Содержание

Лабораторная работа № 3

Построение 2D изображений. 2D аффинные преобразования

Цель работы – изучение и реализация алгоритмов аффинных преобразований фигур на плоскости, получение навыков моделирования двумерных объектов.

Краткая теория и порядок выполнения лабораторной работы

Для выполнения лабораторной работы ознакомьтесь с лекционным материалом по рассматриваемой теме.

Для задания модели двумерного объекта используем массив координат вершин фигур (назовем его «матрица тела»). Преобразования производятся умножением матрицы тела на матрицу преобразований, в результате чего получаем новые значения координат вершин фигуры.

Пример 4.1

Разработаем программный модуль для дискретного сдвига фигурки (квадрата) вверх-вниз, вправо-влево. Сдвиг фигурки будет происходить по нажатию кнопки.

1. Для вывода результатов создадим форму:

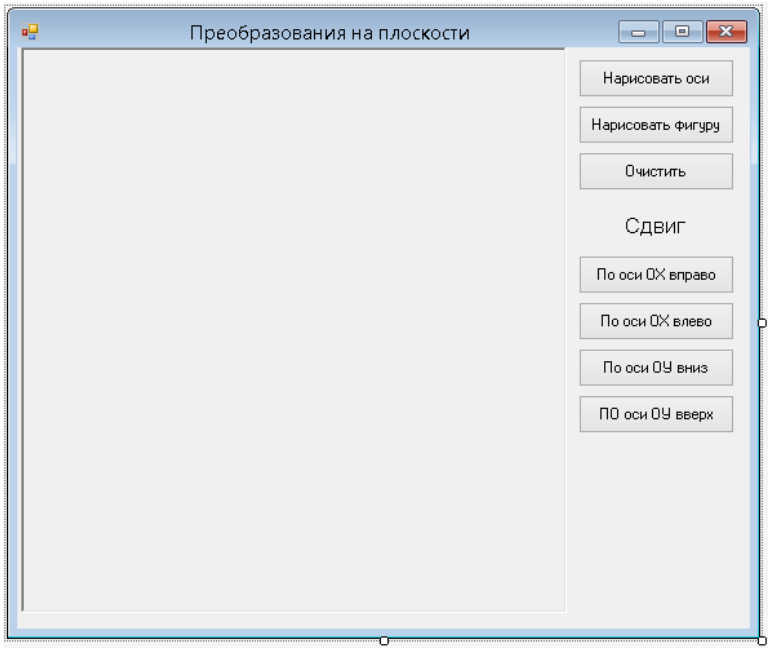


Рисунок 4.1

2. Зададим мировые координаты вершин квадрата:

(-50, 0); (0, 50); (50, 0), (0, -50).

Представим их в однородных координатах:

Однородные координаты

```
//инициализация матрицы тела
        private void Init_kvadrat()
        {
            kv[0, 0] = -50;     kv[0, 1] = 0;       kv[0, 2] = 1;
            kv[1, 0] = 0;       kv[1, 1] = 50;      kv[1, 2] = 1;
            kv[2, 0] = 50;      kv[2, 1] = 0;       kv[2, 2] = 1;
            kv[3, 0] = 0;       kv[3, 1] = -50;     kv[3, 2] = 1;
        }
```

Для перевода мировых координат в экранную систему координат, умножаем их на матрицу сдвига в центр *pictureBox*.

3. Зададим матрицу сдвига:

```
    //инициализация матрицы сдвига
        private void Init_matr_preob (int k1, int l1)
        {
            matr_sdv[0,0]=1;     matr_sdv[0,1]=0;      matr_sdv[0,2]=0;
            matr_sdv[1,0]=0;     matr_sdv[1,1]=1;      matr_sdv[1,2]=0;
            matr_sdv[2,0]=k1;    matr_sdv[2,1]=l1;     matr_sdv[2,2]=1;

         }
```

Пояснения: *т.к. данная матрица будет использоваться дальше для получения двумерного преобразования «сдвиг по оси», определим компоненты матрицы, отвечающие за сдвиг, переменными k1 и l1. Задавая разные значения этих переменных, можно получать разные варианты сдвига.*

4. Для вывода координатных осей создадим массив мировых координат точек, принадлежащих осям:

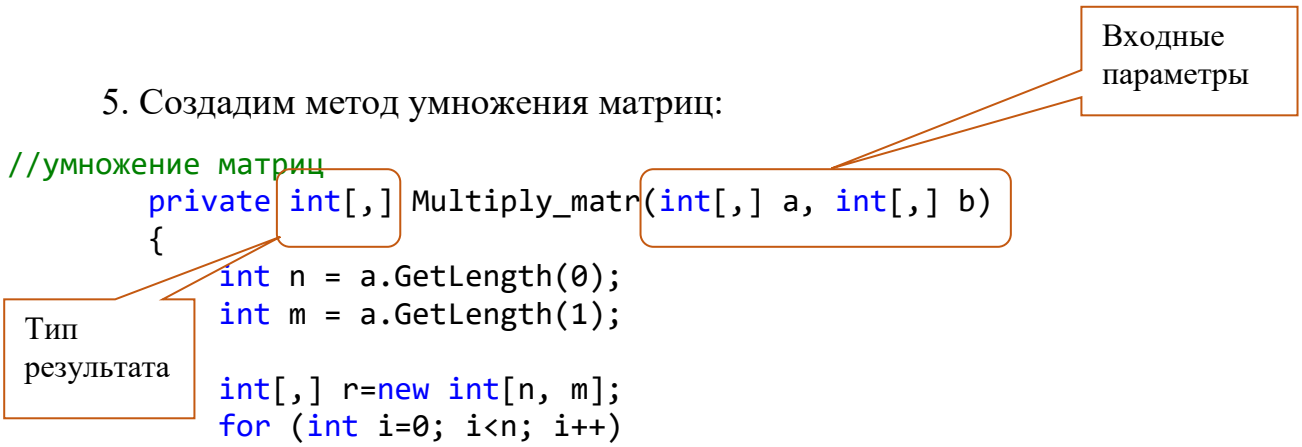
```
        //инициализация матрицы осей
        private void Init_osi()
        {
            osi[0, 0] = -200;   osi[0, 1] = 0;      osi[0, 2] = 1;
            osi[1, 0] = 200;    osi[1, 1] = 0;      osi[1, 2] = 1;
            osi[2, 0] = 0;      osi[2, 1] = 200;    osi[2, 2] = 1;
            osi[3, 0] = 0;      osi[3, 1] = -200;   osi[3, 2] = 1;

}
```

Аналогично, для вывода осей на экран, необходимо переводить их в экранную систему координат, умножая на матрицу сдвига.

