Лабораторная работа №5

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 №5
Исследование возможностей интегрированной среды разработки Visual C# для создания приложений по обработке событий мыши.

Цель работы – исследовать возможности интегрированной среды разработки Visual Studio 2010 и получить практические навыки по созданию приложений по обработке событий мыши.

1. Теоретические сведения

Все действия пользователя при взаимодействии с приложением сводятся к перемещению мыши, нажатию кнопок мыши и нажатию клавиш клавиатуры.

1.1. События мыши

Получение и обработка ввода мыши — важная часть каждого приложения ОС Windows. Можно обработать события мыши для выполнения действий в приложении или использовать положение указателя мыши для проверки нажатия или других действий. Кроме того, можно изменить способ обработки ввода мыши элементами управления в приложении. Основным способом реагирования на ввод мыши является обработка событий мыши. В компонентах Visual C# определен ряд событий, связанных с мышью, приведенный в табл.1.

Таблица 1. Перечень событий мыши

Событие	Описание
Click	Это событие происходит при отпускании кнопки мыши, обычно перед событием **MouseUp**. Обработчик этого события получает аргумент типа **EventArgs**. Данное событие обрабатывается при необходимости определить, когда происходит щелчок.
MouseClick	Это событие возникает при щелчке элемента управления мышью. Обработчик этого события получает аргумент типа **MouseEventArgs**. Данное событие обрабатывается при необходимости получить сведения о мыши в момент щелчка
DoubleClick	Это событие возникает при двойном щелчке элемента управления. Обработчик этого события получает аргумент типа **EventArgs**. Данное событие обрабатывается при необходимости определить, когда происходит двойной щелчок.
MouseDoubleClick	Это событие возникает при двойном щелчке по элементу управления мышью. Обработчик этого события получает аргумент типа **MouseEventArgs**. Данное событие обрабатывается при необходимости получить сведения о мыши в момент двойного щелчка.
MouseDown	Это событие происходит, если пользователь нажимает кнопку мыши в тот момент, когда указатель мыши находится над элементом управления Обработчик этого события получает аргумент типа **MouseEventArgs**.
MouseMove	Это событие возникает при перемещении указателя мыши над элементом управления. Обработчик этого события получает аргумент типа **MouseEventArgs**.
MouseUp	Это событие происходит, если пользователь отпускает кнопку мыши в тот момент, когда указатель мыши находится над элементом управления Обработчик этого события получает аргумент типа **MouseEventArgs**.
MouseEnter	Это событие происходит, когда указатель мыши входит в рамку или клиентскую область элемента управления, в зависимости от типа элемента управления. Обработчик этого события получает аргумент типа **EventArgs**.
MouseHover	Это событие происходит, когда указатель мыши останавливается над элементом управления. Обработчик этого события получает аргумент типа **EventArgs**.
MouseLeave	Это событие происходит, когда указатель мыши покидает рамку или клиентскую

	область элемента управления, в зависимости от типа элемента управления. Обработчик этого события получает аргумент типа **EventArgs**.
DragDrop	Это событие происходит по завершении операции перетаскивания.
DragEnter	Это событие происходит при перетаскивании объекта в пределы элемента управления.
DragLeave	Это событие происходит при перетаскивании объекта за пределы элемента управления.
DragOver	Это событие происходит, когда объект перетаскивается через границу элемента управления.
MouseWheel	Это событие происходит, когда пользователь прокручивает колесико мыши, в то время как на элементе управления установлен фокус. Обработчик этого события получает аргумент типа **MouseEventArgs**. Для определения, насколько прокручена мышь, можно использовать свойство **Delta** аргумента **MouseEventArgs**.

Событие **Click** передает объект **EventArgs** его обработчику событий, указывая, что щелчок был выполнен. Если необходимы более точные сведения о мыши (кнопка, количество щелчков, вращение колесика или положение), следует использовать событие **MouseClick**.

Двойной щелчок определяется параметрами мыши в операционной системе пользователя. Пользователь может задать время между нажатиями кнопки мыши, которые будут считаться двойным щелчком, а не двумя отдельными щелчками. Событие **Click** вызывается каждый раз, когда на элементе управления происходит двойной щелчок. Например, при наличии обработчиков для событий **Click** и **DoubleClick** объекта **Form** события **Click** и **DoubleClick** вызываются, когда на форме происходит двойной щелчок. Если элемент управления при двойном щелчке не поддерживает событие **DoubleClick**, событие **Click** может быть вызвано дважды.

