D-точек Т и С из трехмерных в двумерные для комплексного чертежа;

void **outputAxonometric()**;
метод, выводящий пространственный чертеж.Данный метод выполнится только в том случае, если методы проверки векторов ОСи СТ, а также метод проверки за границы экранаодновременно вернутtrue;

void **outputComplex()**;
метод, выводящий комплексный чертеж;

void **createPoints()**;
метод, осуществляющий создание 3D-точек с соответствующими вводимыми пользователем 3D-координатами точек Т и С и запись их в массивы трехмерных точек storage3DPointT[] и storage3DPointC[] соответственно;

bool **errorOC()**;
метод, осуществляющий проверку вектора ОС. Возвращает значение false, если данный вектор не определен, в противном случае вернет значение true.

bool **errorCTEqual()**;
метод, осуществляющий проверку вектора СТ. Возвращает значение false, если данный вектор не определен, в противном случае вернет значение true.

bool **errorCT()**;
метод, осуществляющий проверку вектора СТ. Возвращает значение false, если данный вектор не пересекает плоскость экрана, в противном случае вернет значение true.

bool **overallTest()**;
метод, осуществляющий проверку выхода какой-либо точки за границы

экрана. Возвращает значение false, если хотя бы одна из точек “вылезла” за границы экрана, в противном случае вернет значение true.

Также в данном внешнем классе происходят обработки следующих событий:

void **pictureBox1_Paint()**;
обработкасобытия перерисовки аксонометрического чертежа;

void **pictureBox2_Paint()**;
обработкасобытия перерисовки комплексного чертежа;

void **trackBarXT_Scroll()**;
обработка события взаимодействия пользователя с ползунковым переключателем для изменения текущего значения координаты xточки T;

void **trackBarYT_Scroll()**;
обработка события взаимодействия пользователя с ползунковым переключателем для изменения текущего значения координаты yточки T;

void **trackBarZT_Scroll()**;
обработка события взаимодействия пользователя с ползунковым переключателем для изменения текущего значения координаты zточки T;

void **trackBarXC_Scroll()**;
обработка события взаимодействия пользователя с ползунковым переключателем для изменения текущего значения координаты xточки C;

void **trackBarYC_Scroll()**;
обработка события взаимодействия пользователя с ползунковым переключателем для изменения текущего значения координаты yточки C;

void **trackBarZC_Scroll()**;
обработка события взаимодействия пользователя с ползунковым переключателем для изменения текущего значения координаты zточки C;

void **radioButtonOrthogonal_CheckedChanged()**;
обработка события взаимодействия пользователя с радиокнопкой для выбора ортогонального проецирования;

void **radioButtonCentral_CheckedChanged()**;
обработка события взаимодействия пользователя с радиокнопкой для выбора центрального проецирования;

void **buttonExit_Click()**;
обработка события нажатия на кнопку выхода из программы;

- AxonometricDrawing

Внутренний класс, описывающий аксонометрический чертеж. Содержит следующие поля и методы:

CPoint4D[] **storage4DPointC**[11]

поле(массив), содержащее 11 точек с начальными однородными координатами (точка C, ее проекции и точки, задающие оси координат)

CPoint4D[] **storage4DPointT**[11]

поле(массив), содержащее 11 точек с начальными однородными координатами (точка T, ее проекции и точки, задающие оси координат) (до преобразования).

CPoint4D[] **storageAfterTransPointT**[11]

поле(массив), содержащее 11 точек с новыми однородными координатами (точка T, ее проекции и точки, задающие оси координат) (после преобразования).

CPoint2D[] **storage2DPointT**[11]

поле(массив), содержащее 11 точек с "обычными" 2D-координатами (после деления каждой координаты на масштабный множитель) (точка T, ее проекции и точки, задающие оси координат)

double **cosRz, sinRz, cosRx, sinRx, c**

поля, содержащие параметры, необходимые для подготовки соответствующие матриц.

double **xC, yC, zC**

поля, содержащие координаты точки C.