***Примечание**: используемые массивы задайте глобальными*

```
int[,] kv = new int[4, 3];  // матрица тела
int[,] osi = new int[4, 3]; // матрица координат осей
int[,] matr_sdv = new int[3, 3]; //матрица преобразования
```

5. Создадим метод умножения матриц:




```
//умножение матриц
        private int[,] Multiply_matr(int[,] a, int[,] b)
        {
            int n = a.GetLength(0);
            int m = a.GetLength(1);

            int[,] r=new int[n, m];
            for (int i=0; i<n; i++)
            {
                for (int j = 0; j < m; j++)
                {
                    r[i, j] = 0;
                    for (int ii = 0; ii < m; ii++)
                    {
                        r[i, j] += a[i, ii] * b[ii, j];
                    }
                }
            }
            return r;
        }
```

***Пояснения**: входные параметры метода – две матрицы a и b, результат метода – матрица r, поэтому в заголовке метода задан тип возвращаемого значения – двумерный массив (**int[,]** Multiply_matr); для передачи результата работы метода (умножения матриц) в другие методы программы в конце тела метода обязательно необходимо задать оператор return r*

6. Создадим метод вывода фигуры на экран (`Draw_Kv`). В данном методе:

6.1) инициализируем массив координат фигуры (*Init_k*`vdrat`)

6.2) инициализируем матрицу сдвига (*Init_matr_preob(k, l)*)

6.3) умножаем матрицу тела на матрицу сдвига (*Multiply_matr(kv, matr_sdv))*

6.4) задаем цвет и ширину карандаша для рисования

6.5) выводим стороны квадрата, используя стандартные методы C#

6.6) освобождаем все используемые ресурсы

```
//вывод квадрата на экран
private void Draw_Kv()
{

  Init_kvadrat(); //инициализация матрицы тела
  Init_matr_preob(k, l); //инициализация матрицы преобразования
  int [,] kv1 = Multiply_matr(kv, matr_sdv); //перемножение матриц

  Pen myPen = new Pen(Color.Blue, 2);// цвет линии и ширина

  //создаем новый обьект Graphics (поверхность рисования) из pictureBox
  Graphics g = Graphics.FromHwnd(pictureBox1.Handle);
   // рисуем 1 сторону квадрата

   g.DrawLine(myPen, kv1[0, 0], kv1[0, 1], kv1[1, 0], kv1[1, 1]);
   // рисуем 2 сторону квадрата
   g.DrawLine(myPen, kv1[1, 0], kv1[1, 1], kv1[2, 0], kv1[2, 1]);
   // рисуем 3 сторону квадрата
   g.DrawLine(myPen, kv1[2, 0], kv1[2, 1], kv1[3, 0], kv1[3, 1]);
   // рисуем 4 сторону квадрата
   g.DrawLine(myPen, kv1[3, 0], kv1[3, 1], kv1[0, 0], kv1[0, 1]);

   g.Dispose();// освобождаем все ресурсы, связанные с отрисовкой
   myPen.Dispose(); //освобождвем ресурсы, связанные с Pen

}
```

***Примечание**: не забудьте задать глобальные переменные k, l*

```
int k, l; // элементы матрицы сдвига
```

7. Вызовем процедуру *Draw_Kv*. в обработчике события нажатия кнопки «Нарисовать фигуру».

```
//вывод квадратика в центре pictureBox
        private void button2_Click(object sender, EventArgs e)
        {
            //середина pictureBox
            k = pictureBox1.Width / 2;
            l = pictureBox1.Height / 2;
```

```
            //вывод квадратика в середине
            Draw_Kv();

        }
```

8. Выполним аналогичные действия для вывода осей координат на экран - создадим метод вывода осей на экран и обработчик события нажатия кнопки «Нарисовать оси»:

```
//вывод осей на экран
    private void Draw_osi()
    {

        Init_osi();
        Init_matr_preob(k, l);
        int[,] osi1 = Multiply_matr(osi, matr_sdv);

        Pen myPen = new Pen(Color.Red, 1);// цвет линии и ширина
        Graphics g = Graphics.FromHwnd(pictureBox1.Handle);

        // рисуем ось OX
        g.DrawLine(myPen, osi1[0, 0], osi1[0, 1], osi1[1, 0], osi1[1,
        1]);

        // рисуем ось OУ
        g.DrawLine(myPen, osi1[2, 0], osi1[2, 1], osi1[3, 0], osi1[3,
        1]);

        g.Dispose();
        myPen.Dispose();

    }

//вывод осей в центре pictureBox
    private void button1_Click(object sender, EventArgs e)
    {
        k = pictureBox1.Width / 2;
        l = pictureBox1.Height / 2;
        Draw_osi();
    }
```

1. Выводим изображение (рис.4.2).

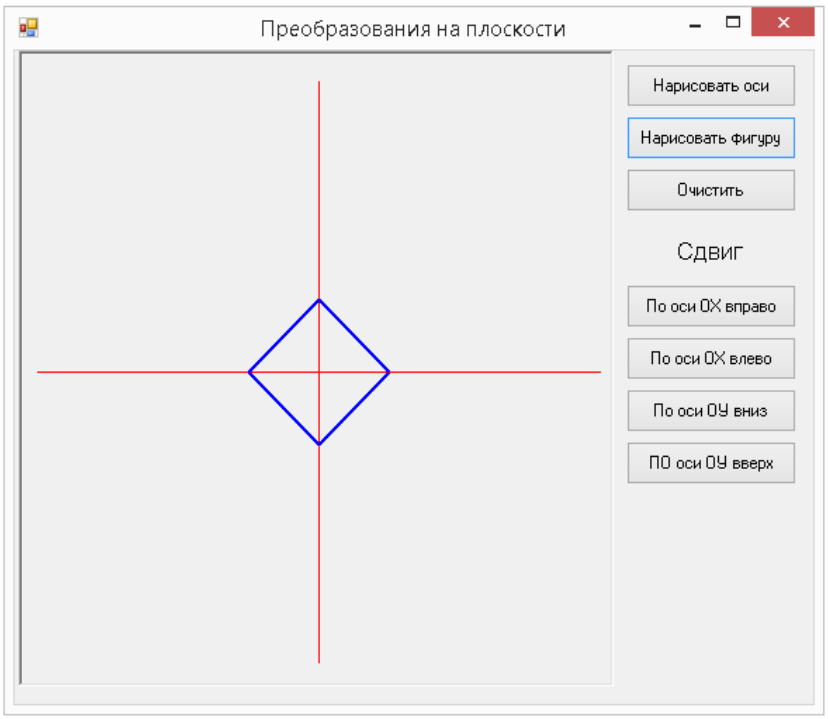


Рисунок 4.2

10. Далее выполняем аффинные преобразования (согласно заданию). Например, для сдвига квадрата право на 5 единиц достаточно увеличить параметр *k* матрицы сдвига, умножить матрицу тела на измененную матрицу сдвига и вывести фигуру на экран (кнопка «По оси OX вправо»):

```
//сдвиг вправо
private void button4_Click(object sender, EventArgs e)
{
    k += 5;  //изменение соответствующего элемента матрицы сдвига
    Draw_Kv(); //вывод квадратика
}
```

Сделав несколько нажатий на кнопку «По оси OX вправо», получаем (рисунок 4.3):

