Уфимский Государственный Авиационный Технический Университет

Кафедра информатики

Пояснительная записка к лабораторной работе № 1

“Визуализация 3D-точки”

по дисциплине

“Математические и алгоритмические основы компьютерной графики”

Выполнил:

студент группы …

……..

Проверил:

……..

Уфа 2020

Краткая теоретическая справка

Ортогональная система трех плоскостей проекций

В основу построения любого изображения положена операция проецирования.

Сущность метода ортогонального проецирования заключается в том, что предмет проецируется на две взаимно перпендикулярные плоскости лучами, ортогональными (перпендикулярными) к этим осям.

Одну из плоскостей проекций $П_1$ располагают горизонтально, а вторую $П_2$ - вертикально. Плоскость $П_1$ называют горизонтальной плоскостью проекций, $П_2$ - фронтальной. Плоскости $П_1$ и $П_2$ бесконечны и непрозрачны.

При построении проекций необходимо помнить, что проекцией точки на плоскость называется основание перпендикуляра, опущенного из данной точки на эту плоскость. Проекцию точки А на горизонтальную плоскость называют горизонтальной проекцией и обозначают $А_1$, проекцию точки А на фронтальную плоскость - фронтальной проекцией и обозначают $А_2$. Каждая из них является основанием перпендикуляра, опущенного из данной точки А соответственно на плоскости $П_1$ и $П_2$. Две проекции точки определяют ее положение в пространстве. Так как каждая фигура или тело представляет собой совокупность точек, то можно утверждать, что две ортогональные проекции предмета (при наличии буквенных обозначений) вполне определяют его форму.

Однако в практике изображения строительных конструкций, машин и различных инженерных сооружений возникает необходимость в создании дополнительных проекций. Чтобы сделать проекционный чертеж более ясным и удобочитаемым, используют третью плоскость, перпендикулярную $П_1$ и $П_2$. Эта плоскость обозначается буквой $П_3$ и называется профильной.

Проекции точек на плоскость $П_3$ называются профильными и обозначают $А_3$.

Плоскости проекций, попарно пересекаясь, определяют три оси: Ox, Oy и Oz, которые можно рассматривать как систему прямоугольных декартовых координат в пространстве с началом в точке О. Система знаков, соответствующая “правой системе” координат, показана на рисунке 1.

Три плоскости проекций делят пространство на восемь трехгранных углов - это так называемые **октанты**. Нумерация октантов дана на рисунке 1

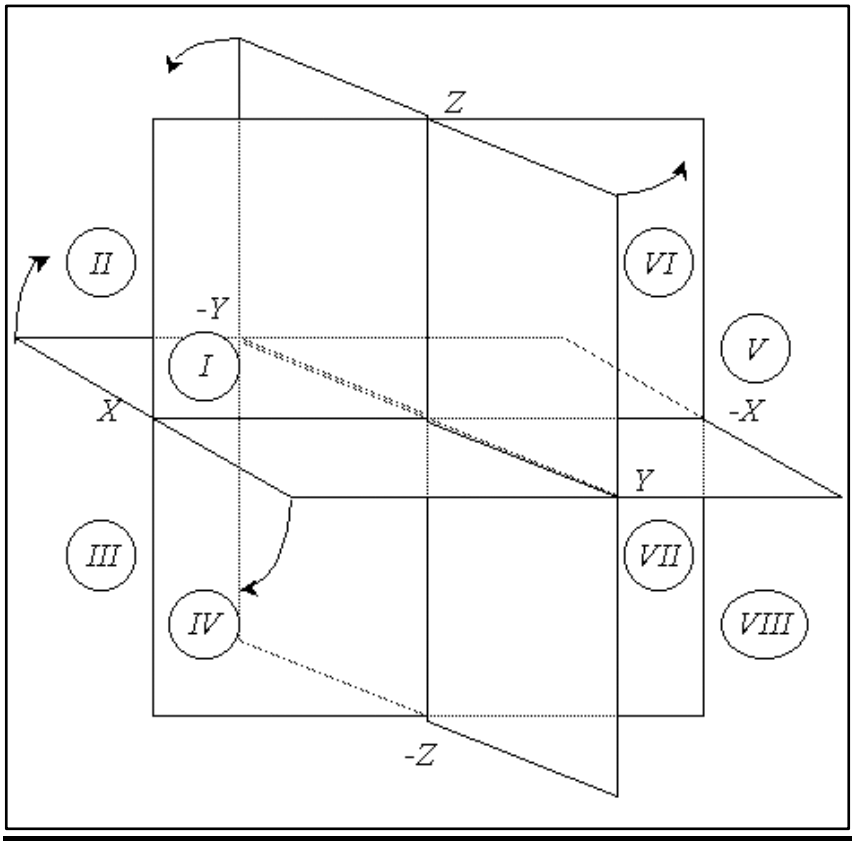

Рис. 1 Ортогональная система плоскостей

Рассматривая ортогональные проекции, предполагают, что наблюдатель находится в первом октанте. Проекционный чертеж, на котором плоскости проекций со всем тем, что на них изображено, совмещены определенным образом одна с другой, называется эпюром. Для получения эпюра плоскости $П_1$ и $П_3$ вращают как показано на рисунке 1, до совмещения с плоскостью $П_2$. В результате вращения передняя полуплоскость $П_1$ оказывается совмещенной с нижней полуплоскостью $П_2$, а задняя полуплоскость $П_1$ - с верхней полуплоскостью $П_2$.

Окончательный вид всех совмещенных плоскостей проекций дан на рисунке 2.

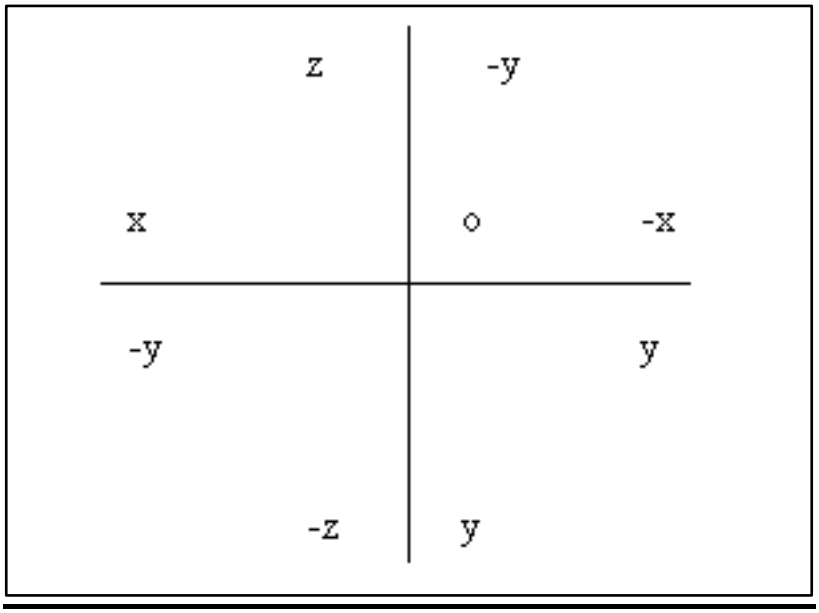

Рис. 2 Вид совмещенных плоскостей

Три координаты и три проекции точки

Координатами *называют числа, которые ставят в соответствие точке для определения ее положения в пространстве или на поверхности.*

В трехмерном пространстве положение точки устанавливают с помощью прямоугольных декартовых координат x, y, z.. Абсцисса x определяет расстояние от данной точки до плоскости $П_3$, ордината y - до плоскости $П_2$ и аппликата z- до плоскости $П_1$. Приняв для отсчета координат точки систему, показанную на рисунке 2 , составим таблицу знаков координат во всех восьми октантах.

Таблица 1. Октанты и знаки октант

Октант	Знаки координат		
	x	y	z
I	+	+	+
II	+	-	+
III	+	-	-
IV	+	+	-
V	-	+	+

VI	-	-	+
VII	-	-	-
VIII	-	+	-

Какая-либо точка пространства А, заданная координатами, будет обозначаться так: А (x, y, z). Построение изображения самой точки и ее проекций на пространственной модели рекомендуется осуществлять с помощью координатного прямоугольного параллелепипеда. Прежде всего на осях координат от точки О откладывают отрезки, соответственно равные единицам длины. На этих отрезках (OA_X, OA_Y, OA_Z), как на ребрах, строят прямоугольный параллелепипед. Вершина его, противоположная началу координат, и будет определять заданную точку А.

Построение параллелепипеда позволяет определить не только точку А, но и все три ее ортогональные проекции.

Лучами, проецирующими точку на плоскости , являются те три ребра параллелепипеда, которые пересекаются в точке А.

Каждая из ортогональных проекций точки А, будучи расположенной на плоскости, определяется только двумя координатами. Так, горизонтальная проекция A_1 определяется координатами x и y, фронтальная проекция A_2 - координатами x и z, профильная проекция A_3 - координатами y и z. Но две любые проекции определяются тремя координатами. Вот почему задание точки двумя проекциями равносильно заданию точки тремя координатами.

На эпюре (рисунок 3), где все плоскости проекций совмещены, проекции A_1 и A_2 окажутся на одном перпендикуляре к оси OX, а проекции A_2 и A_3 - на одном перпендикуляре к оси OZ.