Matrix **Rz, Rx, Mx, Cz, Pz, T, Ao, Ac**

поля, содержащие матрицы необходимых базовых и сложных. геом. преобразований

bool **changeC**

поле, содержащее информацию о том, изменились ли координаты точки C

void **calculatePointsOrth**();

*метод, осуществляющий пересчет однородных координат для точки T и ее проекций на координатные оси и плоскости при ортогональном проецировании(из массива **storage4DPointT[]**, начальные однородные координаты каждой точки умножаются на матрицу сложного преоб-я для ортогон. проецирования и записываются в массив **storageAfterTransPointT[]**, а затем в массив **storage2DPointT[]***

void **calculatePointsCentral**();

*метод, осуществляющий пересчет однородных координат для точки T и ее проекций на координатные оси и плоскости при центральном проецировании. (из массива **storage4DPointT[]**, начальные однородные координаты каждой точки умножаются на матрицу сложного преоб-я для центр проецирования и записываются в массив **storageAfterTransPointT[]**, а затем делятся на масштабный множитель wи записываются в массив**storage2DPointT[]**)*

void **drawPoints**()

метод, осуществляющий отрисовку точки T и всех ее проекций на аксонометрическом макете.

Данный метод делится на два "подметода":

void_**drawLines**();

метод, осуществляющий отрисовку координатных осей и линий связи.

void_**drawPoints**();
метод, осуществляющий отрисовку точек и их наименований.

bool **checkCoords**()
метод, осуществляющий проверку координат точки C для недопущения деления на 0 при вычислении косинусов и синусов для соответствующих матриц поворота. Если координаты нулевые, то функция вернет false, в противном случае, функция вернет true.
void **calculateForMatrix**()
метод, осуществляющий пересчет всех необходимых параметров для матриц (косинусов, синусов и длины радиус-вектора центра проецирования)
void **calculateMatrix**()
метод, объединяющий подготовки всех необходимых базовых преоб-ний.
AxonometricDrawing() *конструктор класса без параметров.*

- Matrix
  Внутренний класс, описывающий тип данных для матриц. Содержит следующие поля и методы:

  List<List<double>>**matrix;**
  поле (список списков), представляющее собой двумерный список, хранящий матрицу.

  Matrix (double *m11,* double *m12,* double *m13,* double *m14,* double *m21,* double *m22,* double *m23,* double *m24,* double *m31,* double *m32,* double *m33,* double*m34,* double *m41,* double *m42,* double *m43,* double *m44)*
  *Конструкторклассас 16-юпараметрамидляинициализациматрицы 4*4*

  Matrix (double*m11,* double*m12,* double*m13,* double*m14)*
  Конструкторклассас 4-мяпараметрамидляинициализациматрицы-строки (используется для строки однородных координат точки).

  Matrixoperation* (Matrixa, Matrixb)
  Метод для перегрузки оператора умножения (*) для двух матриц.

- CPoint4D
  Внутренний класс, описывающий тип данных для 4D-точек. Содержит следующие поля и методы:
  double**x**;
  поле, содержащее значение координаты 4D-точки по оси Ox;

  double**y**;
  поле, содержащее значение координаты 4D-точки по оси Oy;

  double**z**;
  поле, содержащее значение координаты 4D-точки по оси Oz;

  double**w**;
  поле, содержащее значение масштабного множителя;

string**name**;
поле, содержащее наименование 4D-точки точки;

CPoint4D(int**x**, int**y**, int**z**, int**w**, string**name**);
конструктор класса с 5-ю параметрами: координаты трехмерной точки по каждой из трех осей координат; наименование точки; цвет точки.

- BaseMatrixTrasformations
  Интерфейс, содержащий сигнатуры методов подготовки каждой из 14-ти матриц базовых геометрических преобразований:

  Matrix **prepareMatrixRotationOx**(doublecos, doublesin)
  Поворот точки вокруг оси Ox на угол φ против часовой стрелки (если смотреть из точки, расположенной в бесконечности на оси Ox, в направлении начала координат).
  В качестве параметров передаются косинус и синус угла φ.