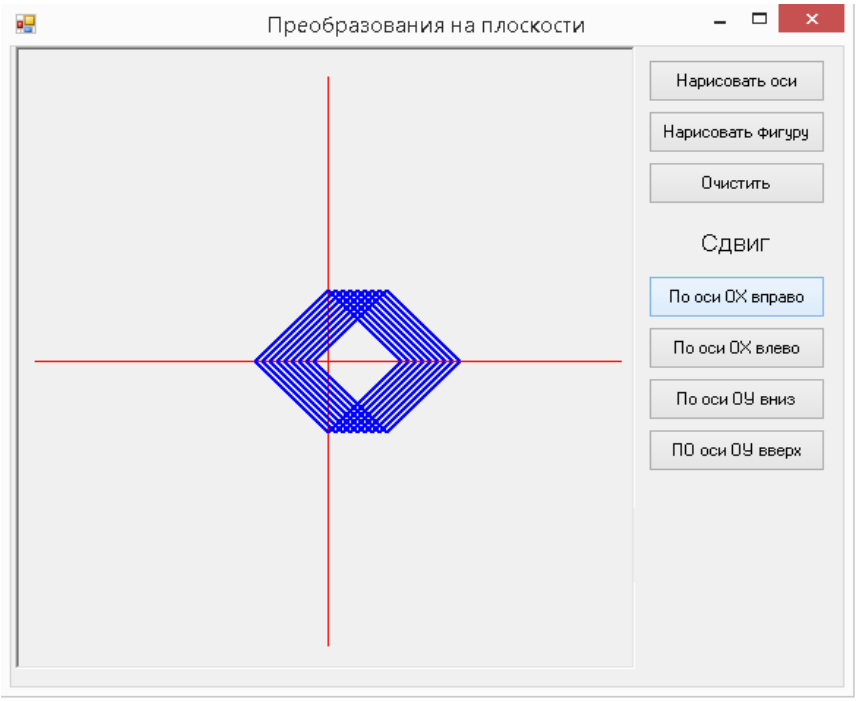


Рисунок 4.3

Пример 4.2

Модифицируем пример 4.1 таким образом, чтобы смещение квадратика вправо происходило не дискретно (т.е. по нажатию кнопки), а непрерывно - автоматически после нажатия кнопки «Старт» (при этом название кнопки «Старт» сменится на «Стоп»). Для остановки перемещения необходимо нажать кнопку «Стоп».

Один из алгоритмов реализации данного задания можно представить так. Введем глобальную булевскую переменную (*f*). Значение *f* при первоначальном выводе квадратика равняется *true*. Цикл перемещения будет происходить по таймеру, пока не нажмем кнопку «Стоп». При нажатии кнопки «Стоп» значению *f* присваивается противоположное значение и движение квадратика останавливается.

Для реализации смещения используем преобразование координат фигуры, описанное в примере 4.1: изменяем компоненты матрицы преобразования, отвечающие за сдвиг, умножаем матрицу тела на матрицу преобразования (в

данном случае, матрицу сдвига), получаем новые координаты фигуры и выводим фигуру на экран. Эти действия повторяем.

1. Добавим на форму кнопку *Button8* и *таймер* (рис.4.4).

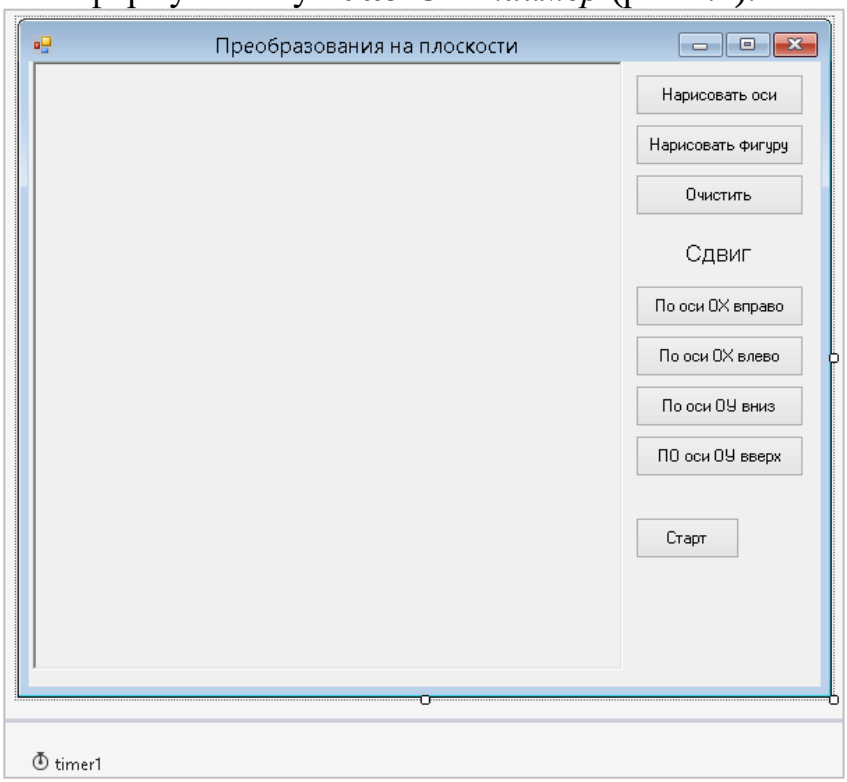


Рисунок 4.4

2. Напишем обработчик нажатия кнопки «Старт»:

```
//непрерывное перемещение
private void button8_Click(object sender, EventArgs e)
{
    timer1.Interval = 100;

    button8.Text = "Стоп";
    if (f == true)
        timer1.Start();
    else
    {
        timer1.Stop();
        button8.Text = "Старт";
    }
    f = !f;
}
```

Пояснения: *меняем название кнопки на «Стоп». Если значение f равно «истина», то запускаем таймер. Иначе останавливаем таймер и меняем название кнопки. Значение переменной f заменяем на противоположное.*

3. Напишем метод, который для каждого тика таймера производит сдвиг и перерисовку фигуры:

```
private void timer1_Tick_1(object sender, EventArgs e)
{
    k++;
    Draw_Kv();
    Thread.Sleep(100);

}
```

Пояснения:

- данный метод должен быть создан как обработчик единственного события (Tick) объекта timer1;

- в Си# можно устанавливать задержку выполнения текущего потока на заданный интервал времени. Для этого необходимо сначала подключить Threading (using System.Threading;). Затем написать следующий код:

Thread.Sleep(100);

Время измеряется в миллисекундах.

