Лабораторная работа №9

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ МЕХАТРОНИКИ ТРАНСПОРТНЫХ СРЕДСТВ

Кафедра информатики

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по проведению практических работ по дисциплине «Программирование» для студентов специальности 6.050201 “Системная инженерия”

Разработчик - доцент кафедры информатики
кандидат технических наук,
старший научный сотрудник
Тимонин Владимир Алексеевич

Харків 2012

Лабораторная работа №9
Исследование возможностей интегрированной среды разработки Visual C# для создания анимационных приложений с использованием изображений.

Цель работы – исследовать возможностей интегрированной среды разработки Visual C# и получить практические навыки по созданию анимационных приложений с использованием изображений.

1. Теоретические сведения

Движение фрагментов изображения можно осуществ лять следующими способами:

1. Рисовать изображение различными цветами переднего плана (пера), а за тем, спустя время, стирать его - т.е. выводить то же самое изображение цветом фона. Затем вновь размещать это же изображение в несколько сме щенном положении и стирать его после задержки на экране. Траектория движения определяется программой.

2. Выводить изображение прежним способом, но стирать его различными фигурами: прямоугольником, эллипсом, сектором и др. Цвет контура и за ливки таких фигур совпадает с цветом фона, габариты фигур несколько превышают габариты изображения.

3. Изображение подвижного фрагмента записать в буфер, который размеща ется в динамической памяти. Затем это изображение выводить из буфера последовательно в разные места экрана. При этом, перед очередном выво дом, изображение в предшествующем месте (после задержки) стирать.

1.1. Методика разработки приложения «Стендовая стрельба»

Мультипликацию с использованием графических примитивов изучим на приме ре приложения «Стрельба стендовая», в котором подвижное изображение создаётся пер выми двумя способами. Во всех способах мультипликации фрагмент изображения выводится каким-либо цветом или даже разными цветами, а затем, через небольшой интервал времени, этот фрагмент стирается. Далее вывод и сти рание фрагмента производятся в несколько смещённом положении. Траектория движения может быть произвольной.

Последовательность действий такова: стрела красного цвета (появляю щаяся слева экрана) летит горизонтально в сторону мишени, расположенной в правой части окна. После попадания стрелы в мишень вылетает другая, зелёная, стрела (несколько ниже прежней), она также вонзается в мишень. Далее всё по вторяется для стрел других цветов. В конечном итоге в мишени будут тор чать 9 стрел разного цвета (рис. 1).

Приложение содержит несколько пользовательских методов Метод **Mishen()** выводит мишень: концентрические эллипсы сиреневого цвета, распо ложенные в правой части экрана, их вертикальная ось превышает горизонтальную в 2 раза.

Метод **Strela(int ix, int idy)** рисует простейшее изображение стрелы. Пара метры ix и idy изменяют местоположение стрелы соответственно по горизонтали и вертикали.

Метод **Pysk_Strelu(int idy, Color Cvet_Strelu)** запускает стрелу заданного цвета (Cvet_Strelu) с выбранной высотой полёта (устанавливается аргументом idy относительно центра экрана по вертикали Y_Centr). Изменением формального параметра ix в методе **Strela()** обеспечивается горизонтальное перемещение стре лы. Скорость такого перемещения регулируется методом **Sleep()** при помощи её единственного аргумента, измеряемого в миллисекундах. Эта функция задержи вает полёт стрелы на заданный интервал времени.

Рис. 1.

Метод **Zapysk_Strl()** осуществляет последовательный запуск всех стрел на ряде высот и с разной окраской.