  Matrix **prepareMatrixRotationOy**(doublecos, doublesin)
  Поворот точки вокруг оси Oy на угол ψпротив часовой стрелки (если смотреть из точки, расположенной в бесконечности на оси Oy, в направлении начала координат)
  В качестве параметров передаются косинус и синус угла ψ.

  Matrix **prepareMatrixRotationOz**(doublecos, doublesin)
  Поворот точки вокруг оси Oz на угол χпротив часовой стрелки (если смотреть из точки, расположенной в бесконечности на оси Oz, в направлении начала координат)
  В качестве параметров передаются косинус и синус угла χ.

  Matrix **prepareMatrixExtension**(doublea, doubleb, doublec)
  Растяжение (сжатие) точки.
  В качестве параметров передаются коэффициенты масштабирования.

  Matrix **prepareMatrixMirrorYOZ**()
  Зеркалирование оси Oxотносительно плоскости YOZ

  Matrix **prepareMatrixMirrorXOZ**()
  Зеркалирование оси Oyотносительно плоскости XOZ

  Matrix **prepareMatrixMirrorXOY**()
  Зеркалирование оси Ozотносительно плоскости XOY

Matrix **prepareMatrixShift**(doublea, doubleb, doublec)
Переноснавектор (a, b, c). В качестве параметров передаются координаты вектора переноса.

Matrix **prepareMatrixProjectYOZ**()
Проецирование на плоскость YOZ (при расположении наблюдателя, смотрящего в направлении начала координат, в бесконечности на оси Ox (в точке (1,0,0,0))).

Matrix **prepareMatrixProjectXOZ**()
Проецирование на плоскость XOZ (при расположении наблюдателя, смотрящего в направлении начала координат, в бесконечности на оси Oy (в точке (0,1,0,0)))

Matrix **prepareMatrixProjectXOY**()
Проецирование на плоскость XOY (при расположении наблюдателя, смотрящего в направлении начала координат, в бесконечности на оси Oz (в точке (0,0,1,0)))

Matrix **prepareMatrixPerspectiveOx**(doublec)
Перспективное преобразование при расположении наблюдателя на оси Ox, в точке (c, 0, 0, 1). В качестве параметра передается длина радиус-вектора центра проецирования.

Matrix **prepareMatrixPerspectiveOy**(doublec)
Перспективное преобразование при расположении наблюдателя на оси Oy, в точке (0, c, 0, 1). В качестве параметра передается длина радиус-вектора центра проецирования.

Matrix **prepareMatrixPerspectiveOz**(doublec)
Перспективное преобразование при расположении наблюдателя на оси Oz, в точке (0, 0, c, 1). В качестве параметра передается длина радиус-вектора центра проецирования.

- ComplexMatrixTrasformations
  Интерфейс, содержащий сигнатуры методов подготовки 2-х матриц сложных геометрических преобразований для ортогонального и центрального проецирования:

  Matrix **prepareMatrixComplexTransformationOrthogonal**(MatrixRz, MatrixRx, MatrixMx, MatrixPz, MatrixT)
  Сложное геометрического преобразование для ортогонального проецирования. В качестве параметров передаются матрицы необходимых базовых геометрических преобразований именно в том порядке, в котором необходимо их перемножить для получения данного сложного геометрического преобразования

Matrix **prepareMatrixComplexTransformationCentral**(MatrixRz, MatrixRx, MatrixMx, MatrixCz, MatrixPz, MatrixT)
Сложное геометрического преобразование для центрального проецирования. В качестве параметров передаются матрицы необходимых базовых геометрических преобразований именно в том порядке, в котором необходимо их перемножить для получения данного сложного геометрического преобразования