4. Сохраните проект и запустите на исполнение. Нажмите на кнопку «Старт» - на экране должно появиться смещение квадрата. Нажмите «Стоп» - движение остановится. На экране должна появится примерно такая форма (рисунок 4.5):

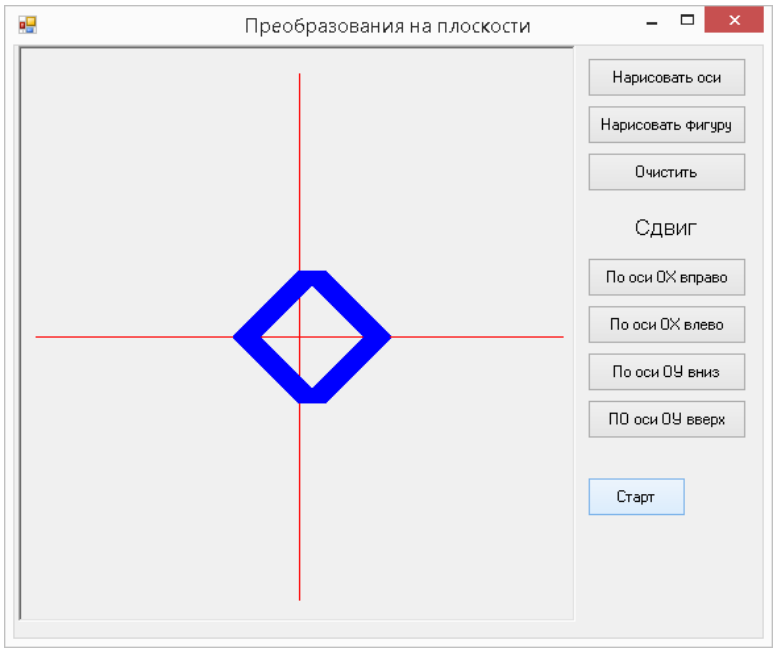


Рисунок 4.5

При повторном нажатии кнопки «Старт» движение продолжится с текущей позиции фигуры.

Задания к лабораторной работе № 3

– ***Задание 1***

Добавьте обработчик нажатия кнопки для очистки поля рисования.

– ***Задание 2***

Постройте двумерное изображение фигуры, соответствующее заданию индивидуального варианта. Начало координат должно располагаться приблизительно в центре фигуры.

– ***Задание 3***

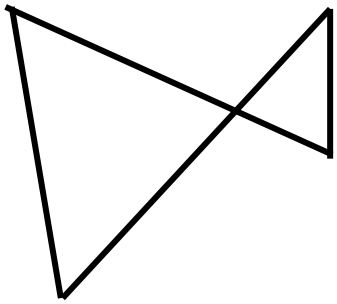
Над фигурой из задания 2 выполните все преобразования: смещение, отражение, масштабирование, поворот. Добавьте процедуры преобразования фигуры как дискретно, так и непрерывно.

– ***Задание 4***

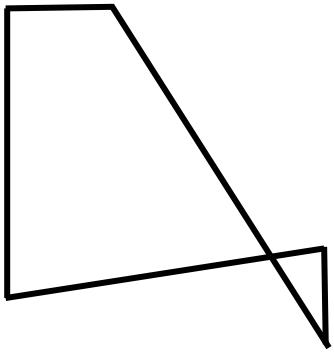
Модифицируйте все методы преобразования фигуры так, чтобы на экране

находилось только одно изображение фигуры (старых изображений фигуры не должно быть видно).