Если требуется обрабатывать события щелчка мыши в определенном порядке, необходимо знать порядок, в котором возникают события щелчка в элементах управления Windows Forms. Все элементы управления Windows Forms, кроме отдельных элементов, вызывают события щелчка в одном и том же порядке после того, как была нажата и отпущена кнопка мыши (вне зависимости от того, какая кнопка).

Порядок событий, вызываемых после одиночного щелчка мыши:
событие **MouseDown** → событие **Click** → событие **MouseClick** → событие **MouseUp**.

Порядок событий, возникающих после двойного щелчка мыши:
событие **MouseDown** → событие **Click** → событие **MouseClick** → событие **MouseUp** → событие **MouseDown** → событие **DoubleClick** → событие **MouseDoubleClick** → событие **MouseUp**.

Для элементов управления **Button, CheckBox, ComboBox, RadioButton** возникают события:

- щелчок левой кнопкой мыши: **Click** → **MouseClick**;
- щелчок правой кнопкой мыши: событие щелчка не вызывается;
- двойной щелчок левой кнопкой мыши: **Click** → **MouseClick**; **Click** → **MouseClick**;
- двойной щелчок правой кнопкой мыши: событие щелчка не вызывается.

Для элементов управления **TextBox, RichTextBox, ListBox, MaskedTextBox, CheckedListBox** возникают события:

- щелчок левой кнопкой мыши: **Click** → **MouseClick**;
- щелчок правой кнопкой мыши: событие щелчка не вызывается;
- двойной щелчок левой кнопкой мыши: **Click** → **MouseClick**→**DoubleClick** → **MouseDoubleClick**;
- двойной щелчок правой кнопкой мыши: событие щелчка не вызывается.

Для элементов управления **ListView, TreeView** возникают события:

- щелчок левой или правой кнопкой мыши: **Click** → **MouseClick**;
- двойной щелчок левой или правой кнопкой мыши: **Click** → **MouseClick**→**DoubleClick** → **MouseDoubleClick**;

1.2. Распознавание источника события, нажатых кнопок, координат курсора

Во все обработчики событий, связанных с манипуляциями мыши (как и во все другие обработчики) передается параметр **sender** типа указателя на **object**. Этот параметр содержит указатель на компонент, в котором произошло событие. Он не требуется, если пишется обработчик события для одного конкретного компонента

Так как часто один обработчик применяется для нескольких компонентов, и при этом какие-то операции могут быть общими для любых источников события, а какие-то требовать специфических действий, то **sender** можно использовать для распознания источника события.

Кроме параметра **sender** в обработчики событий **MouseClick, MouseDoubleClick, MouseDown, MouseMove, MouseUp, MouseWheel** передаются параметры, позволяющие распознать нажатую кнопку, определить координаты курсора мыши. Заголовок обработчика этих событий имеет, например, следующий вид:

```
private void textBox1_MouseDown(object sender, MouseEventArgs e)
    {
    }
```

Объект **MouseEventArgs** указывает, какая кнопка мыши нажата, сколько раз была нажата и отпущена кнопка мыши, координаты мыши и расстояние, на которое переместилось колесико мыши.

Объект типа **MouseEventArgs** имеет следующие свойства:

- **Button** – содержит значение, показывающее, какая кнопка мыши была нажата. Имеет тип **MouseButtons**, определённый в пространстве имен System.Windows.Forms как:

public enum MouseButtons.

Значениями перечисления **MouseButtons** являются:

Left - левая кнопка мыши;
Middle - средняя кнопка мыши;
Right - правая кнопка мыши;
XButton1 - первая расширенная кнопка мыши;
XButton2 - вторая расширенная кнопка мыши.

- **Clicks** – содержит значение, указывающее, сколько раз была нажата и отпущена кнопка мыши.
- **Delta** - содержит значение со знаком, указывающее количество делений, на которое повернулось колесико мыши, умноженное на константу **WHEEL_DELTA**. Делением называется один зубец колесика мыши. Одно деление колеса определено константой ОС Windows **WHEEL_DELTA**, которая равна 120. Положительное значение показывает, что колесико вращается вперед (от пользователя), а отрицательное значение — назад (к пользователю).
- **Location** – содержит координаты месторасположение указателя мыши в момент создания события мыши. Имеет тип **Point**, определённый в пространстве имен System.Windows.Forms. Объект **Point**, содержащий координаты x и y указателя мыши (в пикселях) относительно левого верхнего угла формы.
- **X** - содержит координатух указателя мыши в момент создания события мыши;
- **Y** - содержит координатуу указателя мыши в момент создания события мыши.