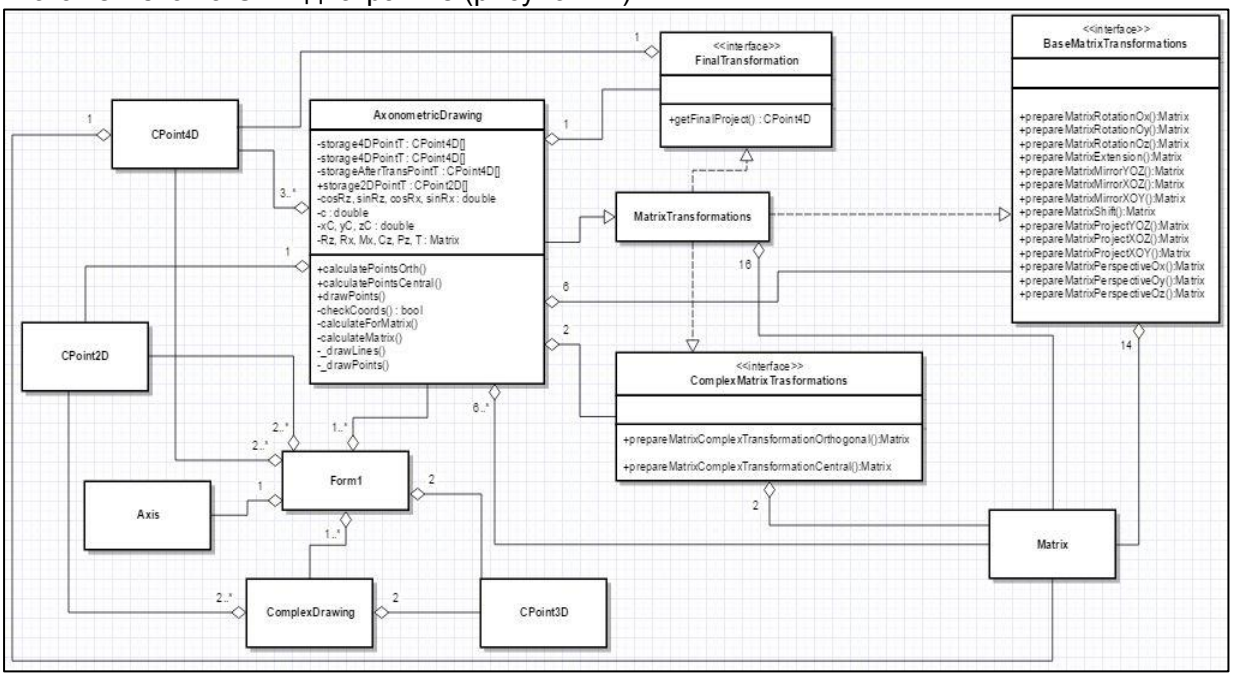
- FinalTrasformation
  Интерфейс, содержащий сигнатуру метода получения итоговых однородных координат точки:

  Matrix **getFinalProject**(CPoint4DT0, MatrixA)
  Получение строки из итоговых однородных координат точки.
  В качестве параметров передаются 4D-точка с начальными однородными координатами и матрица сложного геометрического преобразования.

- MatrixTrasformations
  Класс, реализующий все методы, сигнатуры который содержатся в 3-х вышеописанных интерфейсах.

- Классы Axis, CPoint3D, CPoint2D, ComplexDrawing
  уже были описаны в лабораторной работе № 1. В данной лабораторной работе они не меняются.

Класс AxonometricDrawingнаследуется от класса MatrixTransformations, который в свою очередь реализует все методы каждого из трех интерфейсов: BaseMatrixTrasformations, ComplexMatrixTrasformations, FinalTransformation. Покажем это на UML-диаграмме (рисунок 14):



Покажемна функциональной схеме (рис. 15), как взаимодействуют между собой функции и методы, а также как происходит их взаимодействие с элементами интерфейса.