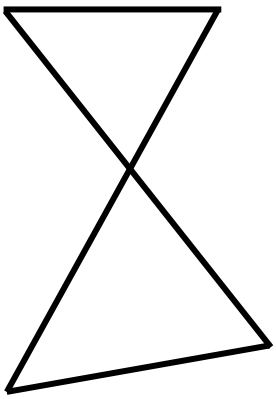
Вариант 1



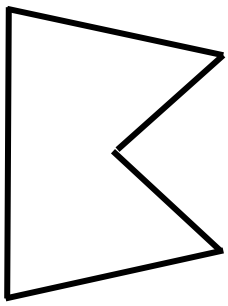
Вариант 2



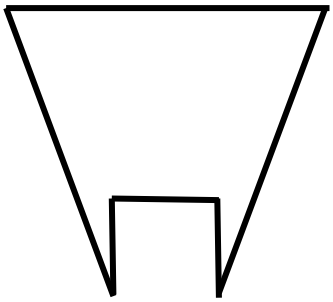
Вариант 3



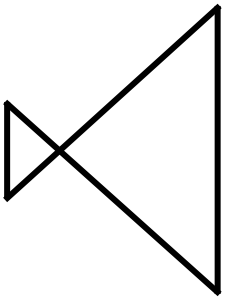
Вариант 4



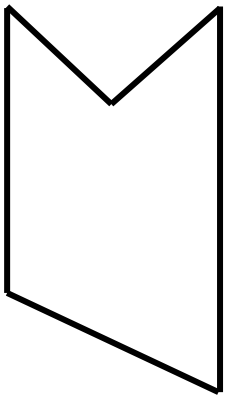
Вариант 5



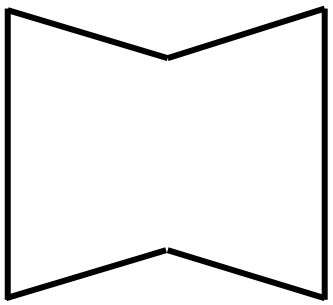
Вариант 6



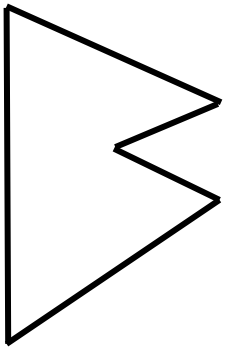
Вариант 7



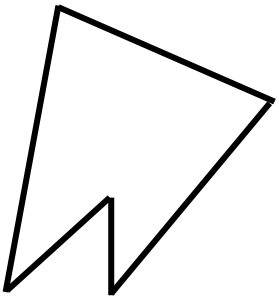
Вариант 8



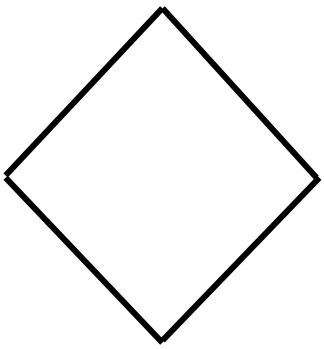
Вариант 9



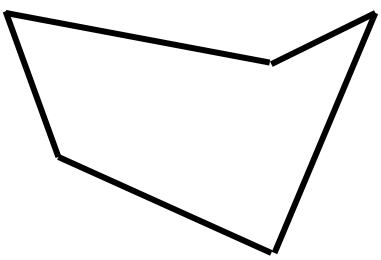
Вариант 10



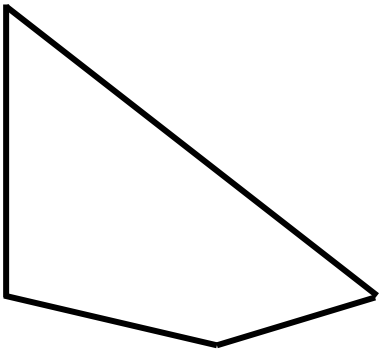
Вариант 11



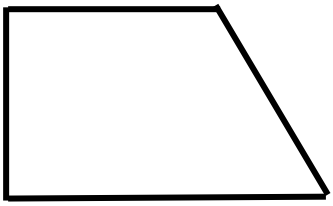
Вариант 12



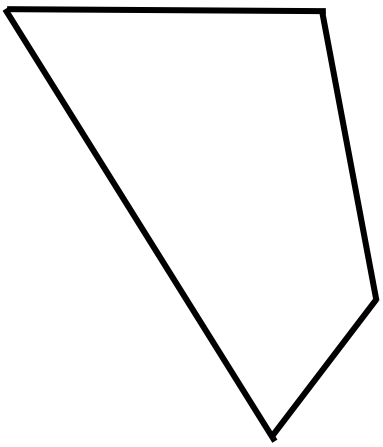
Вариант 13



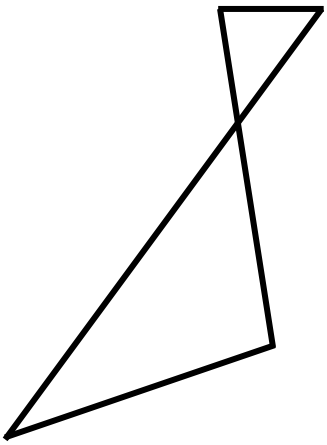
Вариант 14



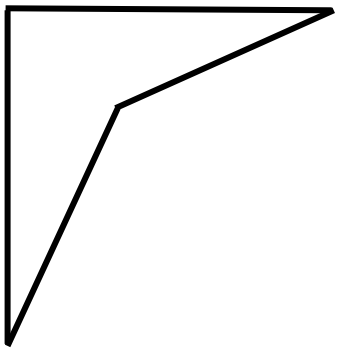
Вариант 15



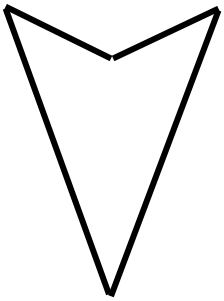
Вариант 16



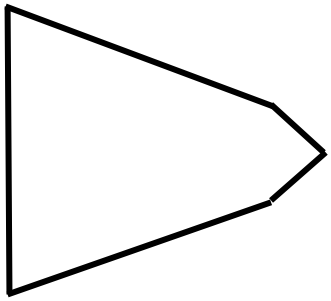
Вариант 17



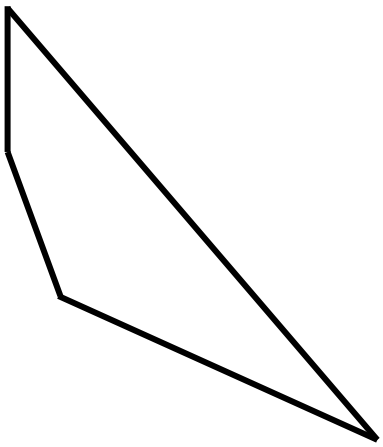
Вариант 18



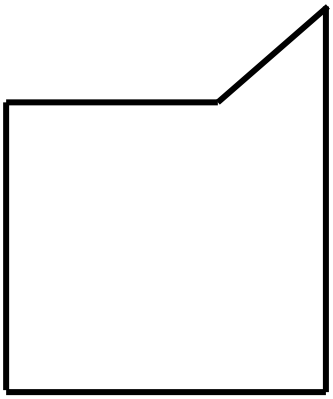
Вариант 19



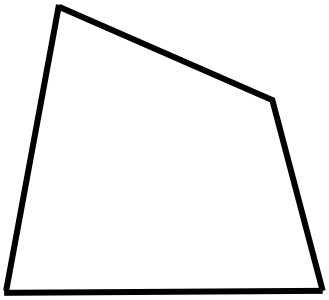
Вариант 20



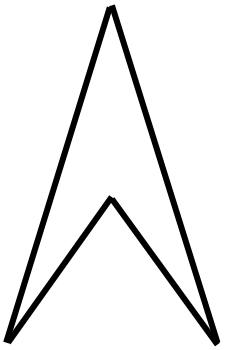
Вариант 21



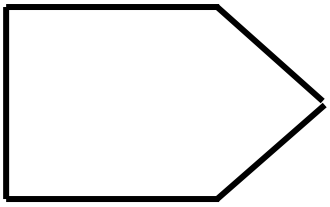
Вариант 22



Вариант 23



Вариант 24



Задания для самостоятельного выполнения к лабораторной работе № 3

– ***Общие требования к программному продукту и защите проекта***

1. При оформлении отчета по индивидуальной части лабораторной работы придумайте по два тестовых вопроса по изученным и реализованным алгоритмам. В каждом отчете должны быть представлены вопросы разного типа (закрытые вопросы с выбором одного правильного ответа, закрытые вопросы с выбором нескольких правильных ответов, открытый вопрос с вводом ответа в виде числа или строки, вопрос на установление соответствия, вопрос-эссе, вопрос на установление правильной последовательности и т.д.).

– ***Задание для руководителя***

1. Откройте проект, созданный в лабораторной работе №1. Для кнопки «Лабораторная работа №3» создайте обработчик (модуль), демонстрирующий все задания участников группы.

2. Реализуйте настройку цвета и стиля линии – толстая (предусмотрите выбор толщины и вида пера), тонкая, сплошная, пунктирная линия (предусмотрите настройку шага пунктира).

3. Реализуйте индивидуальное задание (номер варианта необходимо получить у преподавателя).

4. Подготовьтесь к выполнению лабораторной работы № 4. Для этого ознакомьтесь с теоретическим материалом по теме лабораторной работы № 4.

– ***Задание для технического писателя***

1. ТУСУРу 62 года (21.04.1962 был создани ТИРЭТ). Отобразите в проекте фразу «62 года ТУСУР» и логотип ТУСУР.

2. Реализуйте индивидуальное задание (номер варианта необходимо получить у преподавателя).

3. Напишите отчет к лабораторной работе. Требования к отчету см.п. «Методические указания по выполнению и защите лабораторных работ.

Требования к оформлению отчета».

4. Подготовьтесь к выполнению лабораторной работы № 4. Для этого ознакомьтесь с теоретическим материалом по теме лабораторной работы № 4.

– ***Задание для остальных участников группы***

1. Реализуйте индивидуальное задание (номера вариантов необходимо получить у преподавателя).

2. Подготовьтесь к выполнению лабораторной работы № 4. Для этого ознакомьтесь с теоретическим материалом по теме лабораторной работы № 4.

Варианты индивидуальных заданий на самостоятельную работу к лабораторной работе № 3

1. Написать программу, которая имитирует движение парусной лодки и полет чайки на фоне заката солнца (используя, операции масштабирования и смещения).

2. Написать программу, которая имитирует приближение бумажного самолетика, выпущенного из окна замка (используя, операции масштабирования и смещения).

3. Изобразить на экране правильный треугольник и пятиугольник, каждый из которых вращается вокруг своего центра. Фигуры расположены на расстоянии друг от друга. Начальное вращение происходит в противоположных направлениях. Предоставить возможность управления с клавиатуры скоростью и направлением вращения отдельно для треугольника и пятиугольника.

4. Изобразить на экране два вращающихся правильных треугольника разных размеров, один из которых лежит внутри другого. Начальное вращение происходит в противоположных направлениях. В процессе вращения меньший треугольник начинает расти (до заданного размера), а больший уменьшаться. Предоставить возможность управления с клавиатуры скоростью и направлением вращения отдельно для каждого треугольника.

5. Написать программу, которая имитирует движение велосипеда (вращение колес и педалей) и перемещение по экрану. Предусмотрите возможность превращать пользователю велосипед в тандем.

