Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

Уфимский государственный авиационный технический университет

Кафедра вычислительной математики и кибернетики

Пояснительная записка

к лабораторной работе №2

по дисциплине «Компьютерная графика»

по теме

«Аффинные преобразования и аксонометрическое изображение точки»

Выполнил: студент группы МО-304

Данилов С.О.

Проверил: профессор кафедры ВМиК

Верхотуров М.А.

Уфа - 2016

Оглавление

[1. Постановка задачи 3](#_Toc449897103)

[Содержательная постановка задачи 3](#_Toc449897104)

[Формальная постановка задачи 3](#_Toc449897105)

[2. Теоретическая часть 4](#_Toc449897106)

[3. Структура решения 9](#_Toc449897107)

[4. Обзор и анализ методов решения 10](#_Toc449897108)

[1. Ввод исходных данных. 10](#_Toc449897109)

[Ввод координат точек наблюдателя и пространства 10](#_Toc449897110)

[Переключение между центральным и ортогональным проецированием 10](#_Toc449897111)

[2. Перевод трёхмерных координат в координаты экрана. 10](#_Toc449897112)

[5. Отрисовка (вывод чертежей на экран). 18](#_Toc449897113)

[6. Описание реализации применяемых методов 18](#_Toc449897114)

[6.1 Укрупненный алгоритм. 19](#_Toc449897115)

[6.1.1 Ввод координат точек 19](#_Toc449897116)

[6.1. Вычисления 19](#_Toc449897117)

[6.1.3 Прорисовка. 21](#_Toc449897118)

[1.2 Детализированный алгоритм 22](#_Toc449897119)

[6.2.1. Вычисления 22](#_Toc449897120)

[6.2.3 Прорисовка. 27](#_Toc449897121)

[7. Руководство программиста. 28](#_Toc449897122)

[8. Основная форма. 32](#_Toc449897123)

[9. Руководство пользователя. 33](#_Toc449897124)

[10. Результаты 34](#_Toc449897125)

# 1. Постановка задачи

## Содержательная постановка задачи

Содержание экрана:

• комплексный чертеж точки *T*(только первый октант) с изображением проекций *Т*1, *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)*;

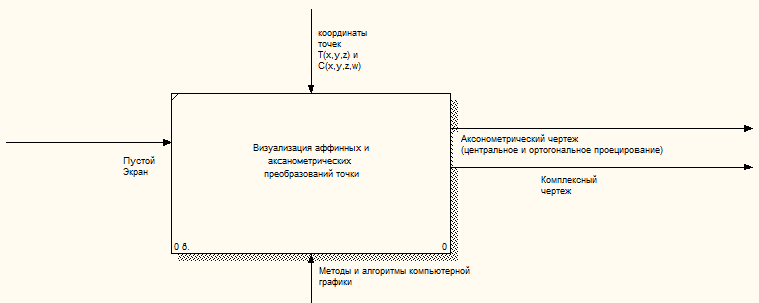
• ползунковые переключатели (3 шт.) для интерактивного изменения координат *(x,y,z)* точек *T, С* (с возможностью выбора одной из них для которой, в текущий момент времени, будет осуществляться изменение координат);

• переключатель *w=0; w=1* для смены ортогонального проецирования на центральное.

Динамика:

при изменении координат точек *Т*и*С* должны изменяться соответствующие чертежи.

## Формальная постановка задачи



# 2. Теоретическая часть

При рассмотрении вопросов, связанных с проецированием, имеет место проблема несоответствия оригиналов и проекций друг другу, соответственно необходимо сказать о расширении Евклидова пространства несобственными точками, прямыми и плоскостями.

Так вот, математическим аппаратом, который реализует это расширение, принято считать – однородные координаты. Эти координаты имеют широкое применение в качестве расширения Евклидова пространства в компьютерной графике и вычислительной геометрии.

Идея этого математического аппарата зародилась в тот момент, когда возникла необходимость в операции обратного проецирования, где из n + 1 мерного пространства необходимо получить n мерное.

Причиной применения данного математического аппарата является наличие, так называемой, бесконечно удаленной точки, которая имеет большое значение в практическом применении данного аппарата, поскольку её введение позволяет утверждать, что любые две прямые пересекаются в одной точке. Аналогично, в проективной геометрии можно утверждать о наличии линии пересечения двух плоскостей. Если плоскости параллельны, то все точки этой линии пересекаются в однородных координатах (X,Y,Z,0).

**Геометрические преобразования в трёхмерном пространстве**

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

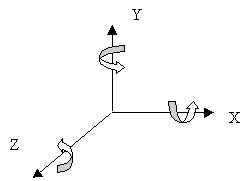
,

или в матричном виде:

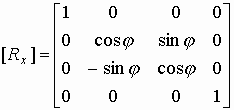


Рассмотрим матрицы, соответствующие следующим базовым геометрическим преобразованиям точки:

1. Повороты

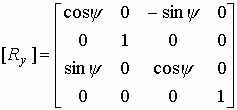


* вокруг оси *X*на угол φ (против часовой стрелки, если смотреть с бесконечности на оси Х с начала координат)



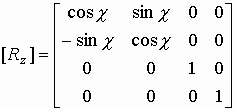
Матрица (2.1)

* вокруг оси *Y*на угол ψ (против часовой стрелки, если смотреть с бесконечности на оси Y с начала координат)



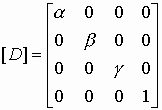
Матрица (2.2)

* вокруг оси *Z*на угол χ (против часовой стрелки, если смотреть с бесконечности на оси Z с начала координат)



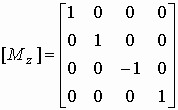
Матрица (2.3)

2. Растяжение (сжатие):

,   
если http://vmk.ugatu.ac.ru/Image76.gif - растяжение, http://vmk.ugatu.ac.ru/Image77.gif - сжатие

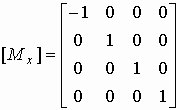
3. Отражение (зеркалирование)

* относительно плоскости *XOY*



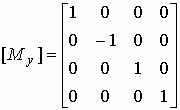
Матрица (2.4)

* относительно плоскости *YOZ*



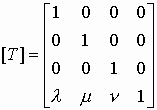
Матрица (2.5)

* относительно плоскости *ZOX*



Матрица (2.6)

4. Перенос (сдвиг, перемещение) на вектор http://vmk.ugatu.ac.ru/Image81.gif



Матрица T (2.7)

5. Проецирование

* на плоскость YOZ(при расположении наблюдателя в бесконечности на оси X(в точке (1,0,0,0)):

Матрица

* на плоскость XOZ(при расположении наблюдателя в бесконечности на оси Y(в точке (0,1,0,0)):

Матрица

* на плоскость XOY(при расположении наблюдателя в бесконечности на оси Z(в точке (0,0,1,0)):

Матрица

1. Перспективное преобразование

Перспективное изображение возникает при центральном проецировании, т.е. когда центр проецирования (глаз наблюдателя) находится на конечном расстоянии от экрана.

1. при расположении наблюдателя на оси X в точке (c,0,0,1)

Матрица (2.11)

1. при расположении наблюдателя на оси Y в точке (0,с,0,1)

Матрица (2.12)

1. при расположении наблюдателя на оси Z в точке (0,0,с,1)

Матрица (2.13)

## 

Необходимо помнить о том, что преобразование точки (с сохранением расположения исходной системы координат) соответствует выполнению обратной операции по отношению к преобразованию системы координат. Например, поворот точки на некоторый угол по часовой стрелке вокруг оси X соответствует повороту системы координат против часовой стрелки на тот же угол.

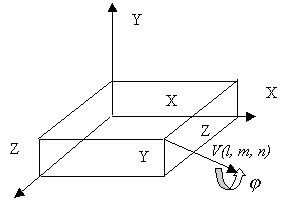
Все остальные преобразования, кроме вышеперечисленных, относятся к сложным преобразованиям. Матрицы сложных преобразований получаются при помощи нахождения произведения матриц, соответствующих базовым геометрическим преобразованиям, которые необходимо осуществить для получения искомого сложного преобразования (\*-умножение производится строго в том порядке, в котором эти базовые преобразования производятся, т.к. преобразования некоммутативны).

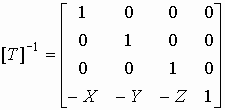
В качестве примера рассмотрим сложное преобразование, заключающееся во вращении на угол http://vmk.ugatu.ac.ru/Image61.gif вокруг прямой, проходящей через точку *T(X, Y, Z)*и имеющую направляющий вектор *V(l, m, n),*причем *l2+m2+n2=1*, т.е. вектор *V*является единичным.

Необходимо разложить преобразование на ряд элементарных шагов (базовых преобразований).