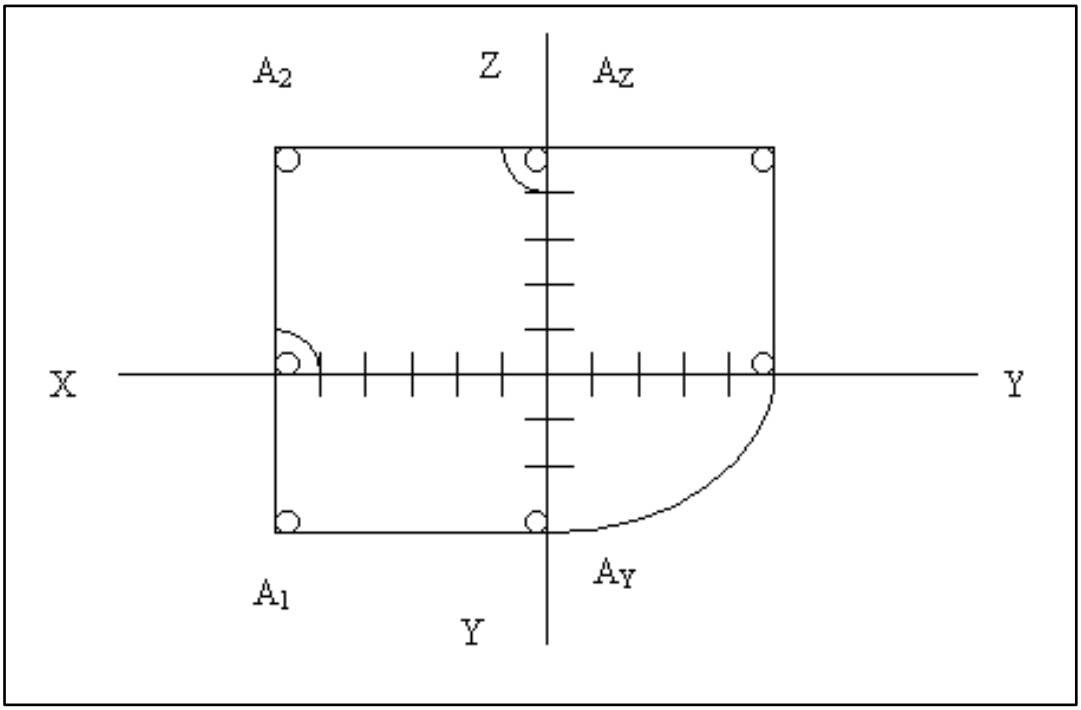

Рис. 3 Проекции точки (1-й октант) на комплексном чертеже

Точки Tx, Ty, Tz, проекции точки T: T1, T2, T3 и углы α, β, γ между горизонтальным вектором , направленным влево, и положительными направлениями осей координат Ox, Oy, Oz соответственно.

Точки **Tx, Ty, Tz** –это координаты точки T, отложенные на осях координат Ox, Oy, Oz.**T1, T2, T3** – проекции точки Tна координатные плоскости$П_1$, $П_2$ и $П_3$.

Так как в данной лабораторной работе пользователь будет иметь возможность изменять положение координатных осей (Ox, Oy, Oz), тодля предоставления данной возможности будем использовать углы **α, β, γ**между горизонтальным вектором , направленным влево, и положительными направлениями осей координат Ox, Oy, Oz соответственно.

На рисунке 4 показано расположение точек Tx, Ty, Tz; проекций точки T: T1, T2, T3на координатные плоскости$П_1$, $П_2$ и $П_3$и углов α, β, γ на пространственном чертеже с используемыми обозначениями, которые нам предстоит отобразить на экране.

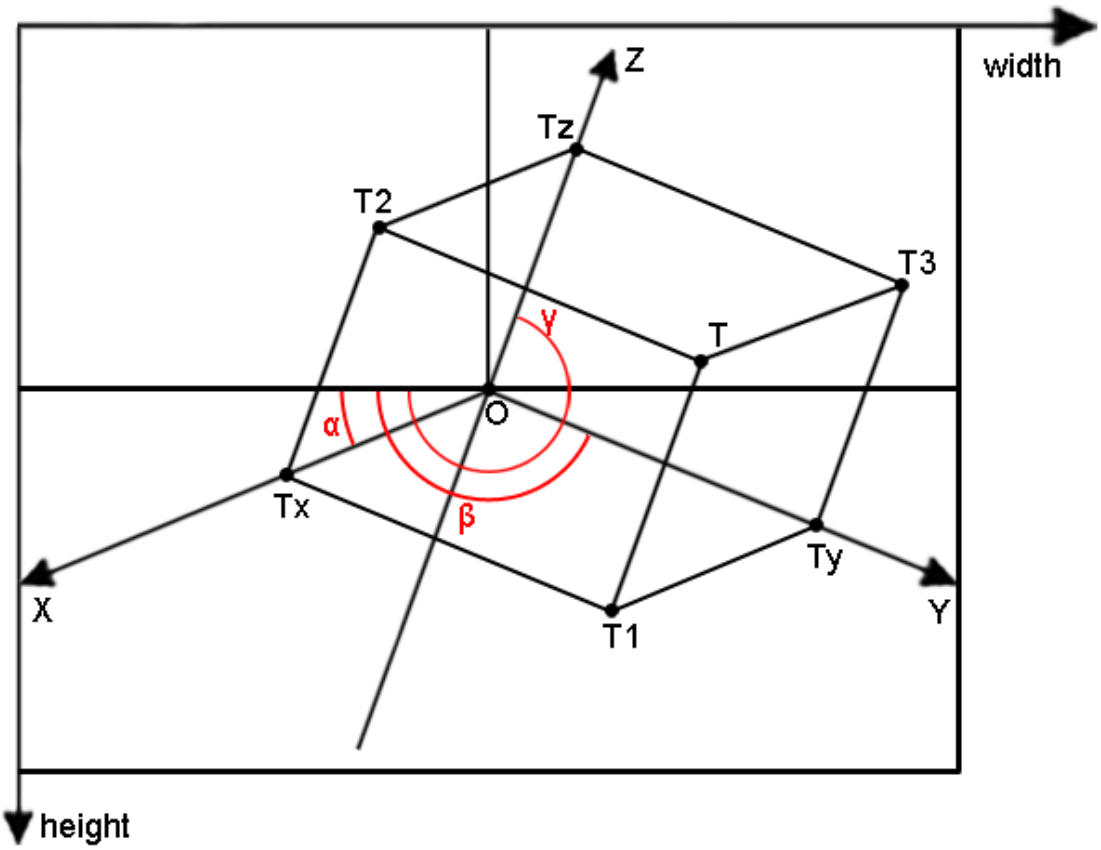

Рис. 4 Пространственный чертеж

Постановка задачи

Содержательная постановка задачи

Задание на лабораторную работу № 1:

Визуализация 3D-точки на пространственном и комплексном чертежах.

Необходимо написать программу, удовлетворяющую следующим требованиям:

Содержание экрана:

- пространственный макет с изображением точки Т, ее проекций T_1, T_2, T_3 и линий связи;
- комплексный чертеж с изображением проекций T_1, T_2, T_3 и линий связи;
- ползунковые переключатели для интерактивного изменения координат (X, Y, Z) точки Т.
- ползунковые переключатели для интерактивного изменения углов α, β, γ на пространственном макете, где α, β, γ – углы между горизонтальным вектором, направленным влево, и положительными направлениями осейOx, Oy, Oz соответственно.

Динамика:

- при изменении координат точки Т должны изменяться соответствующие проекции, линии связи и сама точка Т на пространственном и комплексном чертежах.
- при изменении углов, задающих оси, должны изменяться соответствующие проекции, линии связи и сама точка Ттолько на пространственном чертеже.

Входные данные:

- Пустой экран.

Управляющие параметры:

- Координаты изображаемой точки.
- Углы, задающие оси координат на пространственном макете.

Выходные данные:

- Изображение 3D-точки на пространственном чертеже.
- Изображение точки на комплексном чертеже.

Формальная постановка задачи

Формализуем поставленную задачупри помощи схемы (в нотации IDEF0) , рис. 5.

Рис.5 Схема задачи 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 информации о координатах точки T и углах между горизонтальным вектором и осями координат (для пространственного макета).

2. **Обработка**
   Операция проецирования, осуществляемая двумя способами, т.е. преобразование совокупности трехмерных точек в два набора двумерных точек:
   1) Для пространственного чертежа;
   2) Для комплексного чертежа.

3. **Вывод**
   Визуализация изображения:
   1) Пространственного чертежа;
   2) Комплексного чертежа.

В итоге мы разделили нашу задачу на 5 подзадач (“листьев”), которые подробно рассмотрим на следующем уровне детализации.

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

1. Ввод информации о координатах точки и углах между горизонтальным вектором и осями координат

Необходимо ввести следующие величины:

- координаты точки в пространстве (X, Y, Z);
- углы между горизонтальным вектором, направленным влево, и положительными направлениями осейOx, Oy, Ozсоответственно (α, β, γ);

В данном случае наиболее удобным для ввода является использование ползунковых переключателей, в силу того, что величины, которые необходимо ввести, в общем случае, являются вещественными и непрерывными.
Ползунковые переключатели присутствуют в стандартных средствах разработки. Во многих средах разработки программного обеспечения для операционной системы Windows (например, VisualStudio) компонент ползунковых переключателей представлен классом “Trackbar”.

Обобщенный алгоритм:

1. Считываем значения координат точки T (X, Y, Z) и углов между горизонтальным вектором, направленным влево, и положительными направлениями осейOx, Oy, Ozc соответствующих ползунковых переключателей;
2. Записываем эти значения в соответствующие переменные.

2. *Преобразование совокупности трехмерных точек в два набора двумерных точек*

Для визуализации объектов 3D-пространства на 2D экране, необходимо провести преобразование трехмерных координат в двумерные координаты.

2.1.Преобразование координат точки для пространственного чертежа.

Задание:

1) получить экранные 2D-координаты 3D-точки в пространстве, ее проекций и точек, задающих оси координат и линии связи.

Дано:

1) 3D-координаты точки Тв пространстве: x, y, z.
2) Углы между горизонтальным вектором, направленным влево, и положительными направлениями осей координат Ox, Oy, Oz: α, β, γ.
3) Ширина и высота области рисования: width, height.

Найти:

1) Экранные 2D-координаты:
    a. Точек начала и конца осей Ox, Oy, Oz;
    b. Точки пересечения осей координат (точка O, начало координат)
    c. Точки T;
    d. Проекций точки Тна оси Ox, Oy, Oz;
    e. Проекций точки Тна плоскости Ox, Oy, Oz;

Решение:

Получим формулы, по которым мы будем осуществлять преобразование точек из 3Dв 2D.