Пример 1. Ниже приводится код приложения, в котором реализуется первый способ мультипликации.

```
namespace Strelba_Stendovaja
{
   public partial class Form1 : Form
   {
      int X_Centr, Y_Centr;
      Color Cvet;
      public Form1()
      {
         InitializeComponent();
      }
      void Mishen()
      {
         Graphics im = pictureBox1.CreateGraphics();
         int n= 5, rx= pictureBox1.Size.Width/60; //Радиус наименьшего кольца
         X_Centr = (int)(pictureBox1.Size.Width * 0.9); //Центр мишени по оси x
         Y_Centr = pictureBox1.Size.Height / 2; //Центр мишени по оси y
         System.Drawing.Pen PenUser; // Карандаш пользователя
         PenUser = new System.Drawing.Pen(Color.Purple, 1);
         for (int i= n; i>=1; i--)
            im.DrawEllipse(PenUser, X_Centr - rx*i, Y_Centr - rx*i*2, rx*i*2, rx*i*4);
      }
      void Zapysk_Strl()
      {
         int ihy= Y_Centr/6; // Расстояние между стрелами по высоте
         for (int i= -4, idy; i<=4; i++)
           {
```

```
            switch (i)
                {
                 case -4: Cvet= Color.Red; break;
                 case -3: Cvet = Color.Green; break;
                 case -2: Cvet = Color.Blue; break;
                 case -1: Cvet = Color.Yellow; break;
                 case 0: Cvet = Color.Purple; break;
                 case 1: Cvet = Color.Olive; break;
                 case 2: Cvet = Color.Maroon; break;
                 case 3: Cvet = Color.Navy; break;
                 case 4: Cvet = Color.Teal; break;
                }
             idy = ihy * i; // idy - смещение стрелы от Y_Centr
             Pysk_Strelu(idy, Cvet);
           }
      }
      void Pysk_Strelu(int idy, Color Cvet_Strelu)
      {
         int dl_Strelu = (int)(pictureBox1.Size.Width * 0.1); //Длина стрелы
         for (int ii = 0, ix; ii < 8; ii++)
         {
            ix = dl_Strelu + ii * dl_Strelu;
            Strela(Cvet_Strelu, ix, idy);
            Thread.Sleep(100); // Задержка выполнения программы, измеряется в мс
            Strela(Color.White, ix, idy);
            if (ii == 7) Strela(Cvet_Strelu, ix, idy);
         }
      }
      void Strela(Color Cvet_Strelu, int ix, int idy)
      {
         Graphics im = pictureBox1.CreateGraphics();
         int dl_Strelu = (int)(pictureBox1.Size.Width * 0.1); // Длина стрелы
         System.Drawing.Pen PenUser; // Карандаш пользователя
         PenUser = new System.Drawing.Pen(Cvet_Strelu, 3);
         im.DrawLine(PenUser, ix-18, Y_Centr+idy, ix+dl_Strelu, Y_Centr+idy); //Тело
                                                                              // стрелы
         im.DrawLine(PenUser, ix - 15, Y_Centr + idy - 5, ix, Y_Centr + idy); //Верхнее перо
         im.DrawLine(PenUser, ix - 15, Y_Centr + idy + 5, ix, Y_Centr + idy); //Нижнее перо
      }
      private void button1_Click(object sender, EventArgs e)
      {
         Mishen();
         Zapysk_Strl();
      }
   }
}
```

Пример 2. Приложение, использующее второй способ разработки мультфильмов, от личается лишь методом **Pysk_Strelu()**, её код приведен ниже.

```
void Pysk_Strelu(int idy, Color Cvet_Strelu)
```

```
    {
      Graphics im = pictureBox1.CreateGraphics();
      int dl_Strelu = (int)(pictureBox1.Size.Width * 0.1); //Длина стрелы
      for (int ii = 0, ix; ii < 8; ii++)
      {
        ix = dl_Strelu + ii * dl_Strelu;
        if (ii < 7)
        {
          Strela(Cvet_Strelu, ix, idy);
          Thread.Sleep(100); //Задержка выполнения программы, измеряется в мс
          Rectangle uRect = new Rectangle(ix - 18, Y_Centr + idy - 7, dl_Strelu, 14);
          im.FillRectangle(Brushes.White, uRect);
        }
        else Strela(Cvet_Strelu, ix, idy); // Стрела вонзается в мишень
      }
    }
```

1.2. Разработка приложения «Самолет»

Для формирования динамической картинки используются несколько вариантов создания анимации:

- «классический», то есть предполагается наличие заранее подготовленной серии картинок (кадров), последовательное отображение которых и создает эффект анимации;
- картинка (кадр) формируется из заранее подготовленных фрагментов «на лету» во время работы программы.

Типичным примером такой анимации является перемещение объекта на фоне какой-либо картинки. Чтобы у наблюдателя сложилось впечатление, что объ ект движется, надо вывести изображение объекта, затем, через некоторое время, стереть его и снова вывести, но уже на некотором расстоянии от пер воначального положения. Подбором времени между удалением и выводом изображения, а также расстояния между новым и предыдущим положением объекта (шага перемещения) можно добиться эффекта равномерного дви жения.