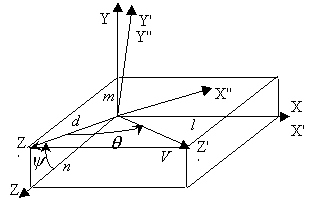
Цель: развернем систему координат так, чтобы ось *Z*совпала с *V*, после чего поворот на угол http://vmk.ugatu.ac.ru/Image61.gif будет возможно произвести путем осуществления базового преобразования - поворота на этот угол вокруг оси *Z.*Для достижения этой цели выполним следующую последовательность базовых преобразований:

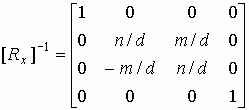
1. Перенос вектора *V* в начало координат:





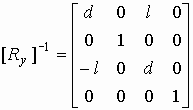
1. Поворот системы координат вокруг оси *Z*на угол http://vmk.ugatu.ac.ru/Image63.gif вокруг оси *X (*т.к. разворачиваем "систему координат" по часовой стрелке, то это тоже самое, что разворот точки против часовой стрелки).

   
http://vmk.ugatu.ac.ru/Image84.gif



1. Поворот системы координат вокруг оси ординат на угол http://vmk.ugatu.ac.ru/Image86.gif

http://vmk.ugatu.ac.ru/Image87.gif



1. Поворот точки вокруг *V* на угол http://vmk.ugatu.ac.ru/Image61.gif, а т.к. *V*совпадает с осью аппликат (матрица (2.3))

А так как нам необходимо вернуться в исходную систему координат, то:

1. Поворот вокруг оси ординат на угол "http://vmk.ugatu.ac.ru/Image90.gif" - [*Ry*]
2. Поворот вокруг оси абсцисс на угол "http://vmk.ugatu.ac.ru/Image91.gif" - [*Rx*]
3. Перенос на вектор *T(X, Y, Z).*

Результирующая матрица имеет следующий вид:

http://vmk.ugatu.ac.ru/Image92.gif  
  

# 3. Структура решения

1. Ввод координат точки пространства и точки наблюдателя
2. Перевод трёхмерных координат в координаты экрана.

а) Расчет координат точки Т и ее проекций на аксонометрическом чертеже.

1) При ортогональном проецировании.

2) При центральном проецировании.

b) Преобразование координат точек пространства и точки наблюдателя на комплексном чертеже.

1. Вывод на экран.
   1. Вывод на аксонометрическом чертеже.
   2. Вывод на комплексном чертеже.

# 4. Обзор и анализ методов решения

## Ввод исходных данных.

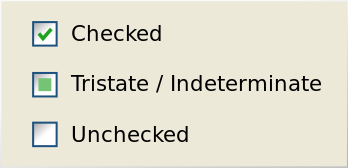
### Ввод координат точек наблюдателя и пространства

В программе должно быть реализовано динамическое изменение координат точки и угла между осями. Современные средства разработки приложений содержат в себе стандартные компоненты, с помощью которых можно обеспечить данную функциональность: переключатели (radioButton), флажки (checkBox), выпадающие списки (comboBox), ползунковые переключатели (Slider или Scale). В рамках данной программы наиболее удобным будет использование ползунковых переключателей. У данного компонента имеются такие свойства как Maximum (максимальное значение), Minimum (минимально допустимое значение) и Value (текущее значение переключателя). С помощью этих свойств можно реализовать выбор значения.

### Переключение между центральным и ортогональным проецированием

В программе должно быть реализовано динамическое перестроение аксонометрического чертежа из ортогонального проецирования в центральное, и наоборот. Для этих целей обычно используют элементы пользовательского интерфейса:

* https://upload.wikimedia.org/wikipedia/commons/c/cb/Radio_button.pngрадио кнопка (radio button)

* флаговая кнопка (check box)

Принципиальное отличие данных элементов пользовательского интерфейса в том, что флаговая кнопка, в отличии от радио кнопки, может быть установлена в трех положениях, при этом совсем не обязательно, чтобы хотя бы одна из их тандема была выбрана (установлена галочка внутри). Говоря простым языком: блок из радиокнопок может иметь только одну выбранную радиокнопку, а в блоке из флаговых кнопок могут быть выбраны все одновременно. Понятно, что в зависимости от выбранного варианта способ реализации контроля над отображением сильно не изменится, однако большую наглядность при использовании программы будет обладать радиокнопка, так как будет максимально прозрачно для пользователя с каким видом проецирования он работает.

Помимо изменения способов отображения чертежа, необходимо так же динамически переключаться между введением координат точки пространства и точки наблюдения, что так же будет входить в обязанности переключателя.

*Обобщенный алгоритм:*

* + - Считать значение ползункового переключателя.
    - Записать это значение в соответствующую переменную
* Считать значение соответствующего переключателя
* Записать в соответствующую переменную

## 2. Перевод трёхмерных координат в координаты экрана.

Для построения чертежей на экране необходимо перевести координаты из реальной трёхмерной системы координат в двумерную систему координат экрана, а также определить по трёхмерной координате координаты на комплексном чертеже и перевести их в двумерную систему координат экрана.

Для построения комплексного чертежа методы построения ничем не отличаются от методов, рассмотренных в предыдущей лабораторной работе.

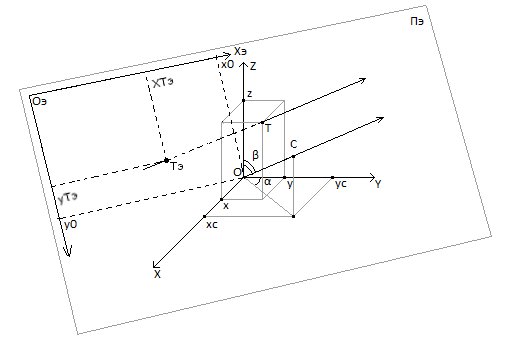
**Входные данные:**

* - координаты точки T в трёхмерном пространстве.
* - координаты наблюдателя C в трехмерном пространстве.

**Выходные данные:**

* экранные координаты точек системы координат.
* экранные координаты точки Т (x, y, z) -
* экранные координаты точек проекций T1, T2, T3 на плоскости П1, П2 и П3 соответственно.
* экранные координаты точек Tz, Ty, Tz на оси OX, OY и OZ соответственно.

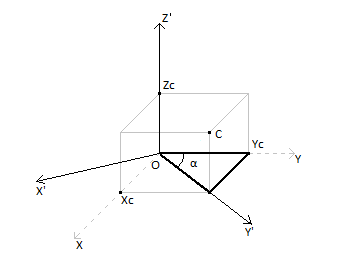
1. ***Для ортогонального проецирования***

При ортогональном проецировании проецирующие лучи перпендикулярны плоскости проекции и параллельны друг другу. Так как у нас нет простого геометрического преобразования, соответствующего ортогональному проецированию при расположении наблюдателя на произвольной прямой, но зато есть операция проецирования при расположении наблюдателя в бесконечности на одной из осей ((2.8), , (2.10)), для выполнения задачи необходимо совместить одну из осей (например, ось Z) с вектором точки наблюдения, и применить операцию проецирования на плоскость XOY, а так же направить оси X и Y соответственно осям и для правильного изображение чертежа на экране. 

Разобьем соответствующее сложное преобразование системы координат на следующую последовательность базовых:

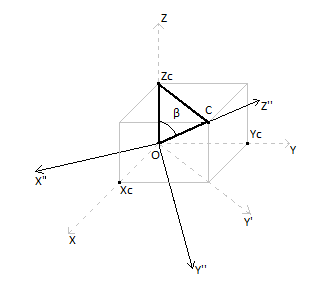
**Шаг 1.**

Поворот системы координат вокруг оси Z по часовой стрелке на угол . Это равносильно повороту точки вокруг оси Z против часовой стрелки (матрица (2.3)). Из рисунка, изображающего соответствующее преобразование, находим значения и :



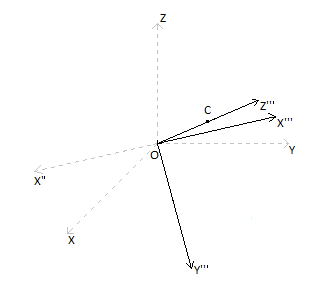
При xC = yC = 0 возникает ошибка деления на ноль. Но такие координаты означают, что камера расположена на оси Z, поэтому необходимости в поворотах системы координат нет, и матрицу можно заменить единичной, т.е и .

**Шаг 2.**

 Поворот системы координат вокруг оси X’ по часовой стрелке на угол . Это равносильно повороту точки вокруг оси X’ против часовой стрелки. (матрица (2.1)). Из рисунка, изображающего соответствующее преобразование, находим значения и :

**Шаг 3.**

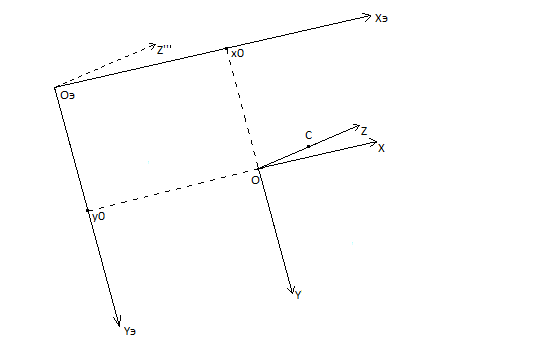
В результате предыдущих поворотов, ось X’’ будет направлена в сторону, противоположную оси Xэ экранной системы координат. В базовых преобразованиях рассматривается преобразование точки, но в данном случае это будет тоже самое. Для этого применим операцию отражения точки относительно плоскости YOZ (Матрица (2.5)).



**Шаг 4.**

Проецирование на плоскость XOY (Матрица (при расположении наблюдателя в бесконечности на оси Z).