Рассмотрим рисунок 6

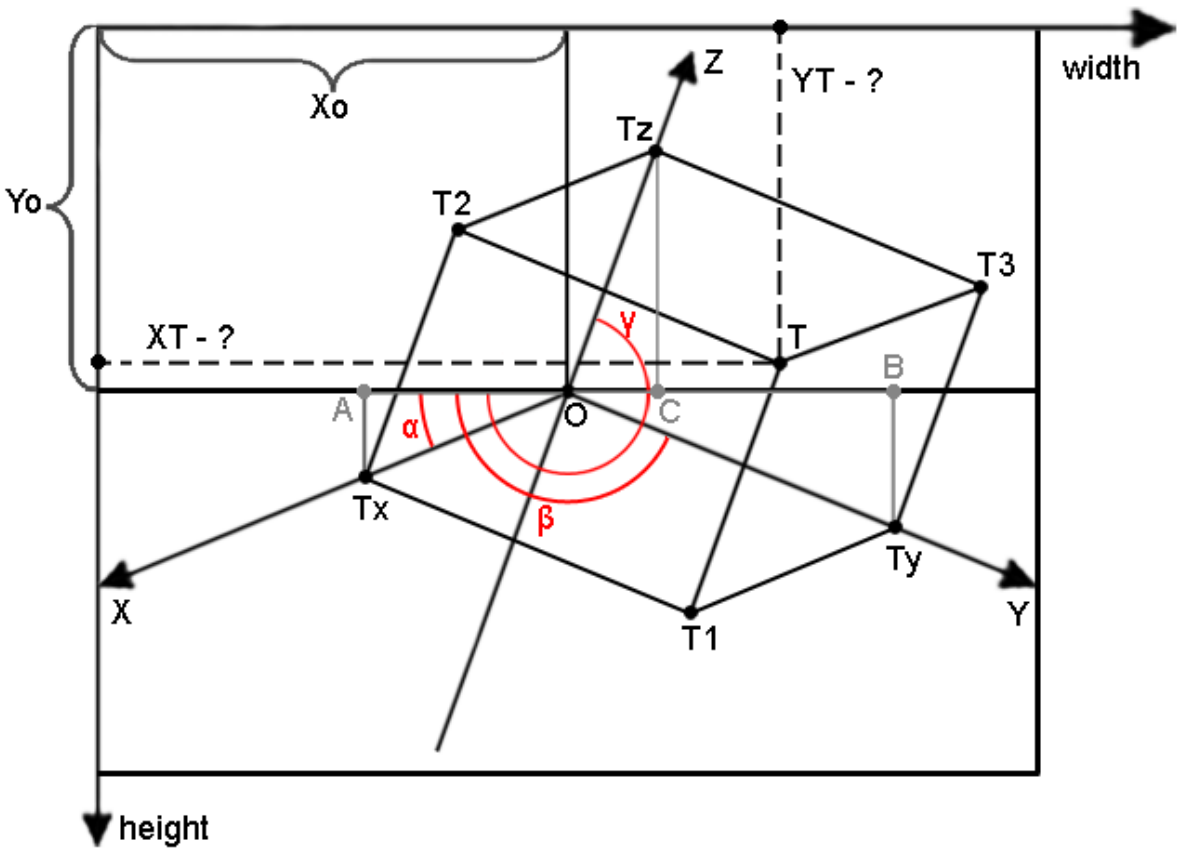

Рис. 6

Пусть:

a1 = OA;	b1 = OB;	c1 = OC;
a2 = ATx;	b2 = BTy;	c2 = CTz;
x = OTx;	y = OTy;	z = OTz;

Xo = width/2;
Yo = height/2;

Тогда:

XT = Xo – a1 + b1 + c1;	YT = Yo – c2 + a2 + b2
a1 = x * cos(α);	a2 = x * sin(α);
b1 = y * cos(π – β) = -y * cos(β);	b2 = y * sin(π – β) = y * cos(β);
c1 = z * cos(γ – π) = -z * cos(γ);	c2 = z * sin(γ – π) = -z * sin(γ);

В итоге, при подстановке получим следующую систему формул:

$$\begin{cases} XT = Xo - x * \cos(\alpha) - y * \cos(\beta) - z * \cos(\gamma); \textbf{(1)} \\ YT = Yo + x * \sin(\alpha) + y * \cos(\beta) + z * \sin(\gamma); \end{cases}$$

С помощью формул **(1)**можно преобразовать трехмерные координаты точек в двумерные для пространственного макета одинаковым образом. Таким образом, наиболее целесообразным будет хранение всех 14-ти точек в массиве 3D-точек, а преобразование проводить в цикле, используя формулы **(1)** для каждой точки одинаково, после чего записывать полученные результаты уже в массив из 14-ти 2D-точек.

Обобщенный алгоритм:

1. Каждую из 14-ти 3D-точек пространственного макета преобразовать в двумерную точку «пространственного» чертежа по системе**(1)**.

2.2. Преобразование координат точки для комплексного чертежа.

Задание:

1) получить экранные 2D-координаты проекций точки T на координатные оси и плоскости и точек, задающих оси.

Дано:

1) 3D-координаты точки Tв пространстве: x, y, z.
2) Ширина и высота области рисования: width, height.

Найти:

1) Экранные 2D-координаты:
    a. Точек начала и конца осей Ox, Oy, Oz;
    b. Точки пересечения осей координат (точка O, начало координат);
    c. Проекций точки Tна координатные оси Ox, Oy, Oz;
    d. Проекций точки Tна координатные плоскостиOx, Oy, Oz;

Решение:

Рассмотрим рисунок 7

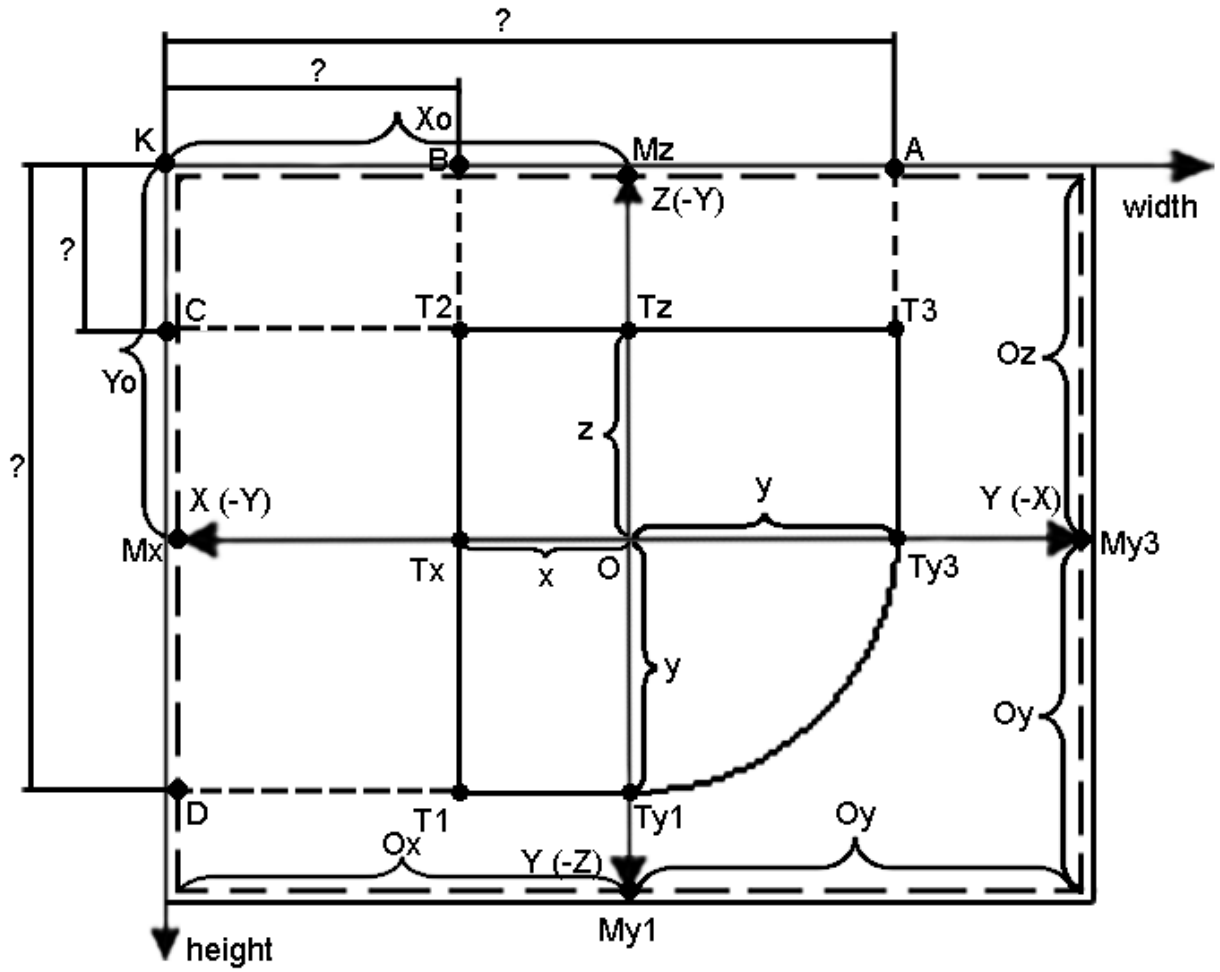

Рис.7

Пусть

Xo = width/2;

Yo = height/2;

AK = y + x +Xo – x = Xo + y;

BK = Xo – x;

CK = Yo – z;

DK = y + z + Yo – z = Yo + y;

a. Координаты точек начала и конца осей:
   Mx = (Xo – Ox, Yo);
   Mz = (Xo ,Yo – Oz);
   My1 = (Xo, Yo + Oy);
   My3 = (Xo + Oy, Yo);

b. Координаты точки O (начала координат):
   O = (Xo, Yo);

c. Координаты проекций точки Tна координатные оси:
   Tx = (BK, Yo) = (Xo – x, Yo);
   Tz = (Xo, CK) = (Xo, Yo – z);
   Ty1 = (Xo, DK) = (Xo, Yo + y);
   Ty3 = (AK, Yo) = (Xo + y, Yo);

d. Координаты проекций точки Tна координатные плоскости:
   T1 = (BK, DK) = (Xo - x, Yo + y);
   T2 = (BK, CK) = (Xo - x, Yo – z);
   T3 = (AK, CK) = (Xo + y, Yo – z);

Как можно заметить, достаточно выделить лишь три формулы, по которым будут производиться преобразования для координат всех 12-ти точек.