6. Изобразить шестиугольник, растущий из центра экрана. При этом изображение меньших размеров не стирать и каждое новое изображение выводить развернутым относительно предыдущего на угол B (угол поворота задает пользователь). Каждый шестиугольник изобразите разным цветом.

7. Изобразить восьмиугольник, уменьшающийся с краев экрана. При этом изображение больших размеров не стирать и каждое новое изображение выводить развернутым относительно предыдущего на угол B (угол поворота задает пользователь). Каждый восьмиугольник изобразите разным цветом.

8. Написать программу, демонстрирующую различные виды отображения, произвольной фигуры (фигура задается интерактивно пользователем): относительно оси X, относительно оси Y, относительно прямой X=Y. Предусмотрите два варианта отображения: 1) каждое отображение происходит, исходя из начальных координат фигуры; 2) каждое отображение происходит, исходя из текущих координат фигуры. Каждое отображение изображайте разным цветом.

9. Написать программу, которая выполняет следующие преобразования над словом, написанным прописными буквами с помощью отдельных точек: масштабирование, перемещение, вращение, растягивание по диагонали (наклон букв). Слово вводится пользователем в указанном месте экрана.

10. Написать программу для отображения полета 2-х космических кораблей вокруг Земли. Каждый корабль совершает полет по своей орбите. Орбита представляет собой окружность. При перемещении корабля на нижний край экрана, он увеличивается в размере (имитация приближения корабля). При переходе на верхний край – уменьшается (имитация удаления корабля).

11. Написать программу для отображения полета 2-х космических кораблей вокруг Земли. Корабли совершают полет на одной орбите, но с разными скоростями и в разных направлениях. Орбита представляет собой окружность. При перемещении корабля на верхний край экрана, он увеличивается в размере (имитация приближения корабля). При переходе на нижний кран – уменьшается (имитация удаления корабля).

12. Написать программу для отображения полета космического корабля вокруг центра. Центр указывается в интерактивном режиме. Маркер центра –

треугольник. Орбита представляет собой окружность. При перемещении корабля на нижний край экрана, он увеличивается в размере (имитация приближения корабля). При переходе на верхний край – уменьшается (имитация удаления корабля).

13. Написать программу, которая имитирует движение поезда (с пассажирским вагоном) с движущимися шатунами и шпалами методом перемещения фона. Задний фон имеет разные виды деревьев. В окнах вагонов происходит отображение меняющихся деревьев, имитирующее движение поезда.

14. Написать программу, которая демонстрирует вращение слова относительно точки, расположенной в центре средней буквы. Слово вводится с клавиатуры в любом месте экрана. Предусмотрите возможность задания размера шрифта и смену цвета букв слова случайным образом.

15. Написать следующую программу: в центре экрана расположена антенна (в качестве прообраза можно взять антенну – символ ТУСУР). Вокруг антенны происходит пульсация окружности с помощью операции масштабирования. Окружность должна увеличиваться в диаметре до тех пор, пока не достигнет границ экрана, затем она начинает сжиматься. Процесс должен циклически повториться, при этом необходимо обеспечить чередование цветов при увеличении и уменьшении диаметра окружности.

16. Написать программу, изображающую падающую в лужицу дождевую каплю и появляющиеся в результате этого расходящиеся круги (овалы) на поверхности воды. Предусмотреть возможность изменения количества и скорости падания капель.

Контрольные вопросы к лабораторной работе № 3

1. Что такое «однородные координаты»? Почему появилась необходимость использовать однородные координаты?
2. Опишите матрицы, используемые для 2D-преобразования.
3. Опишите процесс получения 2D-преобразований.

Приложение А

Образец титульного листа и отчета по лабораторной работе

Министерство науки и высшего образования РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Томский государственный университет систем управления и радиоэлектроники
(ТУСУР)»

Кафедра компьютерных систем в управлении и проектировании (КСУП)

ПРЕОБРАЗОВАНИЯ В ПРОСТРАНСТВЕ. ПРОЕКЦИИ. УДАЛЕНИЕ НЕВИДИМЫХ ЛИНИЙ

Отчет по лабораторной работе № 6
по дисциплине «Компьютерная графика»

Вариант –

Студенты гр.581-1
_____ И. И. Иванов
_____ И. И. Петров

«___» _____ 20__ г.

Проверил
канд. техн. наук,
доцент каф.КСУП
_____ Н. Ю. Хабибулина
«___» _____ 20__ г.

20__ г.

Цель работы и постановка задачи

Цель работы – изучение и реализация алгоритмов построения проекции фигуры, удаления невидимых линий и граней.

Задания к лабораторной работе № 6

1. Откройте проект, созданный в предыдущих лабораторных работах. Для кнопки «Лабораторная работа №6» создайте соответствующий обработчик (модуль), удовлетворяющий нижеследующим требованиям.

2. Напишите программный модуль для отображения преобразований многогранника в пространстве.

Первым действием отобразите многогранник на экране – высота многогранника параллельна вертикальной оси.

Далее реализуйте:

- построение оси вращения (при этом координаты точки, через которую проходит ось вращения, задаются пользователем интерактивно);
- поворот многогранника так, чтобы его высота совпала с заданной осью вращения;
- вращение относительно высоты многогранника, совпадающей с заданной осью вращения;
- перемещение многогранника (по каждой координатной оси, по двум или трем осям одновременно);
- изменение размера многогранника (по каждой оси, по двум или трем координатным осям одновременно);
- настройку цвета и толщины линий;
- изменение направления и скорости вращения / перемещения.

3. Нарисуйте систему координат и ось вращения.

4. Координаты точек – мировые.

Максимальное и минимальное значение мировых координат изобразите на экране. Осуществите автоматический подбор размеров изображения.

5. При выполнении этого задания при удалении невидимых линий программа должна выдавать контурное изображение (невидимые линии рисовать штрих-пунктиром).

6. Отображаемую фигуру, ось вращения, вид проекции и алгоритм удаления невидимых линий выберите в соответствии с вариантом (таблица 3):

№	Фигура	Проекция	Ось вращения	Алгоритм удаления невидимых линий
	Тетраэдр	Изометрия	Прямая, заданная произвольными направляющими косинусами	алгоритм, использующий нормаль грани

Анализ задачи

Для выполнения поставленной задачи необходимо реализовать следующие алгоритмы:

1. Переход от мировых координат к экранным координатам.
2. Трехмерные сдвиг и масштабирование.
3. Трёхмерное вращение вокруг произвольной оси.
4. Изометрическое проецирование трёхмерного изображения на плоскость
5. Удаление невидимых линий (прорисовка пунктиром).

1. Алгоритм перехода от мировых координат к экранным координатам.

Для получения 3D-преобразований воспользуемся векторно-полигональной моделью фигуры. В соответствии с ней любая фигура в трехмерном пространстве может быть представлена матрицей координат своих вершин. Затем, используя данную матрицу, проводят вектора, которые соединяют соответствующие вершины. В результате получают каркасное изображение фигуры. Далее, используя алгоритмы удаления невидимых линий, получают изображение фигуры с отсечением невидимых ребер.