**Шаг 5.**



Перенос точки на вектор ω (x0, y0, 0) (или перенос системы координат на вектор (-x0,- y0,0) ), где x0, y0 – экранные координаты O. Матрица этого преобразования имеет вид:

Матрица Т(2.а.0)

Таким образом, матрица сложного преобразования для ортогонального проецирования равна:

С помощью матрицы мы можем найти необходимую ортогональную проекцию любой точки. Для этого достаточно вектор-строку этой точки умножить справа на матрицу , в результате получим вектор-строку искомой проекции точки:­

Рассмотрим случай, когда изображение построить невозможно: при однородных координатах точки С (0, 0, 0, w) ортогональной проекции не существует, т.к. не определен вектор направления проецирования OC.

В этом случае аксонометрический чертеж не рисуется, а на экран выводится сообщение об ошибке.

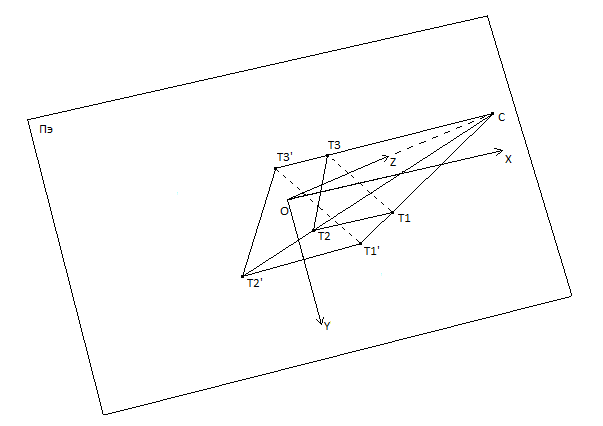
Проверка 0: Если точка С = С' (0,0,0), то вывести сообщение об ошибке.

**Обобщённый алгоритм:**

* 1. Проверка точки наблюдателя на нахождение в начале системы координат (Проверка 0).
  2. Вычисление матрицы поворота относительно оси Z - [Rz](используя матрицу 2.3) с проверкой деления на ноль
  3. Вычисление матрицы поворота относительно оси X - [Rx]( используя матрицу 2.1)
  4. Вычисление матрицы переноса в начало координат - [T] (матрица 2.а.0),
  5. Вычисление матрицы проецирования на плоскость XOY(при расположении наблюдателя в бесконечности на оси Z(в точке (0,0,1,0)) - [Pz] (матрица 2.10)
  6. Вычисление матрицы отражения относительно плоскости YOZ - [Mx](матрица 2.5)
  7. Вычисление матрицы ортогонального проецирования - [A]( используя формулу 2.а.1)
  8. Используя формулу (2.а.2) и матрицу ортогонального проецирования [A] найти:
     1. Экранные координаты точки Т
     2. Экранные координаты системы координат точек О, Х, Y, Z;
     3. Экранные координаты проекций точки T на плоскости П1, П2, П3–T1, T2,Т3
     4. Экранные координаты проекций точки T на координатные оси – Tx, Ty, Tz;

1. ***Для центрального проецирования***

Центральное проецирование используется для построения перспективного изображения, при этом центр проекции находится на конечном расстоянии от плоскости проецирования. Проецирующие лучи проходят через одну точку – центр проецирования (точка расположения камеры).



Последовательность базовых преобразований для центрального проецирования похожа на ту же что и для ортогонального проецирования. Однако, в ней присутствует еще одно базовое преобразование – перспективное преобразование.

Получение матрицы сложного преобразования для центрального проецирования получается следующей последовательностью базовых преобразований:

1. Поворот системы координат вокруг оси Z по часовой стрелке на угол .

2. Поворот системы координат вокруг оси X’ по часовой стрелке на угол .

3. Отражение относительно плоскости YOZ.

4. Перспективное преобразование при расположении наблюдателя на оси Z в точке (0, 0, c, 1).

5. Проецирование на плоскость XOY (при расположении наблюдателя в бесконечности на оси Z).

6. Перенос на вектор (x0, y0, 0).

Шаги 1-4, 6 аналогичны ортогональному проецированию. Матрица перспективного преобразования шага 5 та же, что и матрица (2.13). Необходимо учитывать, что после умножения на данную матрицу, однородные а станут равны (x, y, z, 1 – z/c). Соответствующие им неоднородные координаты равны

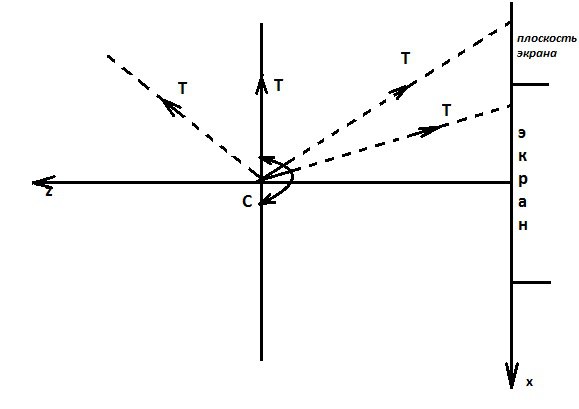
(x / (1 – z/c), y / (1 – z/c), z / (1 – z/c)). (2.б.0)

Таким образом, матрица сложного преобразования для центрального проецирования равна:

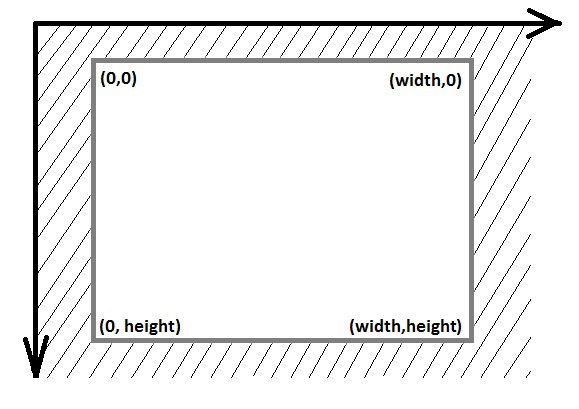
С помощью матрицы мы можем найти необходимую центральную проекцию любой точки. Для этого достаточно вектор-строку этой точки умножить справа на матрицу , в результате получим вектор-строку искомой проекции точки:­

Рассмотрим случаи, когда изображение построить невозможно:

1. Точка С совпадает с началом оси координат
2. Точка С совпадает с точкой пространства Т
3. Если угол γ между векторами СO и СT больше или равен π/2.



1. Выход проекций точки за пределы экрана.



Проверка 1. Если угол между векторами СO и СT больше или равен , то центральное проецирование осуществить невозможно, т.к. проецирующие лучи не пересекут плоскость проецирования.

Проверить угол можно через его косинус, который легко находится через скалярное произведение векторов. Если косинус неположительный, то центральная проекция невозможна.

CO = (xC, yC, zC), CT = (x – xC, y – yC, z – zC)

Т.к. и не влияет на знак , то это выражение можно не учитывать. В итоге, если

то на экран выводится сообщение об ошибке.

Проверка 2. Выход проекций точки за пределы экрана.

Координаты проекции необходимо проверять на возможность отображения на экране. Проверить это, имея исходные координаты точки и камеры сложно, т.к. необходимо найти точку пересечения прямой с плоскостью, в отличие от проверки на этапе, когда мы уже получили координаты точки в экранной системе координат. В этом случае проверка выхода точки за границы экрана реализуется обычными операциями сравнения с шириной и высотой областью для рисования. Если

то на экран выводится сообщение об ошибке (width и height – ширина и высота области для отображения).

Проверка 3. Совпадение точки пространства и точки наблюдателя.

Для проверки этого необходимо будет сравнить на равенство координаты точки пространства и точки наблюдателя. Следовательно, получаем условие вида:

, то вернуть ошибку. (2.б.5)

**Обобщённый алгоритм:**

* 1. Проверка точки наблюдателя на нахождение в начале системы координат (Проверка 0).
  2. Проверка совпадения точки наблюдателя и точки пространства (Проверка 3).
  3. Проверка положительности косинуса угла между векторами CO и CT (Проверка 1)
  4. Вычисление матрицы поворота относительно оси Z - [Rz](используя матрицу 2.3) с проверкой деления на ноль
  5. Вычисление матрицы поворота относительно оси X - [Rx]( используя матрицу 2.1)
  6. Вычисление матрицы переноса в начало координат - [T] (матрица 2.а.0),
  7. Вычисление матрицы проецирования на плоскость XOY(при расположении наблюдателя в бесконечности на оси Z(в точке (0,0,1,0)) - [Pz] (матрица 2.10)
  8. Вычисление матрицы отражения относительно плоскости YOZ - [Mx](матрица 2.5)
  9. Вычисление матрицы перспективного преобразования - ))
  10. Вычисление матрицы центрального проецирования (формула(2.б.1))
  11. Используя формулу (2.б.2) и матрицу центрального проецирования найти:
      + 1. Экранные координаты точки Т
        2. Экранные координаты системы координат точек: О, Х, Y, Z;
        3. Экранные координаты проекций точки T на плоскости П1, П2, П3–T1, T2,Т3
        4. Экранные координаты проекций точки T на координатные оси – Tx, Ty, Tz;
  12. Нормирование координат точек с помощью формулы (2.б.0).
  13. Проверка выхода проекции за границы экрана (Проверка 2)

# 5. Отрисовка (вывод чертежей на экран).

Данный пункт был рассмотрен в лабораторной работе №1.