Пример 1. Нижеприведенный фрагмент кода, используя перечисление MouseButtons, демонстрирует возможность определения того, какая кнопка нажата. В зависимости от того, какая кнопка нажата, выдается соответствующее сообщение:

```
private void Form1_MouseDown(object sender, MouseEventArgs e)
    {
      switch (e.Button)
      {
        case MouseButtons.Left:
          textBox1.Text = "Нажата левая кнопка";
          break;
```

```
            case MouseButtons.Middle:
               textBox1.Text = "Нажата средняя кнопка";
               break;
            case MouseButtons.Right:
               textBox1.Text = "Нажата правая кнопка";
               break;
         }
      }
```

Пример 2. Нижеприведенный фрагмент кода, используя свойства Location, демонстрирует отслеживание нажатия левой кнопки мыши, и в ответ на нажатие кнопки мыши рисуется последовательность сплошных линий.

```
   private void Form1_MouseDown(object sender, MouseEventArgs e)
      {
         Point firstPoint = e.Location;
          firstPoint.X += 20;
          firstPoint.Y -= 20;
         Graphics g = this.CreateGraphics();
         g.DrawLine(Pens.Black, firstPoint, e.Location);
      }
```

Пример 3. Нижеприведенный код приложения, демонстрирует возможность рисования мышью на поверхности формы.

Процесс рисования начинается с помещения указателя мыши на специально отведенную область рисования (в данном случае – это форма) и нажатия кнопки мыши. После нажатия кнопки мышь перемещается в другое место, оставляя за собой след, позволяющий контролировать движения мыши. При отпускании кнопки мыши рисование прекращается. При нажатии правой кнопки мыши форма очищается от рисунков. Один из вариантов результата выполнения приложения представлен на рис. 1.

```
int x_md, y_md; // координаты мыши в момент нажатия
Pen p = new Pen(Color.Red, 3); // создаем перо красного цвета и толщиной 3 пикселя
bool bPaint; // признак окончания процесса рисования
private void Form1_MouseDown(object sender, MouseEventArgs e)
      {
         bPaint = true;
         x_md = e.X;
         y_md = e.Y;
      }
private void Form1_MouseMove(object sender, MouseEventArgs e)
      {
         if (bPaint)
          {
           Graphics g = CreateGraphics();
           int x_mm = e.X;
           int y_mm = e.Y;
            g.DrawLine(p, x_md, y_md, x_mm, y_mm);
            // сохраняем координаты мыши
           x_md = x_mm;
           y_md = y_mm;
           g.Dispose(); // освобождаем ресурсы класса Graphics
```

```
            }
        }
    private void Form1_MouseUp(object sender, MouseEventArgs e)
        {
          bPaint = false;
        }
    private void Form1_MouseClick(object sender, MouseEventArgs e)
        {
          if (e.Button == MouseButtons.Right)
            {
              Graphics g = CreateGraphics();
              g.Clear(this.BackColor);
              g.Dispose();
            }
        }
```