Изначально любая трехмерная фигура задается мировыми однородными координатами. Для ее отображения на экране необходимо получить экранные координаты. Для получения экранных координат используют следующую цепочку преобразований:

мировые координаты (x, y, z) -> видовые координаты $(x_в, y_в, z_в)$ *-> координаты проекций* $(x_{пр}, y_{пр}, z_{пр})$ *-> экранные координаты* $(x_Э, y_Э, z_Э)$.

2. Преобразования сдвига и масштабирования в однородных координатах относительно центра координат имеют одинаковую форму произведения вектора исходных координат вершины фигуры на матрицу преобразования.

Преобразование сдвига.

Преобразование сдвига выглядит следующим образом:

$$v \cdot T = (x\ y\ z\ h) \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ l & m & n & 1 \end{pmatrix},$$

где v – вектор трехмерных однородных координат вершины фигуры, h – произвольный множитель не равный нулю (в нашем случае 1), T – матрица сдвига, l, m, n – величины смещения по осям OX, OY, OZ соответственно.

Преобразование масштабирование.

Преобразование сдвига выглядит следующим образом:

$$v \cdot T = (x\ y\ z\ h) \cdot \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & e & 0 & 0 \\ 0 & 0 & j & 0 \\ 0 & 0 & 0 & s \end{pmatrix},$$

где v – вектор трехмерных однородных координат вершины фигуры, , T – матрица масштабирования: a, e, j – величины масштабирования по осям OX, OY, OZ соответственно; s – полное изменение масштаба (в данном случае a, e, j должны быть равны 1).

3. Трёхмерное вращение вокруг произвольной оси.

Преобразование поворота вокруг произвольной оси, заданной направляющими косинусами и проходящей через заданную точку, выглядит следующим образом:

$$v \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -l & -m & -n & 1 \end{pmatrix} \cdot R \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ l & m & n & 1 \end{pmatrix},$$

где v – вектор трехмерных однородных координат вершины многогранника,

l, m, n – заданные координаты точки, через которую проходит ось вращения,

R – матрица поворота вокруг оси, заданной направляющими косинусами:

$$R := \begin{bmatrix} (n_1)^2 + (1 - n_1)^2 \cos(\theta) & n_1 \cdot n_2 (1 - \cos(\theta)) + n_3 \sin(\theta) & n_1 \cdot n_3 (1 - \cos(\theta)) & 0 \\ n_1 \cdot n_2 \cdot (1 - \cos(\theta)) - n_3 \cdot \sin(\theta) & (n_2)^2 + (1 - n_2)^2 \cdot \cos(\theta) & n_2 \cdot n_3 (1 - \cos(\theta)) + n_1 \cdot \sin(\theta) & 0 \\ n_1 \cdot n_3 \cdot (1 - \cos(\theta)) + n_2 \cdot \sin(\theta) & n_2 \cdot n_3 \cdot (1 - \cos(\theta)) - n_1 \cdot \sin(\theta) & (n_3)^2 + (1 - n_3)^2 \cdot \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

где n_1, n_2, n_3 – направляющие косинусы относительно осей OX, OY и OZ соответственно, Θ – угол поворота.

Положительным направлением считается направление против часовой стрелки, если смотреть вдоль оси вращения к началу координат.

4. Проецирование изображения.

В данной лабораторной работе применяется изометрическая проекция. Матрица проецирования имеет вид:

$$pro := \begin{pmatrix} -\cos\left(\frac{\pi}{6}\right) & \sin\left(\frac{\pi}{6}\right) & 0 & 0 \\ \cos\left(\frac{\pi}{6}\right) & \sin\left(\frac{\pi}{6}\right) & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Проецирование осуществляется умножением вектора трехмерных однородных координат на матрицу проецирования. Оси системы координат расположены под углом 120 градусов относительно друг друга.

5. Для построения более реалистичного изображения трёхмерных сцен необходимо удалять невидимые части объектов (рёбра и грани).

Существует 2 основных подхода к решению данной задачи.

Первый подход заключается в непосредственном сравнении объектов друг с другом для выяснения того, какие части объектов являются видимыми. В данном случае работа ведётся в пространстве объектов.

Второй подход заключается в определении для каждого пикселя экрана ближайшего к нему объекта (вдоль направления проецирования). При этом работа ведётся в пространстве экранных координат.

Рассмотрим один из алгоритмов, реализующий первый подход.

Отсечение нелицевых граней.

Пусть для каждой грани некоторой фигуры задан единичный вектор внешней нормали. Если вектор нормали грани составляет с направлением проецирования (направлением взгляда на объект) тупой угол, то такая грань не может быть видна и называется нелицевой. В случае, когда данный угол является острым, грань видна и называется лицевой.

В случае, когда трёхмерная сцена представляет собой один выпуклый многогранник, удаление нелицевых граней полностью решает задачу удаления невидимых граней.

В общем случае описанная проверка не решает задачу полностью, но позволяет значительно сократить количество рассматриваемых граней.

Алгоритм удаления нелицевых плоскостей, используемый в данной работе.

1. Сформировать многоугольники граней, исходя из списка вершин тела.

2. Вычислить нормальный вектор для каждой полигональной грани тела.

3. Определить нелицевые плоскости:

Определить косинус угла между нормалью к грани и вектором направления наблюдателя.

Если этот косинус угла меньше 0, то плоскость невидима.

Вывести весь многоугольник, лежащий в этой плоскости пунктиром.

4. Вывести остальные грани.

Описание структуры и алгоритма программы

Разработанная программа имеет структуру, изображенную на рисунке 1.