# 6. Описание реализации применяемых методов

Точка в трехмерном пространстве определяется с помощью трех координат X, Y, Z, расположенные на осях Ox, Oy, Oz соответственно. В силу применения однородных координат для трехмерных точек, к трех существующим координатам добавится еще одна W. В двумерном пространстве точка определяется уже двумя координатами X, Y, расположенные на осях Ox, Oy. Так как необходимо спроецировать трехмерную систему координат и точки в ней в двумерную систему координат экрана, имеем тип двухмерных и четырехмерных точек.

Для хранения однородных координат точек трехмерного пространства будем использовать структуру Point4D, которая представляет упорядоченную пару целых чисел-координат X и Y, Z, W, определяющую точку в трехмерной плоскости в однородных координатах. Для хранения и обработки матриц будем использовать структуру Matrix, для которой определены операции между матрицами: сложение, умножение, и т.д. Для хранения координат точек двумерного пространства будем использовать структуру Point2D, которая представляет упорядоченную пару целых чисел-координат X и Y, определяющую точку в двумерной плоскости. Используемая константа для вычисления: int semiaxis\_w, int semiaxis\_h (полуоси, соотвественно равные половине ширины экрана и высоты). Входные параметры для вычисления – значения ползунков (int X, int Y, int Z, int alpha), значение RadioButton (bool Ортогональное).

**Имена используемых переменных и их смысл**

*Общие:* X, Y, Z-значения ползунков по осям Ox, Oy, Oz точки пространства.

– значения ползунков по осям Ox, Oy, Oz точки наблюдателя

C = – точка наблюдателя. В зависимости от того какой вид проецирования выбран, w будет равен: 1, если центральное проецирование; 0, если ортогональное проецирование.

semiaxis\_w (тип int) - полуось, равная половине ширины экрана

semiaxis\_h (тип int) - полуось, равная половине высоты экрана

*Единичная\_Матрица\_4\_на\_4 –* единичная матрица размерностью 4\*4

Позиция\_Переключателя\_Проецироания(тип bool) – значение RadioButton компонента. Если fasle, то пользователь выбрал «Центральное проецирование», иначе «Ортогональное проецирование».

***Для аксонометрического чертежа:***

Для хранения точек трехмерного пространства введем массив точек T4D[O3, x13, y13, z13, T3, T13, T23, T33, Tx3, Ty3, Tz3], каждая точка этой системы имеет тип Point4D, где

O3 - точка начала координат

T3 – исходная точка

T13, T23, T33– проекции точки T на плоскости П1, П2, П3 соответственно

Tx3, Ty3, Tz3– точка на осях X, Y, Z соответственно

x13, y13, z13– точки для построения осей координат.

Так как нам нужно преобразовать трехмерную систему координат в двумерную систему экрана, то введем массив T2D\_area[O2, x12, y12, z12, T2, T12, T22, T32, Tx2, Ty2, Tz2] определяющий двумерную систему координат экрана, где каждая точка этой системы имеет тип Point2D.

***Для комплексного чертежа:***

Данный пункт рассматривался в лабораторной работе №1.

**6.1 Укрупненный алгоритм.**

**6.1.1 Ввод координат точек**

Для ввода координат и угла между осями используется класс TrackBar (ползунковый переключатель), у которого будут использоваться следующие свойства:

|  |  |
| --- | --- |
| Свойство | Значение |
| Maximum | Максимальное значение |
| Minimum | Минимальное значение |
| Value | Текущая позиция |

Свойство Value определяет текущее положение ползунка. Для ввода координат точки и угла alpha воспользуемся именно этим свойством. Значение координат точки целочисленные. Таким образом, точка T3 (исходная точка) из массива T4D задается с помощью значений свойств Value ползунков, представляющих собой значения соответствующих координат.

**6.1. Вычисления**

1. ***Для аксонометрического чертежа***

Трехмерная система координат представляют собой массив точек T4D[O3, x13, y13, z13,T3, T13, T23, T33, Tx3, Ty3, Tz3]. Так как нам нужно преобразовать трехмерную систему координат в двумерную систему экрана, то точки системы экрана будем хранить в массиве T2D\_area[O2, x12, y12, z12, T2, T12, T22, T32, Tx2, Ty2, Tz2]. Добавим значения точек в массив T4D:

T4D[1]:=(0,0,0,1)

T4D[2]:=(semiaxis\_w,0,0,1)

T4D[3]:=(0,semiaxis\_w, 0, 1)

T4D[4]:=(0,0, semiaxis\_h,1)

T4D[5]:=(X,Y,Z,1)

T4D[6]:=(X,Y,0,1)

T4D[7]:=(X,0,Z,1)

T4D[8]:=(0,Y,Z,1)

T4D[9]:=(X,0,0,1)

T4D[10]:=(0,Y,0,1)

T4D[11]:=(0,0,Z,1)

Преобразование трехмерной системы координат в двумерную систему экрана осуществляется: сначала определяется центр экрана с помощью выполнения шагов из пункта 2.а, затем, применяя к каждой точке массива T4D формулу 2.а.2 и взяв первые две координаты получившейся точки, получаем элемент массива T2D\_area. Центральное проецирование строится так же, за исключением того, что вместо формулы (2.а.2) будет применятся, для получения массива T2D\_area из T4D, формула (2.б.2).

Точки x13, y13, z13 массива T4D используют в качестве своих координат значение полуосей semiaxis\_w, semiaxis\_h.

**Ортогональное проецирование**

1. Проверка 0 (точка наблюдателя в начале системы координат).

Для выполнения этой проверки необходимо, чтобы координаты точки С.X, C.Y, C.Z были не равны 0. Если это условие не выполняется, то вернуть ошибку.

1. Для вычисления матрицы поворота относительно оси Z [Rz](матрица 2.3), нужно подсчитать значения

(6.2.2)

Если С.Y или C.Z окажутся равными 0, то необходимо матрицу [Rz] заменить единичной матрицей.

1. Для вычисления матрицы поворота относительно оси X [Rx](матрица 2.1), нужно подсчитать значения
2. Для вычисления матрицы переноса на вектор [T](матрица 2.а.0) необходимы экранные координаты точки О: О.Х = semiaxis\_w, O.Y = semiaxis\_h
3. Вычисление матрицы проецирования на плоскость XOY [Pz] (матрица 2.10)
4. Вычисление матрицы отражения относительно оси X [Mx](матрица 2.5)
5. Матрица ортогонального проецирования [A] рассчитывается, используя формулу (2.а.1).
6. Используя формулу (2.а.2) и матрицу ортогонального проецирования [A] найти координаты проекций всех точек массива T4D:

Цикл по индексам массива T4D

Начало

* + - * 1. *Умножить точку T4D, соответствующую текущему индексу, на матрицу [A]*
        2. *Занести полученные координаты X и Y точки T4D, соответствующей текущему индексу, в координаты точки T2D\_area, соответствующей текущему индексу.*

Конец

**Центральное проецирование**

1. Проверка 3 (совпадения точек пространства и наблюдателя)

Для выполнения этой проверки необходимо, чтобы координаты точки С.X, C.Y, C.Z были не равны соответствующим координатам точки T. Если это условие не выполняется, то вернуть ошибку.

1. Проверка 1 (положительности косинуса угла между векторами CO и CT)
2. Проверка 0 (точка наблюдателя в начале системы координат).

Для выполнения этой проверки необходимо, чтобы координаты точки С.X, C.Y, C.Z были не равны 0. Если это условие не выполняется, то вернуть ошибку.

1. Для вычисления матрицы поворота относительно оси Z [Rz](матрица 2.3), нужно подсчитать значения

(6.2.2)

Если С.Y или C.Z окажутся равными 0, то необходимо матрицу [Rz] заменить единичной матрицей.

1. Для вычисления матрицы поворота относительно оси X [Rx](матрица 2.1), нужно подсчитать значения
2. Для вычисления матрицы переноса на вектор [T](матрица 2.а.0) необходимы экранные координаты точки О: О.Х = semiaxis\_w, O.Y = semiaxis\_h
3. Вычисление матрицы проецирования на плоскость XOY [Pz] (матрица 2.10)
4. Вычисление матрицы отражения относительно оси X [Mx](матрица 2.5)
5. Вычисление матрицы перспективного преобразования
6. Вычисление матрицы центрального проецирования (формула(2.б.1))
7. Используя формулу (2.а.2) и матрицу центрального проецирования [] найти координаты проекций всех точек массива T4D:

Цикл по индексам массива T4D

Начало

* + - * 1. *Умножить точку T4D, соответствующую текущему индексу, на матрицу []*
        2. *Нормирование точки T4D, соответствующей текущему индексу, c помощью формулы (2.б.0)*
        3. *Проверка 2 (выход проекции за границы экрана)*

*Если выполняется условие (2.б.4) по отношению к точке T4D соответствующей текущему индексу,*

*То выдать сообщение об ошибке.*

* + - * 1. *Занести полученные координаты X и Y точки T4D, соответствующей текущему индексу, в координаты точки T2D\_area, соответствующей текущему индексу.*

Конец

1. ***Для комплексного чертежа***

Данный пункт рассматривался в лабораторной работе №1.