Пусть dx, dy, dz–координаты трехмерной точки, для которой производится преобразование в 2D. Тогда:

- Для координат точек Mx, My1, Tx,Ty1,T1:
  (Xo – dx, Yo + dy)**(2)**

- Для координат точек Mz, Tz, T2:
  (Xo – dx, Yo – dz) **(3)**

- Для координат точек My3, Ty3, T3:
  (Xo + dy, Yo – dz) **(4)**

Обобщенный алгоритм:

1. Для каждой из 12-ти 2D-точек комплексного чертежа преобразовать соответствующую точку из 3D точек пространственного макета по соответствующей каждой паре точек формуле **(2), (3) или (4)**.

3. Визуализация изображений пространственного и комплексного чертежей.

Для визуализации изображения необходимо вывести на экран следующие компоненты:

1. Координатные оси
    a. Три оси для пространственного чертежа;
    b. Две оси для комплексного чертежа.

2. Точку Т на пространственном чертеже

3. Проекции точки Т на координатные оси
    a. Для пространственного чертежа;
    b. Для комплексного чертежа.

4. Проекции точки Т на координатные плоскости
    a. Для пространственного чертежа;
    b. Для комплексного чертежа.

5. Линии связи и текст (названия точек)

Каждый из чертежей будет отрисовываться в соответствующем графическом окне, которое во многих средах разработки программного обеспечения для операционной системы Windows (например, VisualStudio) представлено классом PictureBox.

Для отрисовки что пространственного, что комплексного чертежей, нам необходимы одни и те же примитивы(за исключением дуги)

Для вывода данных компонентов воспользуемся отрисовкой:

- Линий (для осей координат и линий связи);
- Окружностей (для точки и ее проекций);
- Дуги (для соединения Ту1, Ту3 на комплексном чертеже).
- Текста

Все необходимые графические примитивы присутствуют в интерфейсе графических устройств (GraphicsDeviceInterface, GDI), который используется для отрисовки 2D-графики в операционной системеWindows.

Для отрисовки пространственного и комплексного чертежей будем использовать функции:

- DrawLine(), отрисовка линии между двумя точками;
- FillEllipse(), отрисовка эллипса (окружности);
- DrawArc(), отрисовка дуги(для комплексного чертежа). с соответствующими размерами, координатами и двумя углами:
  - startAngle - угол (в градусах), который измеряется по часовой стрелке, начиная от оси X и заканчивая начальной точкой дуги.
  - sweepAngle - угол (в градусах), который измеряется по часовой стрелке, начиная от значения параметра startAngle и заканчивая конечной точкой дуги.
- DrawText(), отрисовка текста;

Проблема мерцания является довольно распространенной проблем при работе с графикой. Из-за того, что многие графические операции требуют сложных операций отрисовки, визуализируемые изображения начинают мерцать, тем самым становясь неудобными для рассмотрения и изучения. Существуют следующие варианты устранения данной проблемы:

1) Осуществление отрисовки при помощи битовой карты (Bitmap – объект, используемый для работы с изображениями, определяемых данными о пикселях);
2) Использование двойной буферизации

Решать проблему мерцания мы будем при помощи 2-го варианта: использование **двойной буферизации.**

При двойной буферизации для решения проблем, связанных с многократным повторением операций рисования, используется буфер в памяти. Если двойная буферизация включена, все операции рисования сначала выполняются в памяти, а лишь затем на экране компьютера. После завершения всех операций рисования содержимое буфера копируется из памяти непосредственно на связанную с ним область экрана.

3.2 Визуализация пространственного чертежа

Обобщенный алгоритм:

1. Отрисуем координатные оси, соединив точки их начала и конца линиями, отрисовку которых мы осуществим при помощи метода DrawLine();
2. Отрисуем линии связи, соединив соответствующие точки проекций на координатные оси и плоскости линиями, отрисовку которых мы осуществим при помощи метода DrawLine();
3. Отрисуем точки из соответствующего массива 14-ти 2D-точек при помощи метода FillEllipse() (отрисовку проводим в цикле, так как все 14 точек рисуются одинаковым образом);
4. Подпишем каждую точку и оси координат, отрисовав текст при помощи метода DrawString().

3.3 Визуализация комплексного чертежа

Обобщенный алгоритм:

1. Отрисуем координатные оси, соединив точки их начала и конца линиями, отрисовку которых мы осуществим при помощи метода DrawLine();
2. Отрисуем линии связи, соединив соответствующие точки проекций на координатные оси и плоскости линиями, отрисовку которых мы осуществим при помощи метода DrawLine();

3. Отрисуем дугу, которая будет соединять точки проекций на ось Оупри помощи метода DrawArc(). Причем отрисовывать дугу мы будем двумя способами (сразными начальными углами), в зависимости от того, какие знаки будут у координат точек проекций на ось Оу.
4. Отрисуем точки из соответствующего массива 12-ти 2D-точек при помощи метода FillEllipse() (отрисовку проводим в цикле, так как все 12 точек рисуются одинаковым образом);
5. Подпишем каждую точку и оси координат, отрисовав текст при помощи метода DrawString().

Описание применяемых методов (3-й уровень детализации)

Ввод

Введем необходимые **типы данных**:

- CPoint3D(для 3D-точки)
- CPoint2D (для 2D-точки)

Соответствующие этим типам **структуры данных**:

- structCPoint3D
  {
  x; //Координата точки по Ox
  y; //Координата точки по Oy
  z; //Координата точки по Oz
  name; //Название точки
  color; //Цвет точки
  }

- struct CPoint2D
  {
  x; //Координата точки по Ox
  y; //Координата точки по Oy
  name; //Название точки
  color; //Цвет точки
  }

Базовые (общие) переменные:

- x, y, z – установленные значения соответствующих ползунковых переключателей для осей Ox, Oy, Oz;
- alfa, betta, gamma – установленные значения соответствующих ползунковых переключателей для углов между горизонтальным вектором, направленным влево и положительными направлениями координатных осей Ox, Oy, Oz;
- Center.x, Center.y – координаты середины поля для отрисовки
- lenAxis – длина полуоси;
- offset – дополнительное смещение;
- массив начальных14-ти 3D-точек baseStorage3DPoints[], содержащий следующие точки (в таком порядке):
    1. t – 3D-точка T.
    2. tx, ty, tz – проекции точки Tна координатные оси.
    3. t3, t2, t1 – проекции точки Tна координатные плоскости.
    4. to – точка начала координат.
    5. tox1, tox2 – начальная и конечная точки оси Ox.
    6. toy1, toy2 – начальная и конечная точки оси Oy.
    7. toz1, toz2 – начальная и конечная точки оси Oz.
- fontPoint, brushPoint–шрифт и цвет текста для подписи точек

Переменные для пространственного чертежа:

- Массив 2D-точек storageSpacial2D[t_2d, tx_2d, ty_2d, tz_2d, t3_2d, t2_2d, t1_2d, to_2d, tox1_2d, tox2_2d, toy1_2d, toy2_2d, toz1_2d, toz2_2d], который будет содержать 14 двумерных точек, получившихся в результате преобразования 14-ти трехмерных точек из массива baseStorage3DPoints[].

Переменные для комплексного чертежа:

- Массив 2D-точек storageComplex2D[to, tox, toy3, toz, toy1, t1, t2, t3, tx, ty1, ty3, tz], который содержит 12 двумерных точек, получившихся в результате преобразования 14-ти трехмерных точек из массива baseStorage3DPoints[]:
    - to – точка начала координат;
    - tox, toy3, toz, toy1 – конечные точки для осей Ox, Oy, Oz;
    - t1, t2, t3 – проекции точки Tна координатные плоскости;
    - tx, ty1, ty3, tz – проекции точки Tна координатные оси.

Укрупненный алгоритм:

1. **Ввод информации о координатах точки и углах между горизонтальным вектором и осями координат**

    a) Вводим координаты точки и углымежду горизонтальным вектором и положительными направлениями осей координат при помощи свойства класса ползунковых переключателей TrackBar–Value:
    x = trackBarX.Value;
    y = trackBarY.Value;
    z = trackBarZ.Value;
    alfa = trackBarAlfa.Value;
    beta = trackBarBeta.Value;
    gamma = trackBarGamma.Value.

    b) Вычислим значение длины полуоси по следующей формуле:
    lenAxis = min(Center.x, Center.y) – offset,
    где offset = min(Center.x, Center.y)/3;

    c) Задаемзначенияэлементовмассива baseStorage3DPoints[]:
    baseStorage3DPoints[0] = (x, y, z);
    baseStorage3DPoints[1] = (x, 0, 0);
    baseStorage3DPoints[2] = (0, y, 0);
    baseStorage3DPoints[3] = (0, 0, z);
    baseStorage3DPoints[4] = (0, y, z);
    baseStorage3DPoints[5] = (x, 0, z);
    baseStorage3DPoints[6] = (x, y, 0);
    baseStorage3DPoints[7] = (0, 0, 0);
    baseStorage3DPoints[8] = (lenAxis, 0, 0);
    baseStorage3DPoints[9] = (-lenAxis, 0, 0);
    baseStorage3DPoints[10] = (0, lenAxis, 0);
    baseStorage3DPoints[11] = (0, -lenAxis, 0);
    baseStorage3DPoints[12] = (0, 0, lenAxis);
    baseStorage3DPoints[13] = (0, 0, -lenAxis);

2. **Преобразование трехмерных координат точек в двумерные**

2.1. **Для пространственного чертежа**

1) В цикле каждую из 14-ти трехмерных точек из массива baseStorage3DPoints[] преобразуем в двумерную точку из массива storageSpacial2D[] по формулам **(1)**.

2.2.**Для комплексного чертежа**

1) Вычисляем массив двумерных точек storageComplex2D[] комплексного чертежа для соответствующих трехмерных точек из массива baseStorage3DPoints[] пространственного макета по формулам **(2), (3), (4)** (**расписать подробно, с учетом того, что 3D точек – 14, 2D точек – 12, а формул – 3**)

3. **Визуализация изображения**

3.1 **Визуализация пространственного чертежа**

1) Отрисовка координатных осей
   Необходимо соединить линиями, при помощи метода DrawLine() следующие пары точек:
   - tox1_2d – tox2_2d (storageSpacial2D[8]-storageSpacial2D[9])Ось Ox;
   - toy1_2d – toy2_2d (storageSpacial2D[10] - storageSpacial2D[11])
     ОсьOy;
   - toz1_2d – toz2_2d(storageSpacial2D[12]-storageSpacial2D[13])
     Ось Oz.