Пример 3. Приложение "Самолет" (исходный текст представлен ниже, а результат на рис. 2) демонстрирует принципы создания анимации "на лету", по казывает, как заставить объект двигаться.

```
public partial class Form1 : Form
  {
    Bitmap sky;     // рисунок "небо"
    Bitmap plane; // рисунок "самолет"
    Graphics g;     // графическая поверхность, на которой будем формировать рисунок
    int dx;             // приращение координаты X, определяет скорость полета
    Rectangle rct; // область, в которой находится объект
    Random rnd;  // генератор случайных чисел

    public Form1()
    {
      InitializeComponent();
      try
      {
        sky = new  System.Drawing.Bitmap(Application.StartupPath + "\\sky.bmp");
        plane = new  System.Drawing.Bitmap(Application.StartupPath + "\\plane.bmp");
      }
      catch (Exception e)
```

```
        {
            MessageBox.Show("Ошибка загрузки битового образа: " + e.Message,
                            "Полет", MessageBoxButtons.OK,MessageBoxIcon.Error);
            this.Paint += null;
            return;
        }
        plane.MakeTransparent(); // сделать прозрачным фон
        this.ClientSize = sky.Size;   // установить размер формы равным размеру
                                      // фонового рисунка
        this.BackgroundImage = new Bitmap(sky); // задать фоновый рисунок формы
        g = this.CreateGraphics(); // определяем графическую поверхность
        rnd = new Random();        // инициализация генератора случайных чисел
        // Исходное положение самолета
        rct.X = -40;
        rct.Y = this.Size.Height/3 + rnd.Next(50);
        rct.Width = plane.Width;
        rct.Height = plane.Height;
        // Скорость полета определяется периодом следования сигналов от таймера
        // и величиной приращения координаты X
        dx = 2; // скорость полета - 2 пикселя/тик
        timer1.Interval = 20;
        timer1.Enabled = true;
    }
    private void timer1_Tick(object sender, EventArgs e)
    {
        // Стереть изображение объекта - вывести фрагмент фона в ту область
         // графической поверхности, в которой сейчас находится объект
        Rectangle Rect = new Rectangle(rct.X, rct.Y, rct.Width, rct.Height);
        g.DrawImage(sky, Rect, Rect, GraphicsUnit.Pixel);
        // Вычислить новое положение объекта
        if (rct.X < this.ClientRectangle.Width)
          rct.X += dx;
        else
            {
              // объект достиг правой границы, перемещаем его к левой границе
              rct.X = -40;
              rct.Y = this.Size.Height/3 + rnd.Next(50);
              // скорость полета от 2 до 5 пикселей/тик
              dx = 2 + rnd.Next(4);
            }
        // вывести изображение объекта на новом месте
        g.DrawImage(plane, rct.X, rct.Y);
    }
}
```