**6.1.3 Прорисовка.**

1. ***Для аксонометрического чертежа***

***Отрисовка осей:***

1. Соединить точки O и x12 из массива *T2D\_area* с помощью метода DrawLine, а именно точки *T2D\_area[1] и T2D\_area[2]*
2. Отрисовать строку “Х” на положительном конце оси методом DrawString
3. Соединить точки y12 и O из массива *T2D\_area* с помощью метода DrawLine, а именно точки *T2D\_area[3] и T2D\_area[1]*
4. Отрисовать строку “Y” на положительном конце оси методом DrawString
5. Соединить точки z12 и O из массива *T2D\_area* с помощью метода DrawLine, а именно точки *T2D\_area[1] и T2D\_area[4]*
6. Отрисовать строку “Z” на положительном конце оси методом DrawString

***Отрисовка точек:***

1. Для каждой точки из массива *T2D\_area*:
   1. Нарисовать круг с центром-координатами точки. Отрисовать методом DrawEllipse
   2. Подписать точку методом DrawString

***Отрисовка линий связи:***

1. Соединить точки T12 и Tх2 из массива *T2D\_area* линией с помощью метода DrawLine, а именно точки *T2D\_area[6]* и *T2D\_area[9]*
2. Соединить точки T12 и T2 из массива *T2D\_area* линией с помощью метода DrawLine, а именно точки *T2D\_area[6] и T2D\_area[5]*
3. Соединить точки T2 и T22 из массива *T2D\_area* линией с помощью метода DrawLine, а именно точки *T2D\_area[5] и T2D\_area[8]*
4. Соединить точки T22 и Tх2 из массива *T2D\_area* линией с помощью метода DrawLine, а именно точки *T2D\_area[8] и T2D\_area[9]*
5. Соединить точки T22 и Tz2 из массива *T2D\_area* линией с помощью метода DrawLine, а именно точки *T2D\_area[7] и T2D\_area[11]*
6. Соединить точки T2 и T32 из массива *T2D\_area* линией с помощью метода DrawLine, то а именно точки *T2D\_area[7] и T2D\_area[8]*
7. Соединить точки T12 и Ty2 из массива *T2D\_area* линией с помощью метода DrawLine, а именно точки *T2D\_area[6] и T2D\_area[10]*
8. Соединить точки Tz2 и T32 из массива *T2D\_area* линией с помощью метода DrawLine, а именно точки *T2D\_area[11] и T2D\_area[8]*
9. Соединить точки T32 и Ty2 из массива *T2D\_area* линией с помощью метода DrawLine, а именно точки *T2D\_area[8] и T2D\_area[10]*
10. ***Для комплексного чертежа***

Рассматривалось в лабораторной работе №1.

* 1. **Детализированный алгоритм**

**6.2.1. Вычисления**

1. ***Для аксонометрического чертежа***

**Ортогональное проецирование**

1. Проверка 0 (точка наблюдателя в начале системы координат).

*Проверка\_0()*

*Начало*

*Если С.Х = 0 и С.У = 0 и С.Z = 0 тогда*

*Вернуть Ошибка;*

*КонецЕсли*

*Конец*

1. Вычисление матрицы поворота относительно оси Z [Rz](матрица 2.3)

*Матрица\_Rz()*

*Начало*

*Matrix Rz = Единичная\_Матрица\_4\_на\_4;*

*Если С.X <> 0 и C.Y <> 0 тогда*

*начало*

*Rz[0,0]=Rz[1,1]=cos(C.Y/sqrt(C.X^2+C.Y^2));//по формуле 6.2.1*

*Rz[1,0] = -sin (C.X/sqrt(C.X^2 + C.Y^2)); // по формуле 6.2.2*

*Rz[0,1]= - Rz[1,0];*

*конец*

*Вернуть Rz;*

*Конец*

1. Вычисление матрицы поворота относительно оси X [Rx](матрица 2.1)

*Матрица\_Rx()*

*Начало*

*Matrix Rz = Единичная\_Матрица\_4\_на\_4;*

*Rx[1,1]=Rx[2,2]=cos(C.Z/sqrt(C.X^2+C.Y^2+C.Z^2));//по формуле 6.2.3*

*Rx[1,0] = -sin(sqrt(С.X^2 + C.Y^2)/sqrt(C.X^2+C.Y^2+C.Z^2)); // по формуле 6.2.4*

*Rx[0,1] = Rx[1,0];*

*Вернуть Rx;*

*Конец*

1. Для вычисления матрицы переноса точки на вектор [T](матрица 2.а.0)

*Матрица\_T()*

*Начало*

*Matrix T = Единичная\_Матрица\_4\_на\_4;*

*T[0,3] = semiaxis\_w;*

*T[1,3] = semiaxis\_h;*

*Вернуть T;*

*Конец*

1. Вычисление матрицы проецирования [Pz](матрица 2.10)

*Матрица\_Pz()*

*Начало*

*Matrix Pz = Единичная\_Матрица\_4\_на\_4;*

*Pz[2,2] = 0;*

*Вернуть Pz;*

*Конец*

1. Вычисление матрицы зеркалирования(отражения) относительно оси Х [Mx](матрица 2.5)

*Матрица\_Mx()*

*Начало*

*Matrix Mx = Единичная\_Матрица\_4\_на\_4;*

*Mx[0,0] = -1;*

*Вернуть Mx;*

*Конец*

1. Матрица ортогонального проецирования [A] рассчитывается, используя формулу (2.а.1):

*Матрица\_А()*

*Начало*

*Matrix* Pz = *Матрица\_Pz();*

*Matrix* Rz = *Матрица\_Rz();*

*Matrix* Rx = *Матрица\_Rx();*

*Matrix* T = *Матрица\_T();*

*Matrix* Mx = *Матрица\_Mx();*

*Matrix A = Rz \* Rx \* Mx \* Pz \* T;// по формуле 2.а.1*

*Вернуть A;*

*Конец*

1. Используя формулу (2.а.2) и матрицу ортогонального проецирования [A] найти:
   * + 1. Координаты точки Т
       2. Координаты системы координат точек О, Х, Y, Z;
       3. Координаты проекций точки T на плоскости П1, П2, П3–T1, T2,Т3
       4. Координаты проекций точки T на координатные оси – Tx, Ty, Tz;

Для нахождения проекции точки введем следующую функцию, которая будет возвращать проекцию точки:

*РасчОртПроицир (X,Y,Z)*

*Начало*

*Matrix A = Матрица\_*

*Point4D P = [X, Y, Z, 1]\*A;*

*Вернуть P;*

*Конец*

Соответственно далее находим необходимые проекции точек с помощью этой функции:

* + - 1. Координаты точки Т

T = *РасчОртПроицир* (X, Y ,Z);

* + - 1. Координаты системы координат точек О, Х, Y, Z;

O= *РасчОртПроицир* (0,0,0)

X = *РасчОртПроицир* (semiaxis\_w,0,0)

Z = *РасчОртПроицир* (0,0, semiaxis\_h)

Y = *РасчОртПроицир* (0, semiaxis\_w, 0)

* + - 1. Координаты проекций точки T на плоскости П1, П2, П3–T1, T2,Т3

T1 = *РасчОртПроицир* (X, Y, 0);

T2 = *РасчОртПроицир* (X, 0, Z);

T3 = *РасчОртПроицир* (0, Y, Z);

* + - 1. Координаты проекций точки T на координатные оси – Tx, Ty, Tz;

Tx = *РасчОртПроицир* (X, 0, 0);

Ty = *РасчОртПроицир* (0, Y, 0);

Tz = *РасчОртПроицир* (0,0, Z);

**Центральное проецирование**

1. Проверка 2 (выхода проекции за границы экрана)

*Проверка\_2(Point4D T)*

*Начало*

*Если тогда*

*Вернуть Ошибка;*

*Конец*

1. Проверка 1 (положительности косинуса угла между векторами CO и CT)

*Проверка\_1(X,Y,Z)*

*Начало*

*Если С.X\*(X – C.X) + C.Y\*(Y – C.Y) + C.Z\*(Z – C.Z) =< 0 тогда*

*Вернуть Ошибка;*

*Конецесли*

*Конец*

1. Вычисление матрицы перспективного преобразования

*Матрица\_Cz()*

*Начало*

*Matrix Cz = Единичная\_матрица\_4\_на\_4;*

*Cz[3,2] = -1 / sqrt(C.Z ^2 + C.Y^2 + C.X^2);*

*Вернуть Cz;*

*Конец*

1. *В*ычисление матрицы центрального проецирования (формула(2.б.1))

*Матрица\_( )*

*Начало*

*Matrix* Pz = *Матрица\_Pz();*

*Matrix* Rz = *Матрица\_Rz(С);*

*Matrix* Rx = *Матрица\_Rx(С);*

*Matrix* T = *Матрица\_T();*

*Matrix* Mx = *Матрица\_Mx();*

*Matrix Cz = Матрица\_Cz();*

*Matrix Ac = Rz \* Rx \* Mx \* Cz \* Pz \* T;// по формуле 2.а.1*

*Вернуть Ac;*

*Конец*

1. Используя формулу (2.б.2) и матрицу центрального проецирования найти:
   * + 1. Координаты точки Т
       2. Проверка выхода проекции за границы экрана (условие (2.б.4))
       3. Координаты системы координат точек: О, Х, Y, Z;
       4. Координаты проекций точки T на плоскости П1, П2, П3–T1, T2,Т3
       5. Координаты проекций точки T на координатные оси – Tx, Ty, Tz;