2) Отрисовка линий связи между точкой Ти ее проекциями

Необходимо соединить линиями, при помощи метода DrawLine() следующие пары точек:

- t_2d – t1_2d (storageSpacial2D[0] - storageSpacial2D[4])
- t_2d – t2_2d (storageSpacial2D[0] - storageSpacial2D[5])
- t_2d – t3_2d (storageSpacial2D[0] - storageSpacial2D[6])
- t3_2d – ty_2d (storageSpacial2D[4] - storageSpacial2D[2])
- t3_2d – tz_2d (storageSpacial2D[4] - storageSpacial2D[3])
- t2_2d – tx_2d (storageSpacial2D[5] - storageSpacial2D[1])
- t2_2d – tz_2d (storageSpacial2D[5] - storageSpacial2D[3])
- t1_2d – tx_2d (storageSpacial2D[6] - storageSpacial2D[1])
- t1_2d – ty_2d (storageSpacial2D[6] - storageSpacial2D[2])
- tx_2d – to_2d (storageSpacial2D[1] - storageSpacial2D[7])
- ty_2d – to_2d (storageSpacial2D[2] - storageSpacial2D[7])
- tz_2d – to_2d (storageSpacial2D[3] - storageSpacial2D[7])

3) Отрисовка точек

Необходимо отрисовать точки из массивastorageSpacial2D[]при помощи метода FillEllipse():

- В цикле проходимся по элементам(точкам) массива storageSpacial2D[], рисуя окружность с центром, координаты которого равны координатам текущей точки. А также параллельно с этим отрисовываем текст (названия точек с соответствующими координатами)

3.2 **Визуализация комплексного чертежа**

1) Отрисовка координатных осей

Необходимо соединить линиями, при помощи метода DrawLine() следующие пары точек:

- tox–toy3 (storageComplex2D[1] - storageComplex2D[2]) Ось(X (-Y) – Y(-X));
- toz – toy1; (storageComplex2D[3] - storageComplex2D[4]) Ось (Z(-Y) – Y(-Z));

2) Отрисовка линий связи между проекциямиточки T

Необходимо соединить линиями, при помощи метода DrawLine() следующие пары точек:

- t1 – tx (storageComplex2D[5] - storageComplex2D[8])
- t1 – ty1 (storageComplex2D[5] - storageComplex2D[10])

- t2 – tx (storageComplex2D[6] - storageComplex2D[8])
- t2 – tz (storageComplex2D[6] - storageComplex2D[11])
- t3 – ty3 (storageComplex2D[7] - storageComplex2D[11])
- t3 – tz (storageComplex2D[7] - storageComplex2D[9])

3) Отрисовка дуги между проекциями точкиТ на Оу
   Необходимо соединить дугой, при помощи метода DrawArc() следующие пары точек:
   - ty1 – ty3 (storageComplex2D[9] - storageComplex2D[10])

   Рассмотрим этот момент более подробно.
   Пустьty1(x1, y1); ty3(x3, y3)

   - При x3< 0 (или y1< 0)
     Дуга должна отрисовываться в четвертом квадранте с параметрами:
     1) Центр в точке О;
     2) Радиус, равный расстоянию от точки О до ty1 (или до точки Ty3);
     3) Начальный угол, равный 180 градусов;
     4) Полный угол, равный 90 градусов.

   - При x3> 0 (или y1> 0)
     Дуга должна отрисовываться во втором квадранте с параметрами:
     1) Центр в точке О;
     2) Радиус, равный расстоянию от точки О до ty1 (или до точки Ty3);
     3) Начальный угол, равный 0 градусов;
     4) Полный угол, равный 90 градусов.

4) Отрисовкаточек
   Необходимо отрисовать точки из массивastorageComplex2D[]при помощи метода FillEllipse():
   - В цикле проходимся по элементам(точкам) массива storageComplex2D[], рисуя окружность с центром, координаты которого равны координатам текущей точки. А также параллельно с этим отрисовываем текст (названия точек с соответствующими координатами)

Детализированный алгоритм:

В связи с тем, что координаты каждой точки вычисляются одинаковым образом (для пространственного чертежа), целесообразно реализовать данные вычисления как функцию, основанную на формуле **(1)**:

Функция Преобразовать_координаты_Space(Point2D, Point3D, alfa, betta, gamma, Center)
{
Point2D.x = Center.x –Point 3D.x*cos(alfa) –Point3D.y *cos(betta) – Point3D.z *cos(gamma);

Point2D.y = Center.y + Point3D.x *sin(alfa) + Point3D .y *sin (betta) + Point3D .z *sin(gamma);
}

В связи с тем, что координаты каждой из трёх проекций точки вычисляются одинаковым образом (для комплексного чертежа), целесообразно реализовать данные вычисления как функции, основанные на формулах **(2), (3) и (4)**:

Функция Преобразовать_координаты_Complex1(…)
{
…
}
Функция Преобразовать_координаты_Complex2(…)
{
…
}
Функция Преобразовать_координаты_Complex3(…)
{
…
}

1. **<u>Ввод информации о координатах точки и углах между горизонтальным вектором и осями координат</u>**

Считываем данные с ползунковых переключателей, производим необходимые действия и заносим данные в массив 14-ти 3D-точек baseStorage3DPoints[]

```
x = trackBarX.Value;
y = trackBarY.Value;
z = trackBarZ.Value;
alfa = Перевести_в_радианы(trackBarAlfa.Value);
betta = Перевести_в_радианы(trackBarBeta.Value);
gamma = Перевести_в_радианы(trackBarGamma.Value);
baseStorage3DPoints[0] = (x, y, z);
baseStorage3DPoints[1] = (x, 0, 0);
baseStorage3DPoints[2] = (0, y, 0);
baseStorage3DPoints[3] = (0, 0, z);
baseStorage3DPoints[4] = (0, y, z);
baseStorage3DPoints[5] = (x, 0, z);
baseStorage3DPoints[6] = (x, y, 0);
baseStorage3DPoints[7] = (0, 0, 0);
baseStorage3DPoints[8] = (lenAxis, 0, 0);
baseStorage3DPoints[9] = (-lenAxis, 0, 0);
baseStorage3DPoints[10] = (0, lenAxis, 0);
baseStorage3DPoints[11] = (0, -lenAxis, 0);
baseStorage3DPoints[12] = (0, 0, lenAxis);
baseStorage3DPoints[13] = (0, 0, -lenAxis);
```

2. **Преобразование трехмерных координат точек в двумерные**

2.1. **Для пространственногочертежа**

Цикл: для i от 0 до 13 с шагом 1 выполнить:
{
Преобразовать_координаты_Space(storageSpacial2D[i],
baseStorage3DPoints[i], alfa, betta, gamma, Center)

}

2.2. **Для комплексного чертежа**

// Вычислить 12-ть точек комплексного чертежа storageComplex2D[] по
// соответствующим точкам пространственного макета
// baseStorage3DPoints[], используя соответствующие функции:
// *Преобразовать_координаты_Complex1*(…) или
// *Преобразовать_координаты_Complex2*(…) или
// *Преобразовать_координаты_Complex3*(…).

storageComplex2D[0] = …
…
storageComplex2D[11] = …

3. **Визуализация изображения**

3.1. **Визуализация пространственного чертежа**

1) Отрисовка координатных осей

- DrawLine(storageSpacial2D[8], storageSpacial2D[9]); *//Ox*
- DrawLine(storageSpacial2D[10], storageSpacial2D[11]); *//Oy*
- DrawLine(storageSpacial2D[12], storageSpacial2D[13]); *//Oz*

2) Отрисовка линий связи между точкой T и ее проекциями

- DrawLine(storageSpacial2D[0], storageSpacial2D[4]);*// T-T1*
- DrawLine(storageSpacial2D[0], storageSpacial2D[5]);*// T-T2*
- DrawLine(storageSpacial2D[0], storageSpacial2D[6]);*// T-T3*
- DrawLine(storageSpacial2D[4], storageSpacial2D[2]);*// T3-Ty*

- DrawLine(storageSpacial2D[4], storageSpacial2D[3]);*// T3-Tz*
- DrawLine(storageSpacial2D[5], storageSpacial2D[1]);*// T2-Tx*
- DrawLine(storageSpacial2D[5], storageSpacial2D[3]);*// T2-Tz*
- DrawLine(storageSpacial2D[6], storageSpacial2D[1]);*// T1-Tx*
- DrawLine(storageSpacial2D[6], storageSpacial2D[2]);*// T1-Ty*
- DrawLine(storageSpacial2D[1], storageSpacial2D[7]);*// Tx-O*
- DrawLine(storageSpacial2D[2], storageSpacial2D[7]);*// Ty-O*
- DrawLine(storageSpacial2D[3], storageSpacial2D[7]);*// Tz-O*

3) Отрисовка точеки текста

Цикл: для i = 0 до 13 с шагом 1:
{
FillEllipse(storageSpacial2D[i].x - r, storageSpacial2D[i].y - r, 2*r, 2*r)
DrawString(storageSpacial2D[i].name, storageSpacial2D[i].x, storageSpacial2D[i].y)
}

3.2. **Визуализация комплексного чертежа**

1) Отрисовка координатных осей

- DrawLine(storageComplex[1], storageComplex[2]) //*X(-Y) – Y(-X)*
- DrawLine(storageComplex[3], storageComplex[4])//*Z(-Y) – Y(-Z)*

2) Отрисовка линий связи между проекциямиточки T

- DrawLine(storageComplex2D[5], storageComplex[8]) *//T1 - Tx*
- DrawLine(storageComplex2D[5], storageComplex[10]) *//T1 – Ty1*
- DrawLine(storageComplex2D[6], storageComplex[8]) *//T2 - Tx*
- DrawLine(storageComplex2D[6], storageComplex[11]) *//T2 - Tz*
- DrawLine(storageComplex2D[7], storageComplex[11]) *//T3 – Ty3*
- DrawLine(storageComplex2D[7], storageComplex[9]) *//T1 –Tz*

3) Отрисовкадуги между проекциями точкиТ на Оу

d = 2*|Center.y – Ty3.y|;

Если (y < 0)
	DrawArc(Center.x – d/2, Center.y – d/2, d, d, 180, 90);
иначе
	DrawArc(Center.x – d/2, Center.y – d/2, d, d, 0, 90);

4) Отрисовка точеки текста

Цикл: для i = 0 до 11 с шагом 1:
{
	FillEllipse(storageComplex2D[i].x - r, storageComplex2D[i].y - r, 2*r, 2*r)
	DrawString(storageComplex2D[i].name, storageComplex2D[i].x, storageSpacial2D[i].y)
}

Руководство программиста

Данная лабораторная работа разработана на платформе Microsoft .NETFrameworkверсии 4.5.5, в среде разработки VisualStudio 2010 версии 10.0.3,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 данныхиспользуются ползунковые переключатели, объекты класса “TrackBar”. Для вывода кнопки “Выйти из программы” используется объект класса “Button”. Для вывода необходимого текста используются объекты класса “Label”.

