5

Ввод информации

Мышь, перо и клавиатура

Взаимодействие с программой пользователь осуществляет с помощью пера или аппаратных кнопок на самом устройстве. А где же мышь и клавиатура? Ну, предположим, что клавиатуру можно заменить ее виртуальным аналогом на экране КПК. Она имеет практически ту же функциональность, что и настоящая клавиатура. Кроме того, следует помнить, что существуют устройства с настоящей, хоть и маленькой клавиатурой. Что же касается мыши, то ее роль с успехом выполняет перо, которое иногда еще называют стилусом.

Но стоит заметить, что у пера нет возможности эмулировать нажатие правой кнопки мыши. Когда пользователь применяет перо, то генерируются события `MouseDown`, `MouseMove`, `MouseUp` и `Click`. Первые три события могут сообщить о позиции курсора, как и события из настольной версии Windows.

Отдельно надо отметить, что смартфоны не имеют сенсорного экрана и не могут использовать перо. А роль клавиатуры выполняют цифровые клавиши набора номера и несколько дополнительных кнопок. Следовательно, логика работы программы на смартфонах должна отличаться от работы приложения на устройствах с сенсорным экраном. Для сравнения можно привести пример игры Solitaire, которая входит в состав Windows Mobile. Вы можете запустить эту игру на эмуляторах и сравнить управление в этой игре на устройствах разного типа. На устройствах с сенсорным экраном перетаскивание карт из колоды производится пером и данное действие очень похоже на перетаскивание карты с помощью мыши в этой же игре на настольном компьютере. Совсем по-другому построен интерфейс игры на смартфоне. Все стопки карт пронумерованы и соответствуют клавишам телефона. И пользователю остается только нажимать клавиши с соответствующими цифрами.

Курсоры

Так как пользователь при работе использует перо, то Windows Mobile не отображает на экране устройства стандартную стрелку курсора. Предполагается, что пользователь может самостоятельно попасть острым концом пера в маленькую кнопку или другой элемент. Но у мобильных систем курсоры все же есть. Первый из них является аналогом песочных часов в настольной версии Windows и выглядит как анимированный круг с разноцветными секторами. Второй курсор можно увидеть при вызове контекстного меню. Он выглядит как множество красных маленьких кружков, которые постепенно появляются вдоль воображаемой окружности.

Песочные часы

При выполнении длительных ресурсоемких операций нужно показать пользователю, что устройство работает, а не зависло. Лучше всего вывести на экран устройства курсор ожидания. В карманных компьютерах в качестве такого курсора используются не песочные часы, как в настольных компьютерах, а анимированный разноцветный круг. Установить данный тип курсора в приложении очень просто, что иллюстрирует фрагмент кода, приведенный в листинге 5.1.

Листинг 5.1

```
// Устанавливаем курсор ожидания
Cursor.Current = Cursors.WaitCursor;

// возвращаем курсор по умолчанию
Cursor.Current = Cursors.Default;
```

На рис. 5.1. показано приложение с соответствующим курсором.

Рис. 5.1. Отображение курсора ожидания

Обработка события Tap-and-Hold

Так как в карманных компьютерах не используется правая кнопка мыши, то для вызова контекстного меню используется операция Tap-and-Hold. Пользователь касается пером поверхности экрана и некоторое время удерживает нажатие. Если элемент, на поверхности которого находится перо, связан с элементом `ContexMenu`, то на экране появится контекстное меню. А что делать, если мы хотим создать собственный обработчик события Tap-and-Hold? В этом случае надо добавить в проект таймер и написать код для обработки событий `Mouse_Down`, `Mouse_Up` и `timer1_Tick`. Для таймера следует задать интервал, необходимый для инициирования события. Сам код приведен в листинге 5.2.

Листинг 5.2

```
private void Form1_MouseDown(object sender, MouseEventArgs e)
{
   // включаем таймер
   timer1.Enabled = true;
}

private void Form1_MouseUp(object sender, MouseEventArgs e)
{
   timer1.Enabled = false;
   label1.Text = "";
}

private void timer1_Tick(object sender, EventArgs e)
{
   label1.Text = "Вы нажали на экран";
}
```

ПРИМЕЧАНИЕ

Примеры CursorsAndKeys.

Эмуляция Tap

Когда пользователь касается пером сенсорного экрана, происходит так называемое событие Tap, которое сравнимо с событием Click мыши на обычных компьютерах. В некоторых случаях может понадобиться программно воспроизвести это событие для решения различных задач. Для этого нам понадобится вызвать функцию Windows API mouse_event (листинг 5.3).