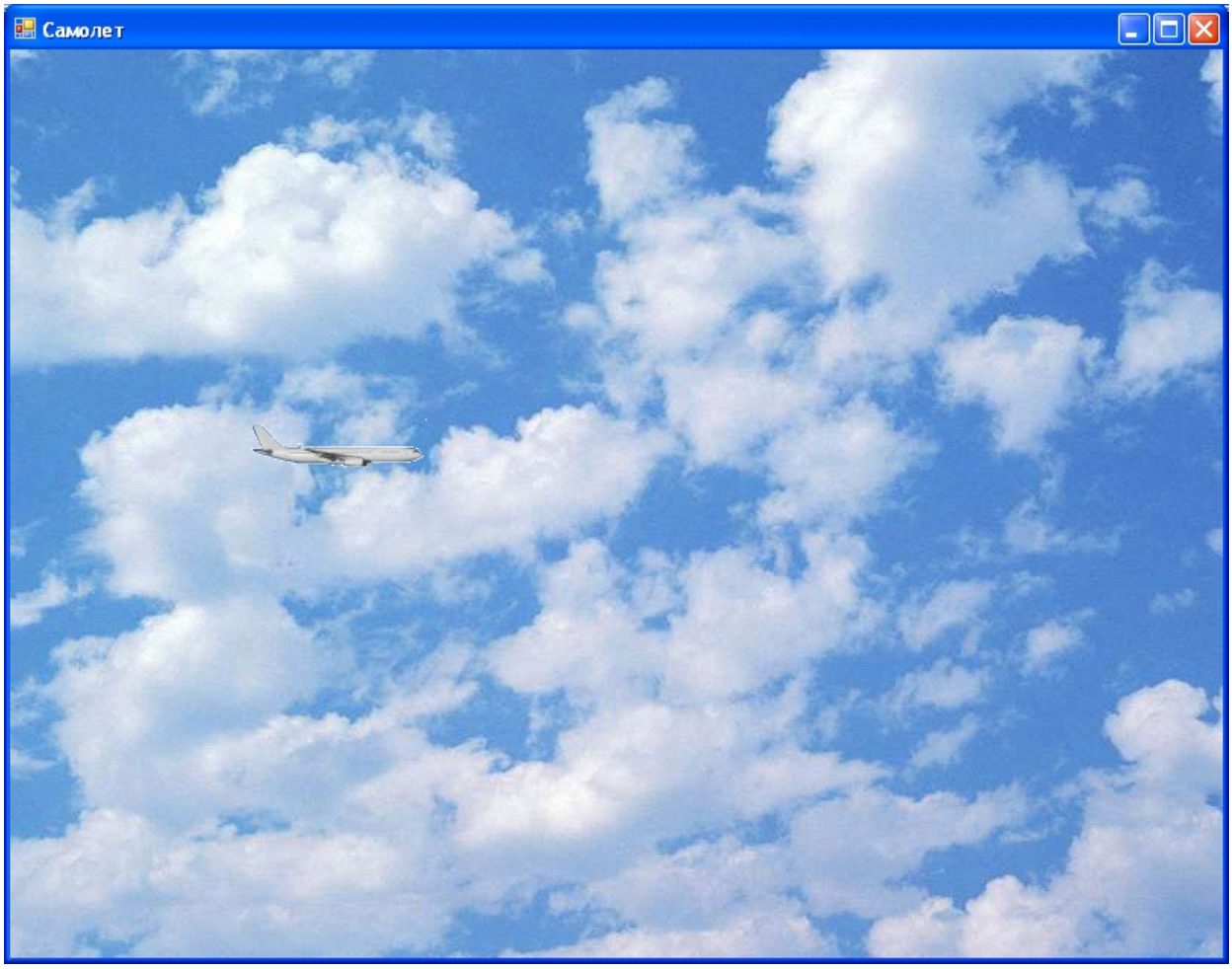

Рис. 2. Результат выполнения приложения«Самолет»

2. Рабочее задание

Задание 1. Модернизируйте приложение «Стрельба стендовая» таким образом, чтобы в качестве механизма задержки программы использовался компонент **Timer**.

Задание 2. Модернизируйте приложение «Стрельба стендовая» (первый способ мультипликации) таким образом, чтобы стрела пролетала сквозь мишень и оставляла в ней отверстие, совпадающее по цвету с цветов стрелы.

Задание 3. Модернизируйте приложение «Стрельба стендовая» таким образом, чтобы стрела пролетала сквозь мишень, и оставляла в ней отверстие красно го цвета. Перед вылетом очередной стрелы отверстие от предшествующей стрелы должно исчезать. Для восстановления изображения мишени (испорченного про лётом стрелы) используйте не метод **Mishen()**, а методы **Save()** и **Restore()** класса **Graphics**.

Задание 4. Модернизируйте приложение «Самолет» таким образом, чтобы:

а) в небе было несколько самолетов;

б) самолеты могли двигаться навстречу друг другу;

в) одиночный самолет делал «мертвую петлю».

Выбор варианта работы приложения должен осуществляться с помощью элементов управления, предоставляемых пользователю.

3. Контрольные вопросы

Литература

1. Голощапов А.Л. Microsoft Visual Studio 2010. – СПб.:БХВ-Петербург, 2011. – 544 с.: ил.
2. Культин Н.Б. Microsoft Visual C# в задачах и примерах. – СПб.: БХВ-Петербург, 2009. – 320 с.: ил.
3. Лабор В.В. Си Шарп: Создание приложений для Windows. – Мн.: Харвест, 2003. – 384 с.
4. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 1. Пер. с англ. - М.: «Русская Редакция», 2002.- 576 с.: ил.
5. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 2. Пер. с англ. - М.: «Русская Редакция», 2002.- 624 с.: ил.
6. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4.0. Пер. с англ. - М.: Издательский дом "Вильямс", 2011. — 1392 с.: ил.
7. Фаронов В.В. Программирование на языке C#. – СПб.: Питер, 2007. – 240 с.: ил.
8. Фленов М.Е. Библия C#. - СПб.: БХВ-Петербург, 2011.– 560с.: ил.