Все компоненты, представленные вышеи составляющиепользовательский интерфейс данной лабораторной работы, отображаются в окне (объекте класса “Form”). Это представлено на рисунке 8.

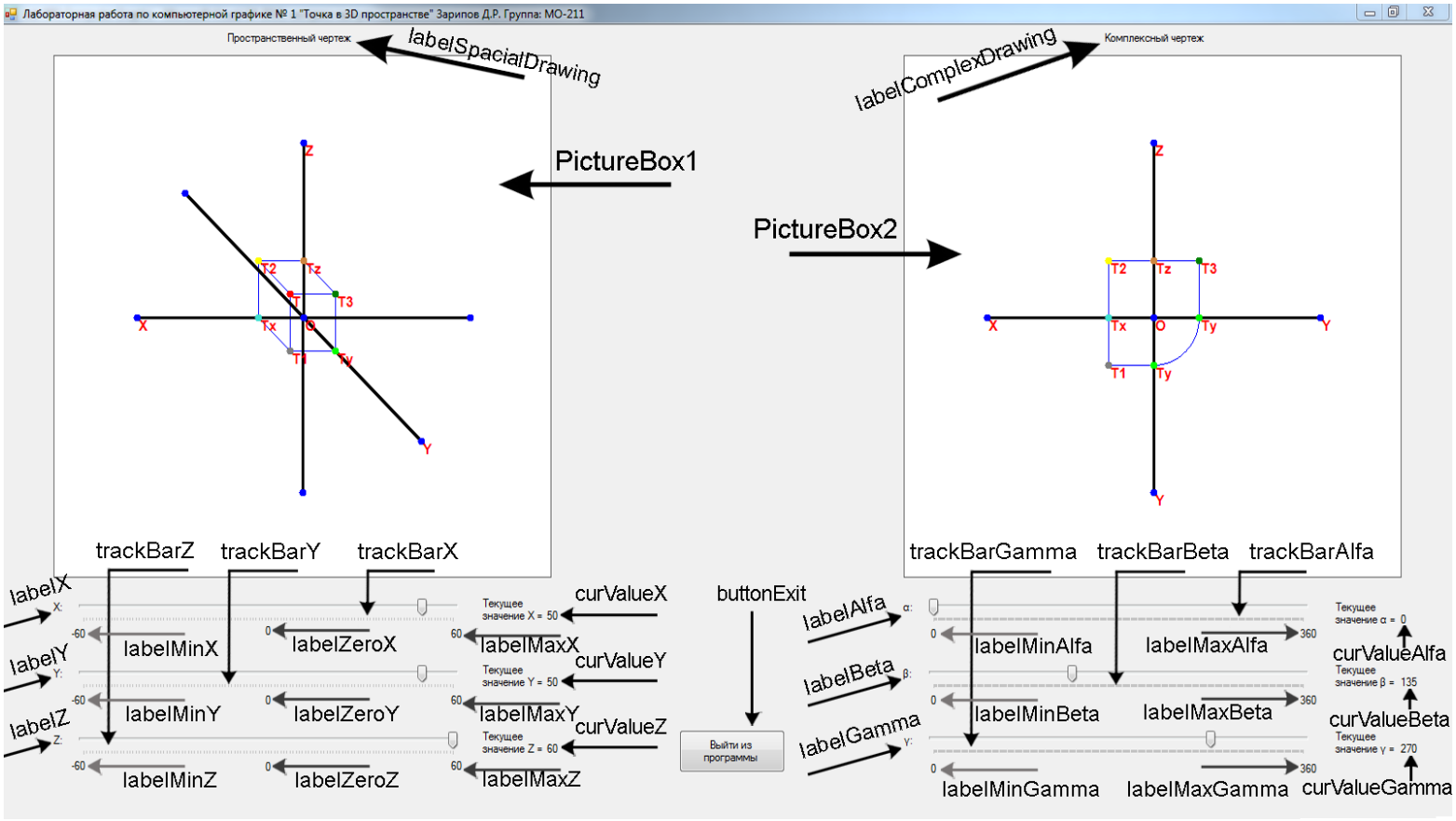

Рис.8 Пользовательский интерфейс

Представим информацию об основных объектах интерфейса:

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

- TrackBar
    1. trackBarX – ввод координаты x точки T;
    2. trackBarY – ввод координаты y точки T;
    3. trackBarZ – ввод координаты z точки T;
    4. trackBarAlfa – ввод угла между горизонтальным вектором, направленным влево и положительным направлением Ox;
    5. trackBarBeta – ввод угла между горизонтальным вектором, направленным влево и положительным направлением Oy;

6. trackBarGamma – ввод угла между горизонтальным вектором, направленным влево и положительным направлением Oz.

- Label
    1. curValueX – показание текущего введенного значения координаты x точки T;
    2. curValueY –показание текущего введенного значения координаты y точки T;
    3. curValueZ – показание текущего введенного значения координаты z точки T;
    4. curValueAlfa–показание текущего введенного значения угла между горизонтальным вектором, направленным влево и положительным направлением Ox;
    5. curValueBeta–показание текущего введенного значения угла между горизонтальным вектором, направленным влево и положительным направлением Oy;
    6. curValueGamma–показание текущего введенного значения угла между горизонтальным вектором, направленным влево и положительным направлением Oz.
    7. labelX– отображение подписи “X”
    8. labelY– отображение подписи “Y”
    9. labelZ– отображение подписи “Z”
    10. labelMinX– отображение минимального значения координаты x
    11. labelMinY– отображение минимального значения координаты y
    12. labelMinZ– отображение минимального значения координаты z
    13. labelZeroX–отображение нуля (координата x)
    14. labelZeroY– отображение нуля (координата y)
    15. labelZeroZ– отображение нуля (координата z)
    16. labelMaxX–отображение максимального значения координаты x
    17. labelMaxY–отображение максимального значения координаты y
    18. labelMaxZ–отображение максимального значения координаты z
    19. labelAlfa– отображение подписи “α”
    20. labelBeta– отображение подписи “β”

21. labelGamma– отображение подписи “γ”
22. labelMinAlfa– отображение минимального значения угла между горизонтальным вектором, направленным влево, и положительным направлением оси Ox (α)
23. labelMinBeta–отображение минимального значения угла между горизонтальным вектором, направленным влево, и положительным направлением оси Oy (β)
24. labelMinGammaотображение минимального значения угла между горизонтальным вектором, направленным влево, и положительным направлением оси Oz (γ)
25. labelMaxAlfa–отображение максимального значения угла между горизонтальным вектором, направленным влево, и положительным направлением оси Ox (α)
26. labelMaxBeta–отображение максимального значения угла между горизонтальным вектором, направленным влево, и положительным направлением оси Oy (β)
27. labelMaxGamma– отображение максимального значения угла между горизонтальным вектором, направленным влево, и положительным направлением оси Oz (γ)
28. labelSpacialDrawing – отображение подписи для изображения пространственного чертежа
29. labelComplexDrawing – отображение подписи для изображения комплексного чертежа

- Button
    1. buttonExit – выход из программы.

1.2. Прикладная часть

Программа состоит из пяти внутренних классов:

- **Axis**
- **CPoint3D**
- **CPoint2D**

- **SpacialDrawing**
- **ComplexDrawing**

и одного внешнего класса:

- **Form1**

Рассмотрим отдельно каждый класс.

- **Axis**

  Внутренний класс программы, отвечающий за передачу центра координат (точки О) области рисования PictureBox.
  Данный класс содержит следующие поля и методы:

  int **xCenter**; - *поле, содержащее координатух точки О;*
  int **yCenter**; - *поле, содержащее координатуу точки О;*

  Axis(int **xCenter,** int **yCenter**); - *конструктор класса с 2-мя параметрами: координатой х точки О и координатой у точки О, которые передаются из других внутренних или внешнего классов программы.*

  int **getCenterX()**; -*метод, возвращающий координату х точки О;*
  int **getCenterY()**; - *метод, возвращающий координату у точки О;*

- **CPoint3D**

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

  int **x**;
  поле, содержащее координату трехмерной точки по оси Ох;

  int **y**;
  поле, содержащее координату трехмерной точки по оси Оу;

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

string **name**;
поле, содержащее наименование трехмерной точки;

Brush **color**;
поле, содержащее цвет закраски трехмерной точки;

CPoint3D(int **x**, int **y**, int **z**, string **name**, Brush **color**);
конструкторклассас5-юпараметрами: координаты трехмерной точки по каждой из трех осей координат; наименование точки; цвет точки.

- **CPoint2D**

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

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

  int **y**;
  поле, содержащее координату двумерной точки по оси Oy;

  string **name**;
  поле, содержащее наименование двумерной точки;

  Brush **color**;
  поле, содержащее цвет закраски двумерной точки;

  CPoint2D(int**x**, int**y**, string**name**, Brush**color**);
  конструкторклассас4-мя параметрами: координаты трехмерной точки по каждой из двух осей координат; наименование точки; цвет точки.