Для нахождения проекции точки введем следующую функцию, которая будет возвращать проекцию точки

*РасчЦентрПроицир (X,Y,Z)*

*Начало*

*Проверка\_Положит\_Угла(X,Y,Z);*

*Matrix = Матрица\_*

*Point4D P = [X, Y, Z, 1] \* ;*

*Point4D T = Нормирование\_точки(P);*

*Вернуть T;*

*Конец*

Соответственно далее находим необходимые проекции точек:

* + - 1. Координаты точки Т

T2D\_area[4] *= РасчЦентрПроицир (T4D[4].X, T4D[4].Y , T4D[4].Z);*

* + - 1. Координаты системы координат точек: О, Х, Y, Z;

T2D\_area[0]= *РасчЦентрПроицир* (*T4D[0].X, T4D[0].Y , T4D[0].Z*)

T2D\_area[1]= *РасчЦентрПроицир* (*T4D[1].X, T4D[1].Y , T4D[1].Z*)

T2D\_area[2]= *РасчЦентрПроицир* (*T4D[2].X, T4D[2].Y , T4D[2].Z*

T2D\_area[3]= *РасчЦентрПроицир*(*T4D[3].X, T4D[3].Y , T4D[3].Z*)

* + - 1. Координаты проекций точки T на плоскости П1, П2, П3–T1, T2,Т3

T2D\_area[5]= *РасчЦентрПроицир* (*T4D[5].X, T4D[5].Y , T4D[5].Z*);

T2D\_area[6]= *РасчЦентрПроицир* (*T4D[6].X, T4D[6].Y , T4D[6].Z*);

T2D\_area[7]= *РасчЦентрПроицир* (*T4D[7].X, T4D[7].Y , T4D[7].Z*);

* + - 1. Координаты проекций точки T на координатные оси – Tx, Ty, Tz;

T2D\_area[8]= *РасчЦентрПроицир* (*T4D[8].X, T4D[8].Y , T4D[8].Z*);

T2D\_area[9]= *РасчЦентрПроицир* (*T4D[9].X, T4D[9].Y , T4D[9].Z*);

T2D\_area [10]=*РасчЦентрПроицир*(*T4D[10].X, T4D[10].Y , T4D[10].Z*);

1. Нормирование точки с помощью формулы 2.б.0

*Нормирование\_Точки(Point4D T)*

*Начало*

*Point4D P = [T.X / T.W, T.Y / T.W , T.Z / T.W, 1];*

*Вернуть P;*

*Конец*

Расчет координат массива T2D\_area производится следующим образом:

*Для каждого номера* ***i*** *из массива T4D сделать:*

*Начало*

*Point4D P;*

*Если Позиция\_Переключателя\_Проецироания = true тогда:*

*Начало*

*P = РасчОртПроицир(T4D[i].X, T4D[i].Y, T4D[i].Z);*

*Конец*

*Иначе*

*Начало*

*P = РасчЦентрПроицир(T4D[i].X, T4D[i].Y, T4D[i].Z);*

*Проверка\_Выхода\_За\_Границы\_Экрана(P);*

*Конец*

*КонецЕсли*

*T2D\_area[i].X = P.X;*

*T2D\_area[i].Y = P.Y;*

*Конец*

*КонецДля*

1. ***Для комплексного чертежа***

Рассматривалось в лабораторной работе №1.

**6.2.3 Прорисовка.**

1. ***Для аксонометрического чертежа***

*//Отрисовка осей и их подпись*

DrawLine(*T2D\_area[2], T2D\_area[0]*);

DrawLine(*T2D\_area[4], T2D\_area[0]*);

DrawLine(*T2D\_area[6], T2D\_area[7]*);

DrawString(“X”, *T2D\_area[1]*);

DrawString(“Y”, *T2D\_area[2]*);

DrawString(“Z”, *T2D\_area[3]*);

*//Отрисовка точек и их подпись*

DrawEllipse(*T2D\_area[5]*,r, r);

DrawEllipse(*T2D\_area[6]*,r, r);

DrawEllipse(*T2D\_area[7]*, r, r);

DrawEllipse(*T2D\_area[8]*, r, r);

DrawEllipse(*T2D\_area[9]*, r, r);

DrawEllipse(*T2D\_area[10]*, r, r);

DrawEllipse(*T2D\_area[11]*,r, r);

DrawString(“T”, *T2D\_area[5]*);

DrawString(“T1”, *T2D\_area[6]*);

DrawString(“T2”, *T2D\_area[7]*);

DrawString(“T3”, *T2D\_area[8]*);

DrawString(“Tx”, *T2D\_area[9]*);

DrawString(“Ty”, *T2D\_area[10]*);

DrawString(“Tz”, *T2D\_area[11]*);

*//Отрисовка линий связи*

DrawLine(*T2D\_area[5], T2D\_area[6]*);

DrawLine(*T2D\_area[5], T2D\_area[7]*);

DrawLine(*T2D\_area[5], T2D\_area[8]*);

DrawLine(*T2D\_area[6], T2D\_area[9]*);

DrawLine(*T2D\_area[6], T2D\_area[11]*);

DrawLine(*T2D\_area[7], T2D\_area[9]*);

DrawLine(*T2D\_area[7], T2D\_area[11]*);

DrawLine(*T2D\_area[8], T2D\_area[10]*);

DrawLine(*T2D\_area[8], T2D\_area[11]*);

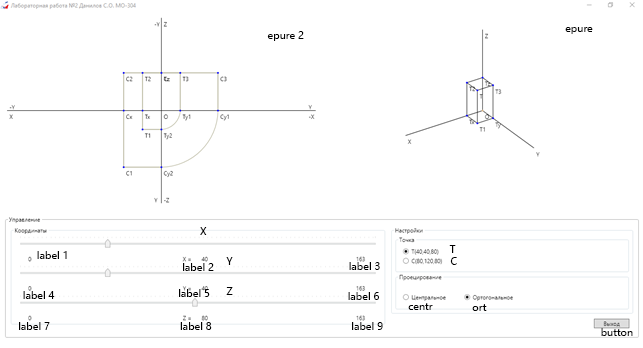
1. ***Для комплексного чертежа***

Рассматривалось в лабораторной работе №1.

# 7. Руководство программиста.

Программа написана на языке С# с использованием технологии WPF. Для работы с матрицами использовалась библиотека MathNet.

Изображения выводятся на пользовательских элементах управления. Ввод данных осуществляется путем взаимодействия с ползунковыми переключателями Slider. Выбор типа проецирования (ортогональное и центральное) и точки, чьи координаты будут меняться при изменении положения Slider, выбирается с помощью элементов RadioButton. Для выхода из программы используется кнопка Button.



Элементы управления, отображающие чертежи:

epure — для отображения пространственного чертежа.

Canvas – на нем непосредственно отображаются элементы, стандартный элемент управления

Line – класс, описывающий с линия

PointT – точка с текстом

Ellipse – стандартный эллипс, примитив класса Shapes

TextBlock – блок текста

Polyline – ломанная, примитив класса Shapes

epure2 — для отображения комплексного чертежа.

Canvas – на нем непосредственно отображаются элементы, стандартный элемент управления

LineText – класс, описывающий линию с четырьмя строками, расположенными вблизи точек начала и конца линии

Polyline – ломанная, примитив класса Shapes

Элементы Slider:

X— изменяет координату х.

Y — изменяет координату y.

Z — изменяет координату z.

Элементы Button.

Button — осуществляет выход из программы.