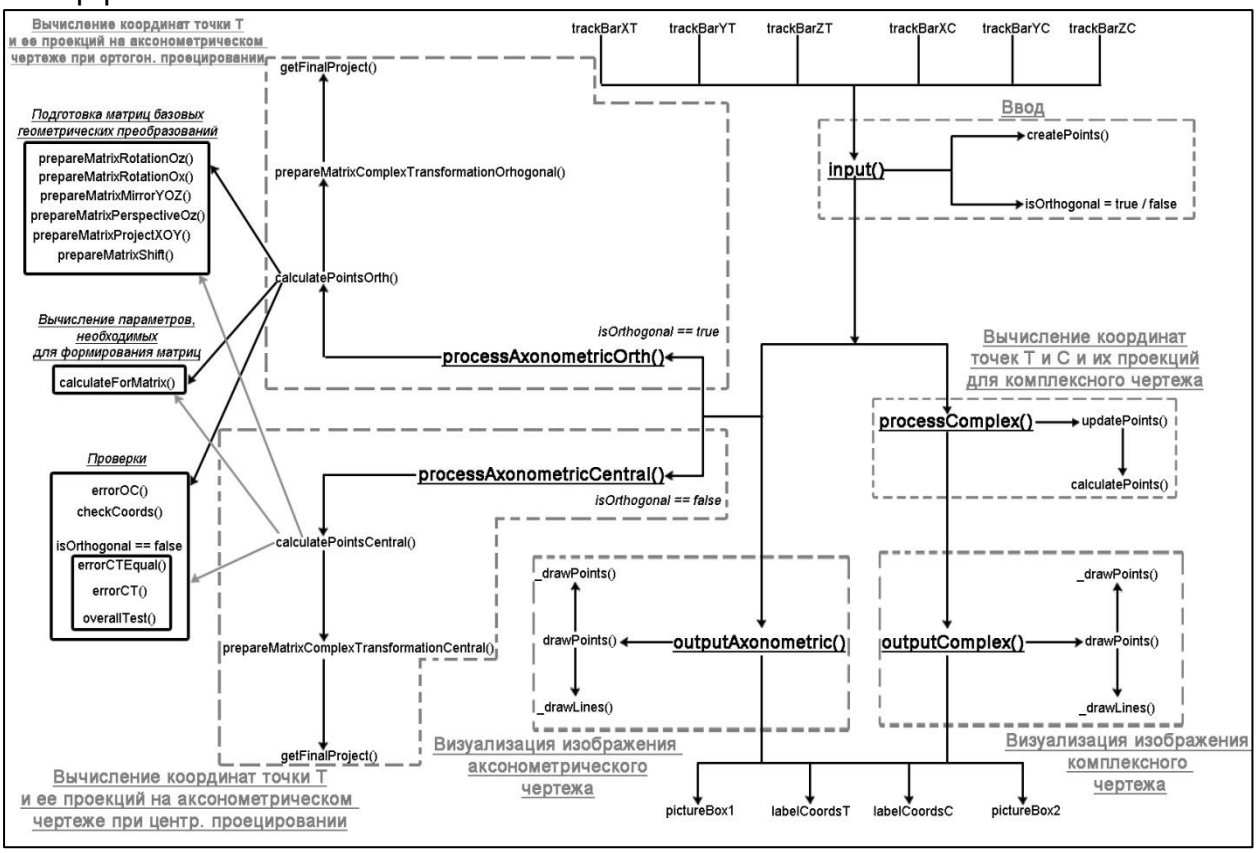


Рис.15 Функциональная схема программы

Как видно из данной функциональной схемы, в программе выделены те самые 6 “листьев”, которые приведены в структуре решения (на первом уровне детализации). Это является минимальным делением программы. На рисунке эти 6 “листьев” данного деления выделены жирным и подчеркнуты.

Также отдельно показаны все функции для подготовки матриц и для проверок.

2. Как программа работает?