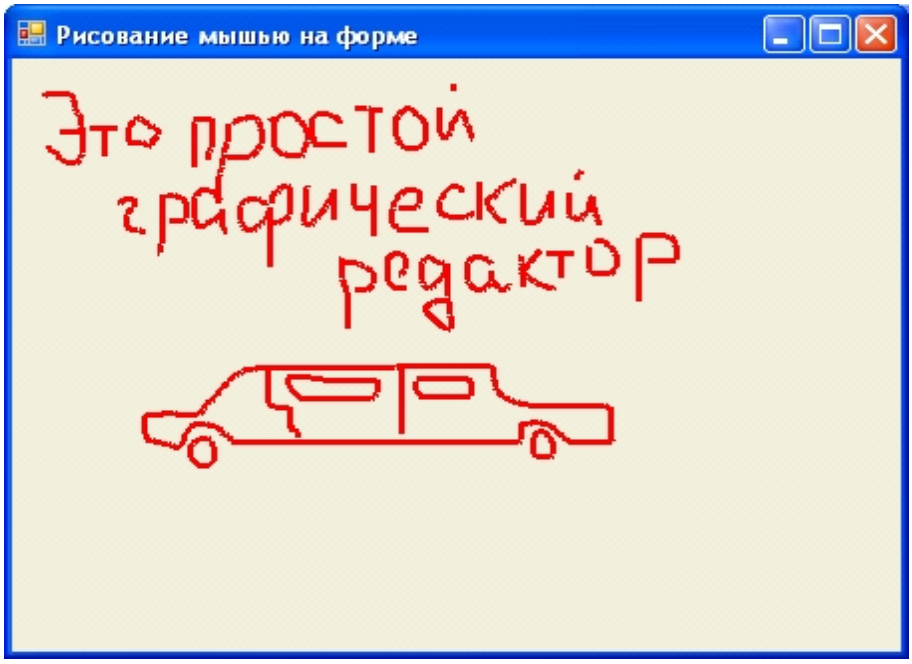

Рис. 1. Результат выполнения приложения«Рисование мышью на форме»

2. Рабочее задание

Задание 1. Разработать приложение «Движение мыши над объектами», с помощью которого отслеживается движение мыши по экрану.

При разработке интерфейса приложения использовать различные компоненты, например, **TextBox, ComboBox, PictureBox** и др. (один из вариантов интерфейса представлен на рис. 2).

Рис. 2. Внешний вид приложения «Движение мыши над объектами»

Задание 2. Разработать приложение «Прямоугольники с координатами», с помощью которого можно рисовать прямоугольники с отображением координат верхнего левого и правого нижнего углов.

При разработке интерфейса приложения использовать компоненты, на которых можно рисовать (**Form, PictureBox** и др.), **Button** (один из вариантов интерфейса представлен на рис. 3).

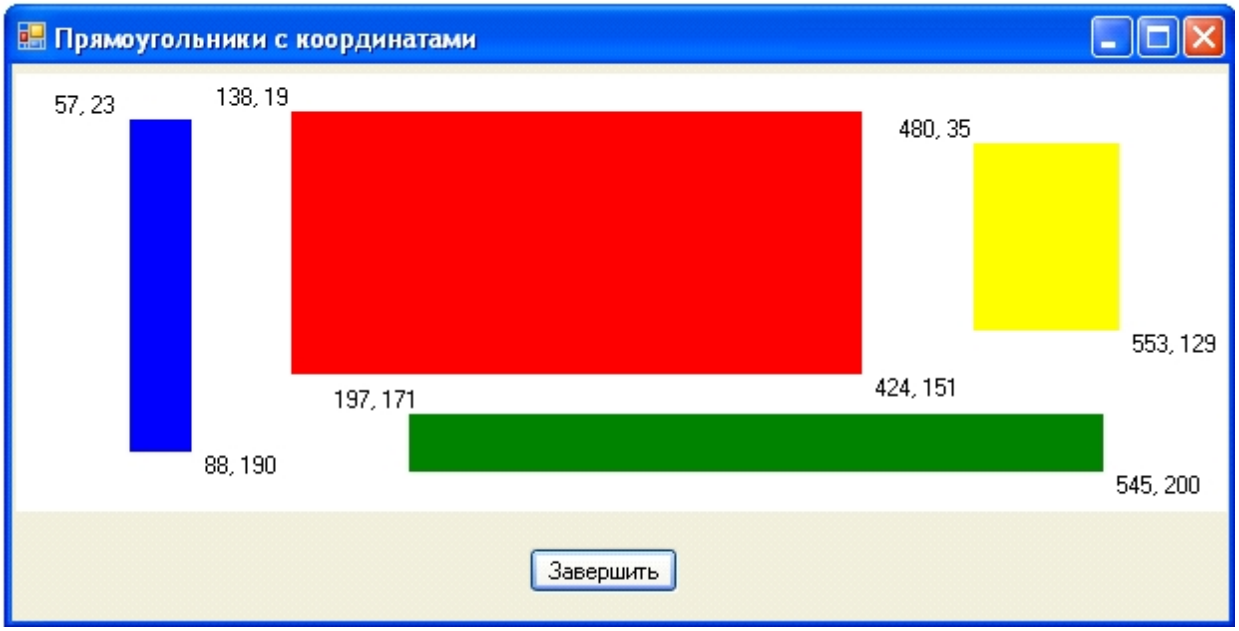

Рис. 3. Внешний вид приложения «Прямоугольники с координатами»

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.