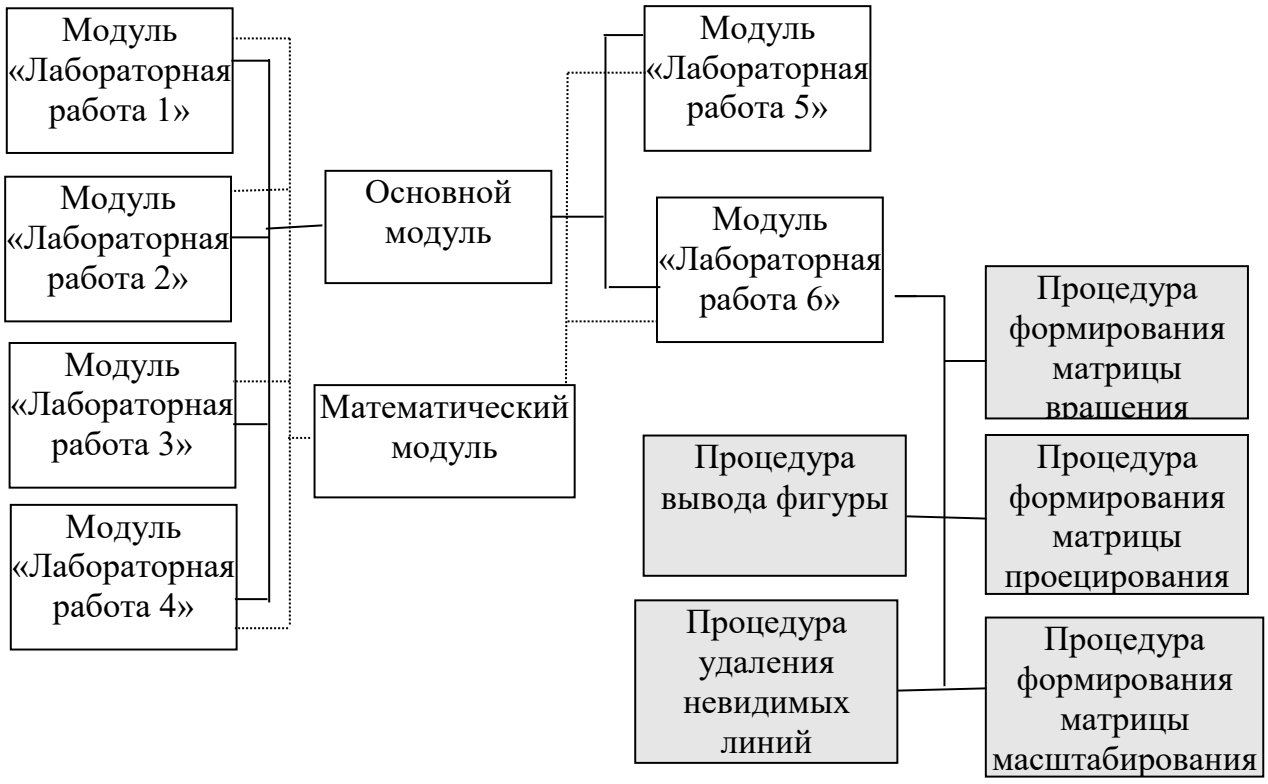


Рисунок 1

Разработанный в данной лабораторной работе модуль имеет имя lab6.pas. Его форма lab6.dfm.

Основной алгоритм программного модуля:

1. Формируем матрицу мировых координат фигуры

2. Формируем матрицы преобразования.

3. Организуем цикл поворота фигуры:

3.1 очистка экрана;

3.2 умножаем матрицу фигуры на матрицу смещения;

3.3 умножаем матрицу фигуры на матрицу поворота;

3.4 умножаем матрицу фигуры на матрицу обратного смещения;

3.5 умножаем матрицу фигуры на матрицу проецирования;

3.6 определяем видимость ребер и выводим фигуру красным цветом с новыми координатами;

3.7 увеличиваем угол поворота;

3.8 переходим на шаг 3.

Описание основных процедур и функций

Модуль состоит из нескольких процедур:

. . .

(описание процедур – название, входные, выходные параметры, ее назначение и подробное пошаговое описание основных процедур)

procedure FormCreate(Sender: TObject); - процедура для создания формы.

procedure Draw(A:TMatrix; f:boolean); – в зависимости от значения f выводит грань пунктирной или сплошной линией.

procedure ImageClear; – очистка экрана.

function VectorMul(A:TMatrix):Tmatrix; – функция возвращает нормаль к грани.

function ScalarMul(Vector:TMatrix):real; – функция определяет косинус угла между нормалью и направлением проекцирования.

procedure DrawXY(A:TMatrix); – выводит оси на экран.

Пошаговое описание основной процедуры Draw

Шаг 1. Выделяем память под массивы граней фигуры, нормалей к этим граням, матрицу переноса, матрицу поворота фигуры вокруг оси, матрицу изометрической проекции;

Шаг 2.Задаем начальные значения для всех этих матриц;

Шаг 3.Начальный угол поворота равен 0;

Шаг 4.Цикл пока не нажата кнопка «Стоп»;

Шаг 5. Повернуть 1-ю грань фигуры на угол Phi

Шаг 6. Определить нормаль к этой грани

Шаг 7. Определить угол между нормалью и вектором проецирования

Шаг 8. Если угол тупой, грань не лицевая f1=false, иначе лицевая f1= true

Шаг 9. Повернуть 2-ю грань фигуры на угол Phi

Шаг 10. Определить нормаль к этой грани

Шаг 11. Определить угол между нормалью и вектором проецирования

Шаг 12. Если угол тупой, грань не лицевая f2=false, иначе лицевая, f2= true

Шаг 13. Повернуть 3-ю грань фигуры на угол Phi

Шаг 14. Определить нормаль к этой грани

Шаг 15. Определить угол между нормалью и вектором проецирования

Шаг 16. Если угол тупой, грань не лицевая f3=false, иначе лицевая f3=true

Шаг 17. Повернуть 4-ю грань фигуры на угол Phi

Шаг 18. Определить нормаль к этой грани

Шаг 19. Определить угол между нормалью и вектором проецирования

Шаг 20. Если угол тупой, грань не лицевая f4=false, иначе лицевая f4=true

Шаг 21. Если 1-ая грань не лицевая – нарисовать пунктиром, иначе ничего не выводить

Шаг 22. Если 2-ая грань не лицевая – нарисовать пунктиром, иначе ничего не выводить

Шаг 23. Если 3-ая грань не лицевая – нарисовать пунктиром, иначе ничего не выводить

Шаг 24. Если 4-ая грань не лицевая – нарисовать пунктиром, иначе ничего не выводить

Шаг 25. Если 1-ая грань лицевая – нарисовать сплошной линией, иначе ничего не выводить

Шаг 26. Если 2-ая грань лицевая – нарисовать сплошной линией, иначе ничего не выводить

Шаг 27. Если 3-ая грань лицевая – нарисовать сплошной линией, иначе ничего не выводить

Шаг 28. Если 4-ая грань лицевая – нарисовать сплошной линией, иначе ничего не выводить

Шаг 29. Увеличить угол вращения

Шаг 30. Обновить матрицу вращения

Шаг 31. Кнопка «Стоп» нажата? Да выход, Иначе переход на Шаг 5.

Руководство пользователя

Исполняемый файл программы – labb.exe. После запуска программы появляется основная форма (рисунок 2).

. . .

Для запуска лабораторной работы 6 нажмите кнопку «Лабораторная работа7». Появиться форма (рисунок 3).

. . .

Для поворота фигуры нажмите кнопку «Поворот». Для изменения скорости вращения передвиньте ползунок на полосе прокрутки (рисунок 4).

. . .

Для выхода из программы закройте форму.

Ответы на контрольные вопросы

…….

Тестовые вопросы

…….

Заключение

В ходе проделанной работы изучили алгоритмы получения 3D-преобразований фигуры, а именно: алгоритм формирования векторно-полигональной модели фигуры, алгоритм перехода от мировых координат к экранным, алгоритмы вращения, смещения, масштабирования в пространстве, алгоритм проецирования, алгоритм удаления невидимых линий.

В результате создан программный продукт, реализующий все изученные алгоритмы и предоставляющий следующие возможности:

- отображение трехмерной фигуры на экране;
- задание оси вращения;
- вращение фигуры вокруг заданной оси;
- перемещение и масштабирование фигуры по каждой координатной оси, по двум или трем осям одновременно;
- удаление невидимых ребер фигуры;
- настройку скорости и направления вращения / перемещения;
- настройку цвета и стиля линий.