При запуске программы происходит инициализация главной формы, создание переменных: **xT, yT, zT, xC, yC, zC, isOrthogonal** и резервирование памяти для массивов:

CPoint3D **storage3DPointT[14];**

CPoint3D **storage3DPointC[14];**

CPoint4D **storage4DPointT[11];**

CPoint4D **storage4DPointC[11];**

Далее создаются объекты классов аксонометрического и комплексного чертежей:

AxonometricDrawing **axonometricDrawing**;
(после чего резервируется память для массивов:
CPoint4D **storageAfterTransPointT[11]**;
CPoint2D **storage2DPointT[11]**;
и создаются переменные
cosRz, sinRz, cosRx, sinRx, c, Rz, Rx, Mx, Cz, Pz, T, changeC)
Также вызывается конструктор класса AxonometricDrawing, где changeC = true.

ComplexDrawing **complexDrawing**;
(после чего резервируется память для массивов:
CPoint4D **storage2DPointT[12]**;
CPoint2D **storage2DPointC[12]**;)

Далее происходит создание всех элементов интерфейса, после чего вызываются следующие события, связанные с ними:

pictureBox1
Вызывается обработка события отрисовки изображения аксонометрического чертежа на данном поле рисования **pictureBox1.Paint**, которая в свою очередь вызывает следующие функции по порядку:

1. **input()**, которая:
    a. присваивает переменной **isOrthogonal** значение 1, если пользователь выбрал радиокнопку **radioButtonOrthogonal**(ортогональное проецирование), и 0, если выбрана радиокнопка **radioButtonCentral**(центральное проецирование).
    b. вызывает функцию **createPoints()**, где массивы **storage3DPointT[14],storage3DPointC[14],storage4DPointT[11],storage4DPointC[11]** заполняются значениями, введенными пользователем на ползунках **trackBarXT, trackBarYT, trackBarZT, trackBarXC, trackBarYC, trackBarZC.**

2. **errorOC()**(при любом проецировании) и **errorCT(), errorCT()** (если выбрано центральное проецирование)**,**которые проверяют возможно ли построение аксонометрического чертежа (если хотя бы одна их функций вернет значение false, то отрисовка аксонометрического чертежа невозможна, следовательно, дальнейшие действия выполняться не будут, и выведется соответствующее сообщение об ошибке).

3. Если выбрано ортогональное проецирование (isOrhogonal == 1), то вызывается функция **processAxonometricOrth()**, которая в свою очередь вызывает метод класса

AxonometricDrawingaxonometricDrawing.**calculatePointsOrth()**, где происходит пересчет всех начальных однородных координат точек из массивов **storage4DPointT[11], storage4DPointC[11].** То есть вызывается функция
calculateMatrix() где вызываются следующие функции:

- **calculateForMatrix()** подсчет параметров, необходимых для формирования матриц и подготовки матриц базовых геом. преобразований.
- **checkCoords()** (для проверки координат, чтобы избежать ошибки деления на 0)
- **prepareMatrixRotationOz()**,
  prepareMatrixRotationOx(),
  prepareMatrixMirrorYOZ(),
  prepareMatrixProjectXOY(),
  prepareMatrixShift()
  (подготовки матриц базовых геом. преобразований)

Затем вызываются следующие функции:
prepareMatrixComplexTransformationOrthogonal(),
getFinalProject()

Важно: функции подготовки матриц базовых и сложногогеом. преобразований вызываются только в том случае, если значение переменной changeC равно true (то есть только в тех случаях, если произошел запуск программы или изменились координаты точки С)

После последовательного выполнения данных функций записываем получившиеся из начальных новые однородные координаты в массив **storageAfterTransPointT[11]**, а затем в **storage2DPointT[11]** (записываем сразу, так как при ортогональном проецировании w=1,и делить на него каждую координату не нужно).