- **SpacialDrawing**

  Внутренний класс программы, описывающий пространственный чертеж и содержащий следующие поля и методы:

  CPoint3D[] **storageSpacial3D[14]**;
  поле(массив), содержащее 14 начальных 3D-точек

  CPoint2D[] **storageSpacial2D[14]**;
  поле(массив), содержащее 14 2D-точек, получаемых при помощи преобразования 14-ти 3D-точек из массива storageSpacial3D[] в 14 2D-точек.

  double **alfa**;
  поле, содержащее угол между горизонтальным вектором экрана, направленным влево и положительным направлением Ox;

  double **betta**;
  поле, содержащее угол между горизонтальным вектором экрана, направленным влево и положительным направлением Oy;

  double **gamma**;
  поле, содержащее угол между горизонтальным вектором экрана, направленным влево и положительным направлением Oz;

  SpacialDrawing();
  конструктор класса без параметров

  void **calculatePoints()**;
  метод, осуществляющий преобразование трехмерных 14-ти точек из массива storageSpacial3D[] в 14 двумерных точек, записывая результаты в массив storageSpacial2D[];

  void **updatePoints(**CPoint3D[] **storage**, double**alfa**, double**betta**, double **gamma)**;
  метод, осуществляющийпересчеткоординат 2D-точекприизмененииввodимыхтрехмерныхкоординат (x, y, z) иугловмеждугоризонтальнымвектором, направленнымвлево, иположительныминаправлениямиосейOx, Oy, Oz (alfa, betta, gamma)

  void **drawPoints**(Graphics**g**, Font**fontNames**, Pen**colorLines**);
  метод, осуществляющийотрисовкукоординатныхосей, всехточек,

*линийсвязи(цвет:*colorLines)*итекста (наименованийточек; шрифт:*fontNames)*напространственноммакете. Данныйметод "разделен" надва "подметода":*

void**_drawLines**(Graphics**g**, Pen**colorLines)**;
метод, осуществляющийотрисовкукоординатныхосейилинийсвязи(цвет: colorLines*);*

void**_drawPoints**(Graphics**g**, Font**fontNames)**;
метод, осуществляющийотрисовкуточекиихнаименований(шрифт : fontNames*)*

- **ComplexDrawing**

Внутренний класс программы, описывающий пространственный чертеж и содержащий следующие поля и методы:

CPoint3D[] **storageComplex3D[14]**;
поле(массив), содержащее 14 начальных 3D-точек

CPoint2D[] **storageComplex2D[12]**;
поле(массив), содержащее 12 2D-точек, получаемых при помощи преобразования 14-ти 3D-точек из массива storageComplex3D[] в 12 2D-точек.

ComplexDrawing();
конструктор класса без параметров

void **calculatePoints()**;
метод, осуществляющий преобразование трехмерных 14-ти точек из массива storageComplex3D[] в 12 двумерных точек, записывая результаты в массив storageComplex2D[];

void **updatePoints(**CPoint3D[] **storage)**;
метод, осуществляющийпересчеткоординат 2D-точекприизмененииввводимыхтрехмерныхкоординат (x, y, z);

void **drawPoints**(Graphics**g**, Font**fontNames**, Pen**colorLines**);
*метод, осуществляющийотрисовкукоординатныхосей, всехточек, линийсвязи(цвет:*colorLines)*итекста (наименованийточек; шрифт:*fontNames)*напространственноммакете. Данныйметод "разделен" надва "подметода":*

void**_drawLines**(Graphics**g**, Pen**colorLines)**;
метод, осуществляющийотрисовкукоординатныхосейилинийсвязи(цвет: colorLines*);*

void**_drawPoints**(Graphics**g**, Font**fontNames)**;
метод, осуществляющийотрисовкуточекиихнаименований(шрифт: fontNames*)*

- **Form1**

Внешний класс программы, содержащий следующие поля и методы:

CPoint3D[] **baseStorage3DPoints [14]**;
поле(массив), содержащее 14 вводимых пользователем 3D-точек

int **x**;
поле, содержащее координату по оси Ox, вводимую пользователем;

int **y**;
поле, содержащее координату по оси Oy, вводимую пользователем;

int **z**;
поле, содержащее координату по оси Oz, вводимую пользователем;

double **alfa**;
поле, содержащее угол, вводимый пользователем, между горизонтальным вектором экрана, направленным влево, и положительным направлением Ox;

double **betta**;
поле, содержащее угол, вводимый пользователем, между горизонтальным вектором экрана, направленным влево, и положительным направлением Oy;

double **gamma**;
поле, содержащее угол, вводимый пользователем, между горизонтальным вектором экрана, направленным влево и положительным направлением Oz;

SpacialDrawing **spacialDrawing;**
поле, представляющее собой объект класса пространственного чертежа;

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

void **input()**;
метод, обрабатывающий вводимые пользователем данные о координатах и об углах;

void **processSpacial()**;
метод, пересчитывающий вводимые пользователем координаты 3D-точки из трехмерных в двумерные для пространственного чертежа;

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

void **outputSpacial()**;
метод, выводящий пространственный чертеж;

void **outputComplex()**;
метод, выводящийкомплексныйчертеж;

void **createPoints()**;
метод, осуществляющий создание 3D-точек с соответствующими вводимыми пользователем 3D-координатамии запись их в массив трехмерных точек baseStorage3DPoints;

double **convertToRadian(**int**angle)**;
метод, осуществляющий перевод градусов в радианы для данного угла(angle)*;*

Также в данном внешнем классе происходят обработки следующих событий:

void **pictureBox1_Paint()**;
событие перерисовки пространственного чертежа;

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

void **trackBarX_Scroll()**;
событие взаимодействия пользователя с ползунковым переключателем для изменения текущего значения координаты ув соответствующем label;

void **trackBarY_Scroll()**;
событие взаимодействия пользователя с ползунковым переключателем для изменения текущего значения координаты ув соответствующем label;

void **trackBarZ_Scroll()**;
событие взаимодействия пользователя с ползунковым переключателем для изменения текущего значения координаты ув соответствующем label;

void **trackBarAlfa_Scroll()**;
событие взаимодействия пользователя с ползунковым переключателем для изменения текущего значения угла между

горизонтальным вектором, направленным влево, и положительным направлением оси Ox в соответствующем label;

void **trackBarBeta_Scroll()**;
событие взаимодействия пользователя с ползунковым переключателем для изменения текущего значения угла между горизонтальным вектором, направленным влево, и положительным направлением оси Oy в соответствующем label;

void **trackBarGamma_Scroll()**;
событие взаимодействия пользователя с ползунковым переключателем для изменения текущего значения угла между горизонтальным вектором, направленным влево, и положительным направлением оси Oz в соответствующем label;

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

Покажем на функциональной схеме (рис. 9), как взаимодействуют между собой функции и методы, а также как происходит их взаимодействие с элементами интерфейса.

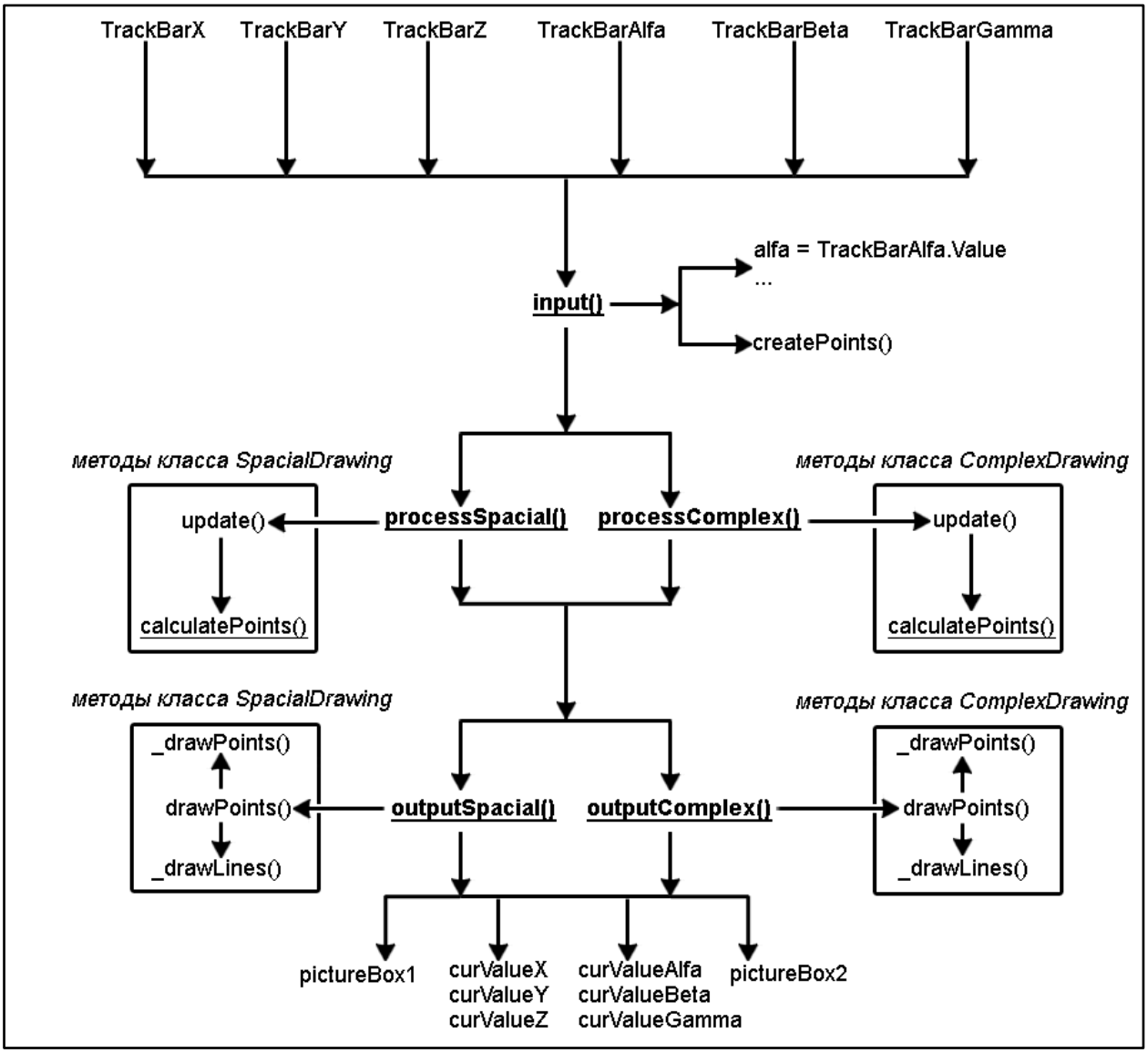

Рис.9 Функциональная схема программы

Как видно из данной функциональной схемы, в программе выделены 5 “листьев”, которые обозначили в структуре решения (на первом уровне детализации), выполняющихся последовательно и независимо друг от друга. Это является минимальным делением программы. На рисунке эти 5 “листьев” данного деления выделены жирным и подчеркнуты.