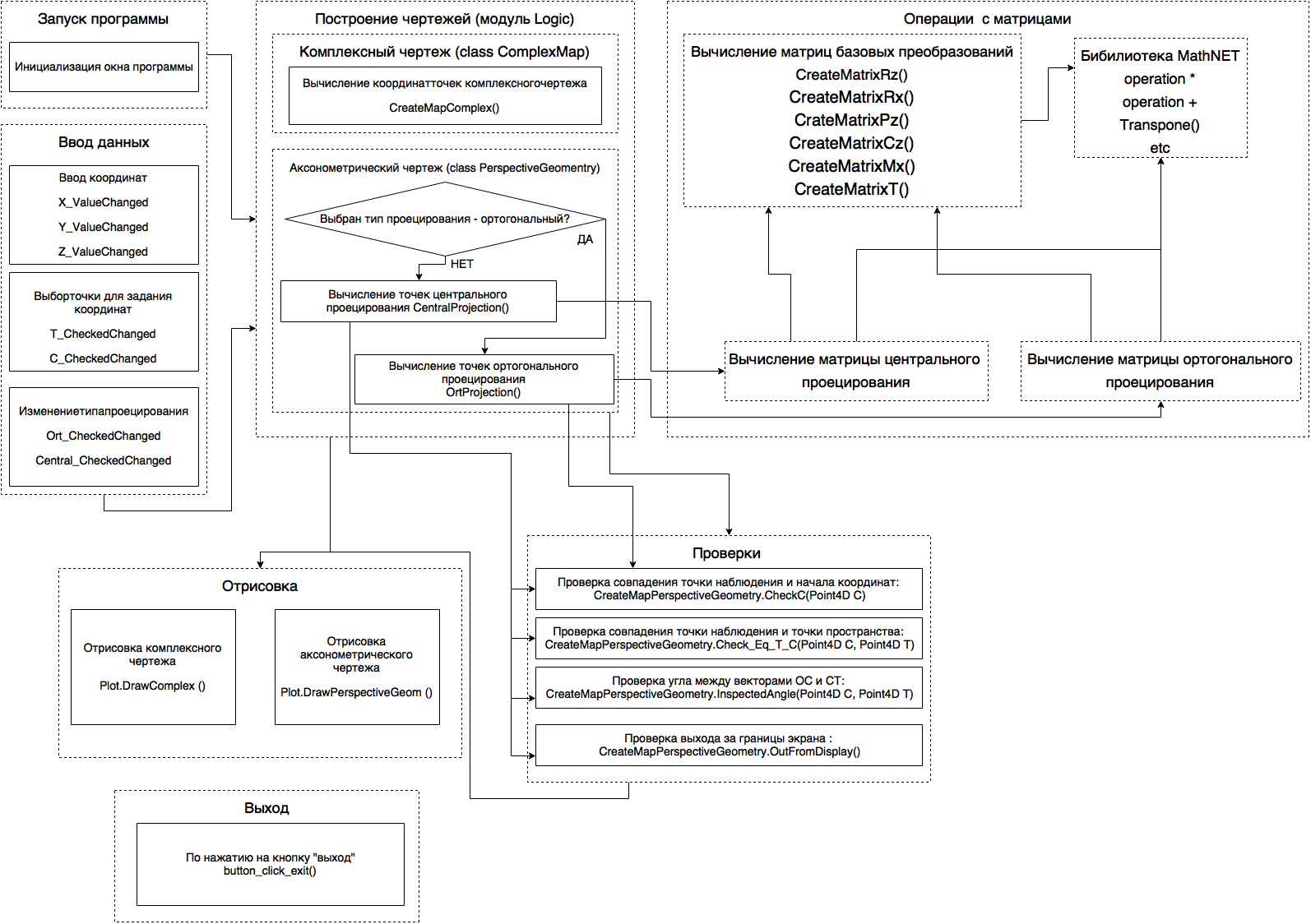
Элементы RadioButton:

Centr – выбор центрального проецирования для отображения

Ort- выбор ортогонального проецирования для отображения

T – выбор координат точки Т для изменения

С – изменение координат точки С



Point4D T4D[11]-массив точек в однородных координатах размерности 11, где каждая точка имеет тип Point4D.

Point2D T2D\_area[11]-массив двумерных точек для аксонометрического чертежа размерности 11, где каждая точка имеет тип Point2D.

Point2D T2D\_complex[12]- массив двумерных точек для комплексного чертежа размерности 12, где каждая точка имеет тип Point2D.

## 8. Описание функциональной схемы.

При запуске программы происходит начальная инициализация программы и загрузка необходимых компонентов интерфейса. Как только инициализация главного окна завершена, происходит инициализация комплексного и аксонометрического чертежей, а так же заполнение массивов T4D точками типа Point4D, массивы T2D\_area и T2D\_complex инициализируются нулевыми значениями. Затем методе MainWindow() главного окна осуществляется расчёт координат для начальных значений входных данных.

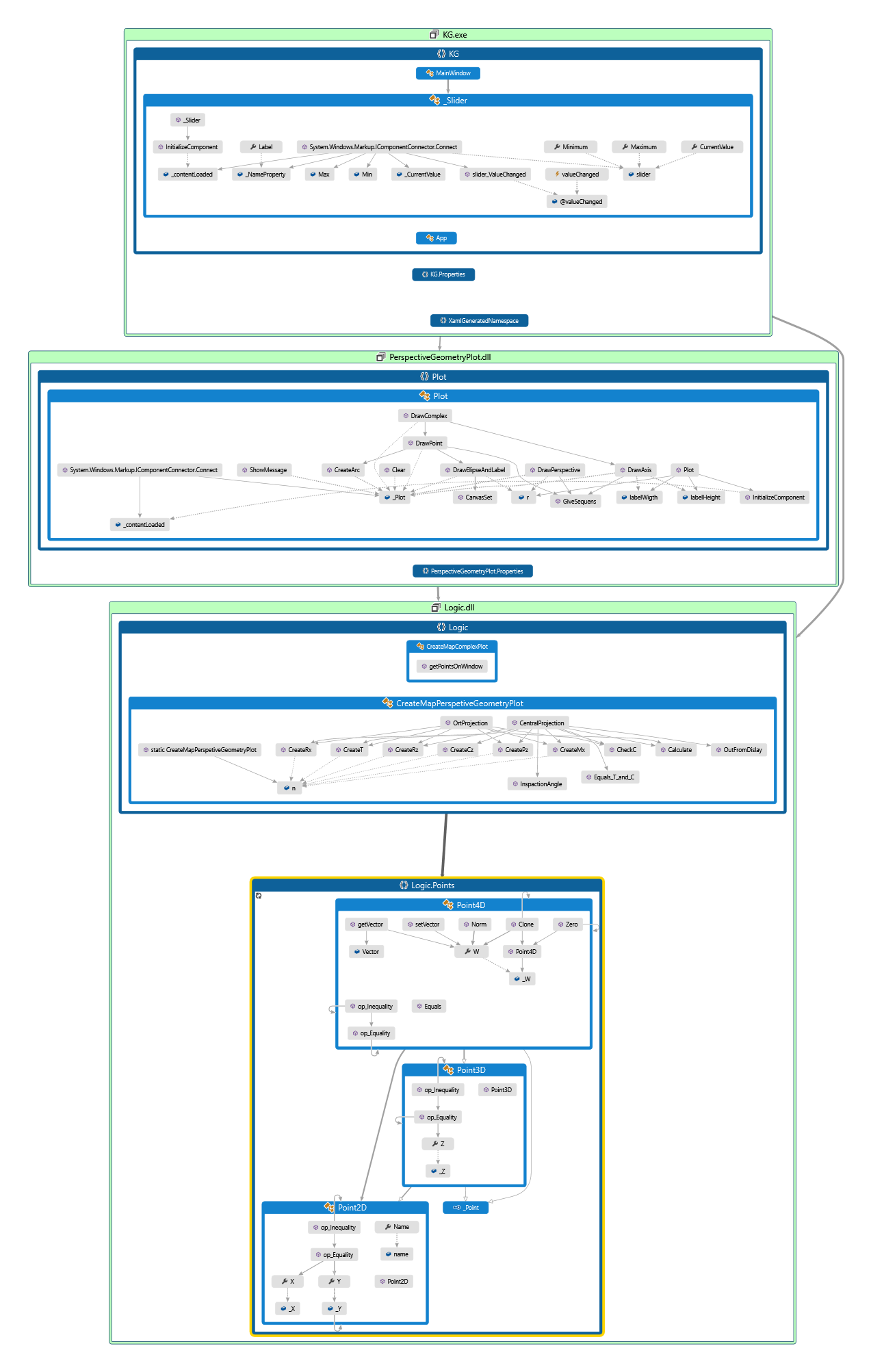
При изменении координат точки T или камеры C (X\_ValueChanged, Y\_ValueChanged, Z\_ValueChanged) и при смене типа проецирования (Orth\_CheckedChanged, Central\_CheckedChanged) происходит вызов метода Inptut(), в котором вызываются методы для перерасчета координат и перерисовки чертежей. Все расчеты производятся в библиотеке Logic, статическими классами: PerspectiveGeometry, ComplexMap. Расчет координат для комплексного чертежа осуществляется в методе ComplexMap.CreateMapComplex(). Расчет координат точек центрального проецирования для аксонометрического чертежа производится в методе PerspectiveGeometry.CentralProjection(), а для ортогонального проецирования используется метод PerspectiveGeometry.OrtProjection(). Расчет матриц базового преобразования происходит методами: CreateMatrixRz(), CreateMatrixRx(), CrateMatrixPz(), CreateMatrixMx(), CreateMatrixT(), которые вызываются при вызове любой функции расчета координат точек проецирования (PerspectiveGeometry.CentralProjection(), PerspectiveGeometry.OrtProjection()), но при вызове функции PerspectiveGeometry.CentralProjection() к этому списку добавляется еще вызов метода для расчета матрицы перспективного преобразования: CreateMatrixCz().

Для работы с матрицами была использована сторонняя библиотека MathNet, которая имеет классы, у которых определены основные операции над матрицами: Matrix, Vector, etc. С помощью метода getVector класса Point4D можно получить объект класса Matrix (матрица, состоящая из 1 строки и числа столбцов равному числу координат) из библиотеки MathNET, затем с его помощью рассчитываются координаты для всех точек из T2D\_area, используя массив T4D. Проверка выхода за границы экрана (CreateMapPerspectiveGeometry.OutFromDisplay()) применяется к координатам точки после умножения на матрицу сложного преобразования, все остальные производятся с исходными координатами точек (проверка совпадения точки наблюдения и начала координат: CreateMapPerspectiveGeometry.CheckC(Point4D C), проверка совпадения точки наблюдения и точки пространства: CreateMapPerspectiveGeometry.Check\_Eq\_T\_C(Point4D C, Point4D T),

проверка угла между векторами ОС и СТ: CreateMapPerspectiveGeometry.InspectedAngle(Point4D C, Point4D T)) к координатам точек типа Point4D.