Если выбрано центральное проецирование (isOrthogonal == 0), то вызывается функция **processAxonometricCentral()**, которая в свою очередь вызывает метод класса
AxonometricDrawingaxonometricDrawing.**calculatePointsCentral()**, где происходит пересчет всех начальных однородных координат точек из массивов **storage4DPointT[11], storage4DPointC[11].** То есть вызывается функция:**calculateMatrix()** где вызываются следующие функции:

- **calculateForMatrix()** подсчет параметров, необходимых для формирования матриц и подготовки матриц базовых геом. преобразований.
- **checkCoords()** (для проверки координат, чтобы избежать ошибки деления на 0)

- **prepareMatrixRotationOz()**,
  prepareMatrixRotationOx(),
  prepareMatrixMirrorYOZ(),
  prepareMatrixPerspectiveOz(),
  prepareMatrixProjectXOY(),
  prepareMatrixShift()
  (подготовки матриц базовых геом. преобразований)
  Затем вызываются следующие функции:

prepareMatrixComplexTransformationOrthogonal(),
getFinalProject()

Важно: функции подготовки матриц базовых и сложногогеом. преобразований вызываются только в том случае, если значение переменной changeC равно *true*.(то естьтолько в тех случаях, если произошел запуск программы или изменились координаты точки С).

После последовательного выполнения данных функций записываем получившиеся из начальных новые однородные координаты в массив **storageAfterTransPointT[11]**, а затем, поделив каждую координату на масштабный множитель w (который при центральном проецировании не равен 1), в **storage2DPointT[11]**.

Важно: Все действия, перечисленные выше при центральном проецировании, не относятся к трем точкам, задающим координатные оси (Tox, Toy, Toz).
При центральном проецировании вполне возможны выходы какой-либо точки за границы экрана, из-за чего изображение аксонометрического чертежа не визуализируем. Но если за границы экрана “вышла” хотя бы одна из точек Tox, Toy, Toz, а в остальном чертеж отображается корректно, было бы неправильным вместо визуализации чертежа выводить соответствующую ошибку. В целях недопущения выхода за границы экрана какой-либо из точек Tox, Toy, Toz будем умножать однородные координаты этих точек на матрицу сложного геом. преобразования при ортогональном проецировании, благодаря чему оси координат не будут “вылезать” за границы экрана.

4. **outputAxonometric()**(для отрисовки аксонометрического чертежа), которая вызывает функцию **overallTest()**для проверки выхода какой-либо точки за границы экрана. Если данная функция вернет *true*, вызывается функция **drawPoints()**, которая в свою очередь вызывает функции **_drawPoints()**и **_drawLines()**для отрисовки точек и линий связи соответственно. Если же **overallTest()**вернет *false*, то выводится соответствующее сообщение об ошибке.

pictureBox2
Вызывается обработка события отрисовки изображения комплексного чертежа на данном поле для рисования **pictureBox2.Paint**. Данная обработка аналогична той, что рассматривались в 1-й лабораторной работе.

При изменении положения ползунковых переключателей **trackBarXT, trackBarYT, trackBarZT, trackBarXC, trackBarYC, trackBarZC** вызывается событие **Scroll**, в обработке которого фиксируются значения текущих величин на соответствующих **Lable**:
labelCoordsT.Text = **trackBarXT**.Value.ToString() + **trackBarYT**.Value.ToString() + **trackBarZT**.Value.ToString()
labelCoordsC.Text = **trackBarXC**.Value.ToString() + **trackBarYC**.Value.ToString() + **trackBarZC**.Value.ToString()

Также при изменении положения ползунковых переключателей **trackBarXT, trackBarYT, trackBarZT** (то есть при изменении какой-либо координаты точкиT) полю класса AxonometrixDrawing **changeC** присваивается значение false, а при изменении положения ползунковых переключателей**trackBarXC, trackBarYC, trackBarZC**(то есть при изменении какой-либо координаты точки C) присваивается значение true. Это необходимо для того, чтобы не заново не пересчитывать все матрицы, так как они не изменятся при изменении координат точки T. То есть, пересчитываем матрицы только в случае изменения координат точки C, тем самым избавившись от лишних вычислений.