Кроме того, необходимо выделить также так называемое “Нормальное” деление программы – это деление программы с учетом специфики “Методов решения” (а именно, четырех методов, при помощи которых происходит пересчет координат всех точек из 3D в 2D для пространственного (один метод) и комплексного (три метода) чертежей). То есть в данном делении программы добавляются еще четыре “листа” к 5 выделенным ранее в структуре решения (итого получается 9 “листьев”). На рисунке показаны 7 из 9-ти “листьев” (они подчеркнуты).

2. Как программа работает?

При запуске программы происходит инициализация главной формы и создание переменных:**x, y, z, alfa, betta, gamma.**

Далее создаются объекты классов пространственного и комплексного чертежей:
SpacialDrawing**spacialDrawing**;
ComplexDrawing**complexDrawing**;
после чего резервируется память для массивов:
CPoint3D **baseStorage3DPoints[];**
CPoint2D **storageSpacial2D[];**
CPoint2D **storageComplex[];**

Далее вызывается функция **input()**, которая присваивает текущие значения на соответствующих ползунках переменным **alfa, betta, gamma**:
alfa = trackBarAlfa.Value;
betta = trackBarBeta.Value;
gamma = trackBarGamma.Value.
Затем вызывается функция **createPoints()**, которая присваивает переменным **x**, **y**, **z**текущие значения с соответствующих ползунковых переключателей:
x = trackBarX.Value;
y = trackBarY.Value;
z = trackBarZ.Value;
ипослечегозаполняетмассив 3D-точек **baseStorage3DPoints[]**

Далее вызывается функция **processSpacial()**(пересчет координат точек для пространственного чертежа), которая в свою очередь вызывает функцию **updatePoints(baseStorage3DPoints[], alfa, betta, gamma)**(обновление точек для пространственного чертежа), которая в свою очередь вызывает функцию **calculatePoints()**, где происходит преобразование координат точек из 3Dв 2D для пространственного чертежа.
После этого вызывается функция **outputSpacial()**(вывод пространственного чертежа), которая в свою очередь вызывает функцию отрисовки пространственного чертежа **drawPoints()**, которая в свою очередь вызывает две функции: отрисовка линий связи **_drawLines()**и

отрисовку точек с наименованиями **_drawPoints()**для пространственного чертежа.

Далее аналогично происходит и с комплексным чертежом:
Вызывается функция **processComplex()**(пересчет координат точек для комплексного чертежа), которая в свою очередь вызывает функцию **updatePoints(updatePoints(baseStorage3DPoints[])**(обновление точек для комплексного чертежа), которая в свою очередь вызывает функцию **calculatePoints()**, где происходит преобразование координат точек из 3Dв 2D для комплексного чертежа.
После этого вызывается функция **outputSpacial()**(вывод комплексного чертежа), которая в свою очередь вызывает функцию отрисовки комплексного чертежа **drawPoints()**, которая в свою очередь вызывает две функции: отрисовка линий связи **_drawLines()**и отрисовку точек с наименованиями **_drawPoints()**для комплексного чертежа.

Примечание: *Так как при изменении углов между горизонтальным вектором, направленным влево и положительными направлениями осей меняется только пространственный чертеж, а комплексный остается неизменным, мы в функцию обновления точек пространственного чертежа передаем в качестве параметров массив 3D-точек и данные углы, а в функцию обновления точек комплексного чертежа передаем в качестве параметров только массив 3D-точек.*

В итоге пространственный и комплексный чертежи выводятся соответственно в “пикчербоксы”**pictureBox1**и **pictureBox2**, а текущие значения координат точки T: x, y, zи углов между горизонтальным вектором, направленным влево и положительными направлениями осей: Ox, Oy, Oz выводятся соответственно в “лэйблы”**curValueX**, **curValueY**, **curValueZ**, **curValueAlfa**, **curValueBeta**, **curValueGamma**.

При нажатии на кнопку(Button) **buttonExit** (выход из программы), программа, вызывая команду Application.Exit(), завершает свою работу.

При изменении положения ползунковых переключателей **trackBarX, trackBarY, trackBarZ, trackBarAlfa, trackBarBeta, trackBarGamma** вызывается событие **Scroll**, в обработке которого фиксируются значения текущих величин на соответствующих **Lable**:

curValueX.Text = trackBarX.Value.ToString(),
curValueY.Text = trackBarY.Value.ToString(),
curValueZ.Text = trackBarZ.Value.ToString(),
curValueAlfa.Text = trackBarAlfa.Value.ToString(),
curValueBeta.Text = trackBarBeta.Value.ToString(),
curValueGamma.Text = trackBarGamma.Value.ToString()

Кроме того, событие **Scroll** вызывает функцию обновления (перерисовки) изображения **Refresh()**(для изображений пространственного и комплексного чертежей в случае изменения значений координат точки T и только для пространственного чертежа, в случае изменения значений углов между горизонтальным вектором, направленным влево, и положительными направлениями осей координат), после чего происходит событие Paintдля пространственного макета (**pictureBox1**), обработка которого последовательно вызывает функции:

- input();
- processSpacial();
- outputSpacial();

и событие Paint для комплексного чертежа (**pictureBox2**), обработка которого последовательно вызывает функции:

- input();
- processComplex();
- outputComplex();

Руководство пользователя

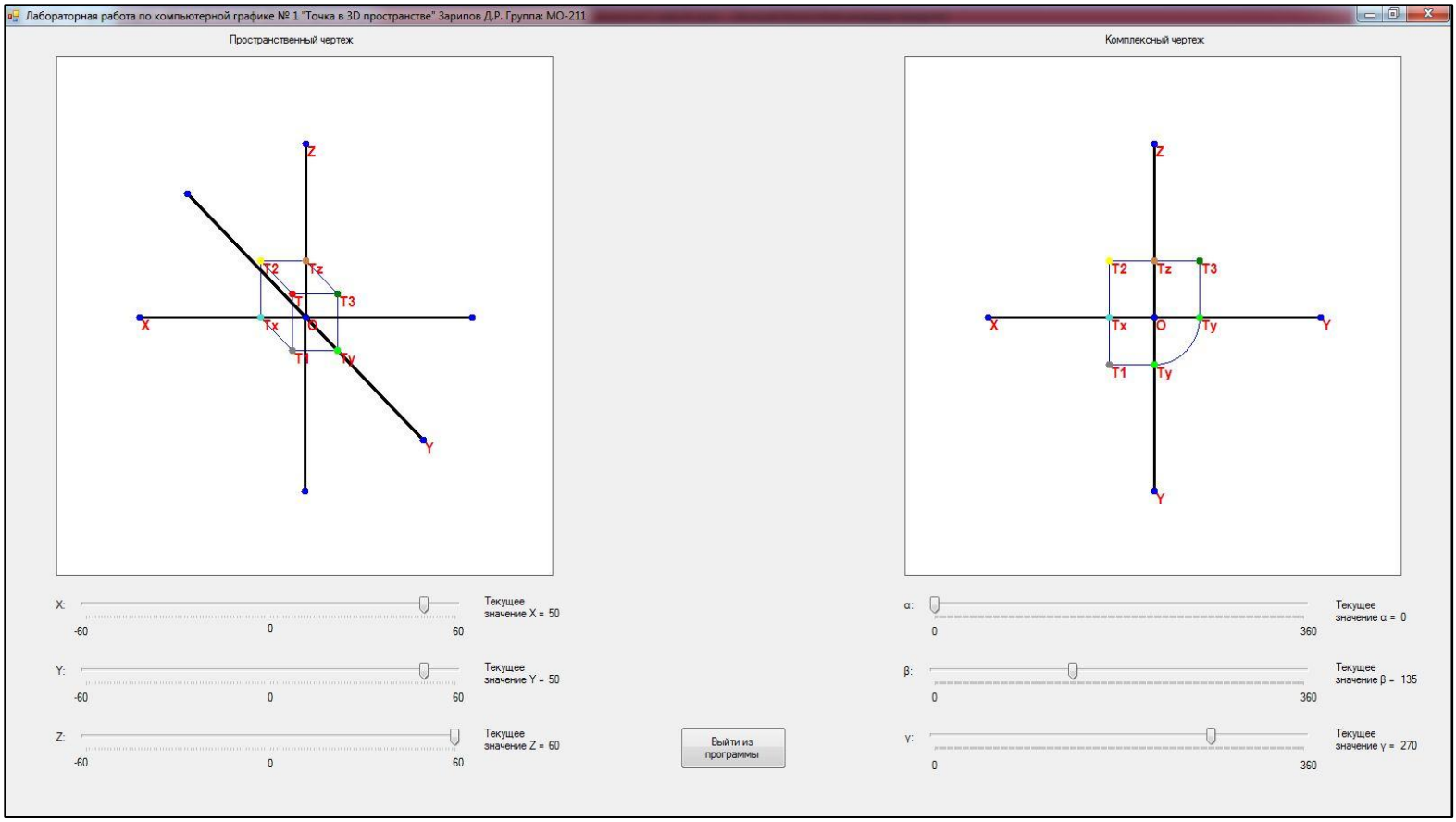

Рис.10

Данная программа предназначена для визуализации пространственного макета и комплексного чертежа 3D-точки T.

Рассмотрим рисунок 10.

Левое изображение является изображением пространственного макета. Правое изображение является изображением комплексного чертежа. Соответствующие наименования точек на обоих изображенияхнаходятся справа от самих точек.

Для того, чтобы задать нужные значения координат точки T: x, y, z необходимо использовать соответствующие ползунковые переключатели в левой нижней части экрана.

Для того, чтобы задать нужные значения углов между горизонтальным вектором, направленным влево и положительными направлениями осей Ox,

Oy, Oz: α, β, γнеобходимо использовать соответствующие ползунковые переключатели в правой нижней части экрана

Минимальные и максимальные возможные значения величин находятся рядом (соответственно с левой и правой сторон) с соответствующими им ползунковым переключателями. Для ползунковых переключателей, задающих значения координат точки Ттакже обозначен переход через 0. Справа от каждого ползункового переключателя отображено текущее значение, зафиксированное пользователем.

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