После того, как координаты проекций точек рассчитаны и находя в массиве T2D\_area, а для комплексного чертежа в массиве T2D\_complex, вызываются методы для построения комплексного и аксонометрического чертежей: DrawComplex() и DrawPersptivegeom().

При нажатии на кнопку «Выход» происходит завершение работы программы.

**Структура решения**

CreateMapComplex()

Point4D-класс, описывающая трехмерную точку в однородных координатах. В нем имеются четыре атрибута: X-координата по оси Ox, Y-координата по оси Oy, Z по оси Oz и W однородная координата.

Методы:

Clone() – возвращает новый объект этого же класса с теми же атрибутами, что и у вызываемого объекта

Zero()- возвращает нулевой новый объект этого же класса со всеми атрибутами равными нулю

Norm() – совершает нормирование координат точки.

getVector() – возвращает новый объект класса Matrix из библиотеки MathNET, который представляет координаты (атрибуты объекта класса) в виде матрицы, состоящей из 1 строки и 4 столбов.

setVector(v) – соответственно, преобразует матрицу из 1 строки и 4 столбцов

Point2D-класс, описывающая двумерную точку. В нем имеются только два атрибута: X-координата по оси Ox, Y-координата по оси Oy.

Point3D-класс, описывающая трехмерную точку. В нем имеются три атрибута: X-координата по оси Ox, Y-координата по оси Oy, Z – координата по оси Z.

Point – пустой интерфейс, введенный во имя использования прелестей ООП

Logic – библиотека реализующая расчет координат проекций для вывода аксонометрического чертежа и расчет координат для вывода комплексного чертежа

Class ComplexMap – статический класс, имеющий методы для расчета экранных координат комплексного чертежа для переданных точек:

CreateMapComplex() – метод, генерирующий массив двумерных точек (Point2D), по заданному списку трехмерных точек(Point3D).

Class PerspectiveGeometry – статический класс, имеющий методы для расчета координат проекций центрального и ортогонального проецирования аксонометрического чертежа:

CentralProjection() – метод, генерирующий массив двумерных точек(Point2D), представляющих собой центральное проецирование,по входному списку четырехмерных точек (Point4D)

OrtProjection() – метод, генерирующий массив двумерных точек(Point2D), представляющих собой ортогональное проецирование, по входному

Методы

CreateRz() – возвращает матрицу поворота относительно оси Z

CreateRx() – возвращает матрицу поворота относительно оси X

CreatePz() – возвращает матрицу проецирования на плоскость YOX

CreateMx() – возвращает матрицу отражения относительно оси X

CreateCz() – возвращает матрицу перспективного преобразования

Calculate() – возвращает матрицу сложного преобразования ортогонального проецирования

Методы проверок:

CheckC - проверка совпадения точки С и начала координат О

EqualsT\_C – проверка совпадения точки C и точки пространства Т

OutFromDisplay – проверка выхода точки за границы отрисовываемой области

InspactionAngle – проверка cos’a угла между векторами CO и CT

Константы:

n – порядок матриц простого преобразования

PerspectiveGeometryPlot – библиотека, реализующая компонент, унаследованный от стандартного класса UserControl, для отображения чертежей на произвольной экранной формы.

Методы:

DrawComplex(Point2D points[]) – по переданному массиву точек рисует комплексный чертеж

DrawPerspective(Point2D points[]) – по переданному массиву точек рисует аксонометрический чертеж

ShowMessage(string Message) – выводит сообщение, переданное в функцию, в центре области для рисования

GiveSequence(string Seq, Point2D points[]) – по переданной строке и массиву точек, строится последовательность, представляющая собой пусть отрисовки кривой линии, для отображения «костяка» чертежа (проецирующие линии, оси координат)

DrawAxis() – отрисовывает оси координат и подписывает их для комплексного чертежа

CreateArc(Point4D yx) – отрисовывает дугу на комплексном чертеже, используя координаты переданной точки.

Константы:

r - радиус отрисовываемой точки на чертежах

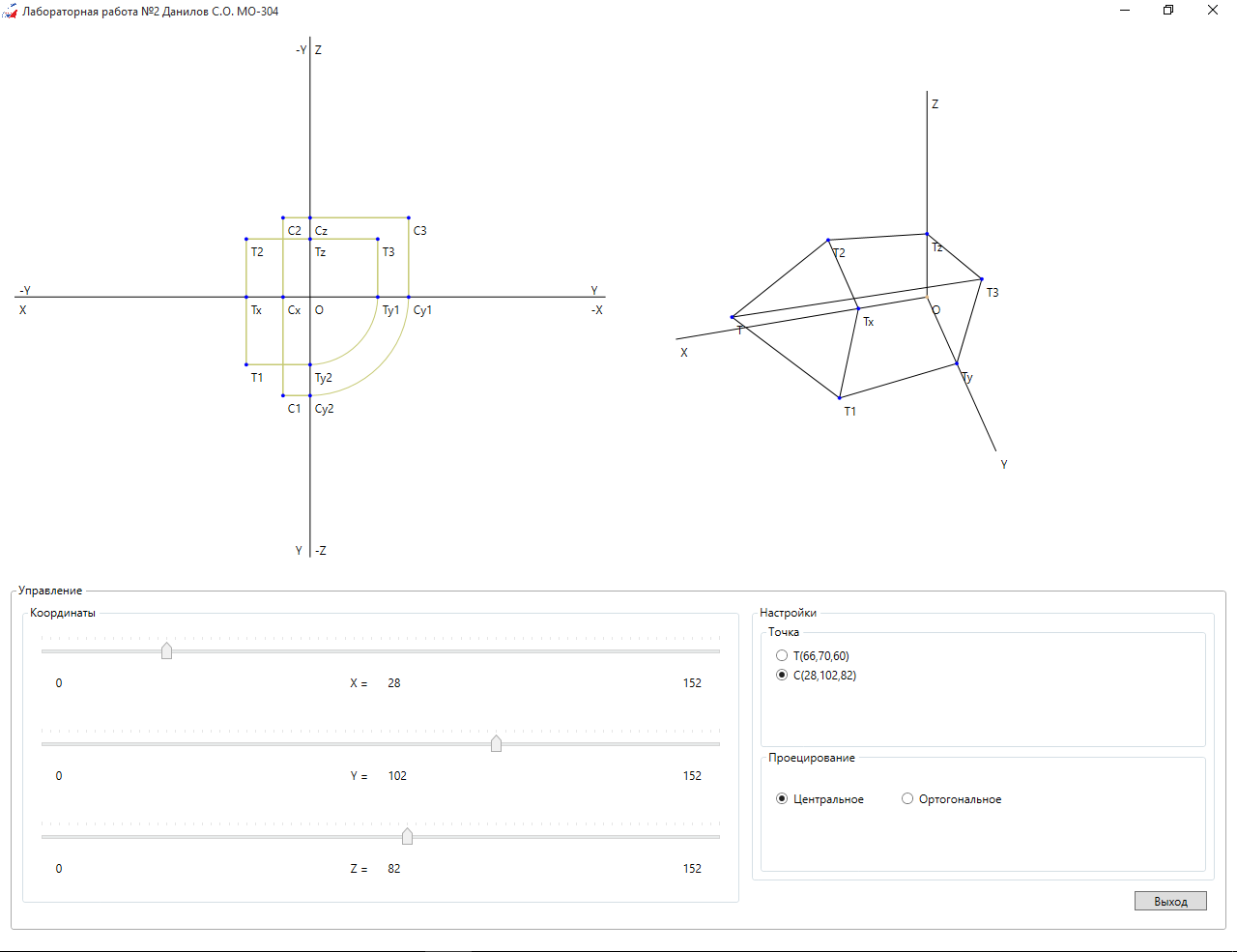
labelHeigth – высота подписи точки

labelWight – ширина подписи точки

Matrix- класс, из используемой из библиотеки MathNET, представляющая собой матрицу и поддерживающая все операции между матрицами.

Переменные semiaxis\_w, semiaxis\_h-значение полуосей, равные половине ширины и длины экрана соответственно.

# 9. Руководство пользователя.



Программа выводит на экран комплексный и аксонометрический чертежи проекций точки Т с возможностью выбора типа проецирования между центральным и ортогональным.

Три ползунковых переключателя отвечают за изменение трехмерных координат точки Т и камеры С, есть возможность выбора точки, для которой в данный момент изменяются координаты.

Рядом с ползунками отображаются минимальное значение соответствующей координаты, максимальное и текущие. Так же информация о координатах точки Т и точки С отображается в отдельной области экрана.

При нажатии на кнопку «Выход» программа завершается.

# 10. Результаты

В ходе выполнения лабораторной работы была написана программа, позволяющая динамически менять координаты точки Т, камеры С, тип проецирования (центральный и ортогональный), и отображающая соответствующие аксонометрический и комплексный чертежи.