Кроме того, событие **Scroll,** вызывает функцию обновления (перерисовки) изображения**Refresh()** (для изображений аксонометрического и комплексного чертежей в случае изменения значений координат точки T и C), после чего происходит событие Paint для аксонометрического чертежа (**pictureBox1**) и событиеPaint для комплексного чертежа (**pictureBox2**).

При нажатии на кнопку **buttonExit** происходит событие **Click**, при обработке которого вызывается функция Application.Exit(), которая заканчивает работу программы.

Элементы интерфейса
labelAxonometricDrawing, labelComplexDrawing, labelXT, labelYT, labelZT, labelMinXT, labelMinYT, labelMinZT, labelMaxXT, labelMaxYT, labelMaxZT, labelXC, labelYC, labelZC, labelMinXC, labelMinYC, labelMinZC, labelMaxXC, labelMaxYC
Служат для простого отображения необходимых подписей. Пользователь никак не взаимодействует с ними, и они остаются неизменными.

Руководство пользователя

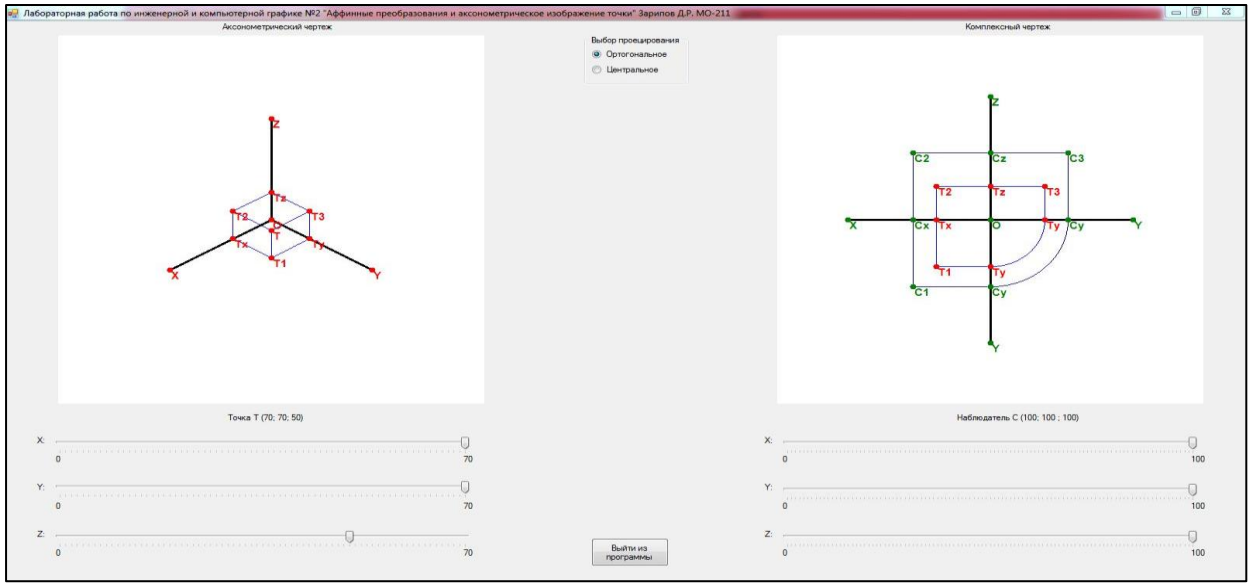


Рис.16

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

Рассмотрим рисунок 16.

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 Ти C. Соответствующие наименования точек на обоих изображениях находятся справа от самих точек.

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

Минимальные и максимальные возможные значения величин находятся рядом (соответственно с левой и правой сторон) с соответствующими им ползунковым переключателями.
Сверху соответствующих ползунковых переключателей находятся значения текущих установленных координат точек T и C.

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

Заключение (результаты)

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 C, позволяющей динамически менять значения координат точек T и C и выбирать тип проецирования между ортогональным и центральным.