Листинг 5.3

```
using System.Runtime.InteropServices;

[DllImport("coredll")]
private static extern void mouse_event(MOUSEEVENTF dwFlags,
int dx, int dy, int dwData, UIntPtr dwExtraInfo);

[Flags()]
private enum MOUSEEVENTF
{
    MOUSEEVENTF_MOVE = 0x1,
    MOUSEEVENTF_LEFTDOWN = 0x2,
    MOUSEEVENTF_LEFTUP = 0x4,
    MOUSEEVENTF_ABSOLUTE = 0x8000,
}

private void menuItemClickStart_Click(object sender, EventArgs e)
{
    // Перемещаем перо в верхний левый угол экрана
    // и нажимаем пером на экран (Tap)
    // Тем самым мы нажимаем кнопку Пуск и вызываем меню этой кнопки
    mouse_event(MOUSEEVENTF.MOUSEEVENTF_MOVE | MOUSEEVENTF.MOUSEEVENTF_ABSOLUTE,
        2, 2, 0, UIntPtr.Zero);
    // Нажимаем и отпускаем
    mouse_event(MOUSEEVENTF.MOUSEEVENTF_LEFTDOWN, 0, 0, 0, UIntPtr.Zero);
    mouse_event(MOUSEEVENTF.MOUSEEVENTF_LEFTUP, 0, 0, 0, UIntPtr.Zero);
}
```

Для наглядности примера мы перемещаем воображаемый курсор мыши в верхнюю часть экрана с координатами 2,2 и эмулируем нажатие пера в этой точке. Так как в левом верхнем углу экрана находится кнопка Пуск, то программное нажатие в этой точке вызывает меню этой кнопки.

ПРИМЕЧАНИЕ

Пример MouseEventDemo.

Добавляем функциональность

На данный момент .NET Compact Framework пока не поддерживает методы OnEnter и OnLeave для элементов. Решение проблемы нашлось на сайте MSDN. Так как метод OnMouseMove под-

держивается, мы можем воспользоваться им в сочетании с методом Capture для определения, когда указатель мыши входит или выходит за пределы элемента управления. Для демонстрации мы создадим простой элемент управления MouseCapture, который будет окрашиваться синим цветом при движении пера внутри элемента и серым цветом при выходе за пределы элемента. Выход за пределы элемента определяется при помощи свойства ClientRectangle и метода OnMouseMove. Для начала создаем новый класс, MouseCapture (листинг 5.4).

Листинг 5.4

```
public class MouseCapture : Control
{
    public MouseCapture()
    {
        this.BackColor = Color.LightGray;
    }

    // Если перо над элементом, то Capture равно true
    protected override void OnMouseMove(MouseEventArgs e)
    {
        this.Capture = this.ClientRectangle.Contains(e.X, e.Y);
        if (this.Capture == true)
            this.BackColor = Color.Blue;
        else
            this.BackColor = Color.LightGray;
    }
}
```

Теперь осталось добавить код в событие Load основной формы (листинг 5.5).

Листинг 5.5

```
private void Form1_Load(object sender, EventArgs e)
{
    MouseCapture mc = new MouseCapture();
    mc.Parent = this;
    mc.Bounds = new Rectangle(50, 50, 140, 150);
}
```

Обратите внимание, что изменение цвета будет происходить при условии, если вы будете перемещать перо, как это вы обычно делаете при операции перетаскивания.

ПРИМЕЧАНИЕ

Пример AddedFunctionality.

InputPanel

На большинстве мобильных устройств нет стандартной клавиатуры, поэтому ввод текста осуществляется с помощью виртуальной клавиатуры SIP. В Visual Studio 2008 клавиатура SIP представлена элементом управления InputPanel, о котором мы немного поговорили в главе 3. Рассмотрим работу с этим элементом подробнее. Создадим новый проект и добавим в проект элемент InputPanel. Вы уже наверняка видели, что данная виртуальная клавиатура имеет несколько предопределенных режимов ввода информации. Предположим, мы хотим программно переключиться на режим распознавания символов (листинг 5.6).

Листинг 5.6

```
private void button1_Click(object sender, EventArgs e)
{
    foreach (InputMethod m in inputPanel.InputMethods)
    {
        if (m.Name == "Letter Recognizer")
            inputPanel.CurrentInputMethod = m;
    }
    inputPanel1.Enabled = true;
}
```

Пример прекрасно работает в эмуляторе, но на самом деле в коде есть потенциальная ошибка. На реальном устройстве вполне может не оказаться метода `Letter Recognizer`. Именно так обстоит дело на моем реальном устройстве. Таким образом, нам необходимо сначала получить список имеющихся методов панели ввода. Но сначала мы получим число имеющихся методов (листинг 5.7).

Листинг 5.7

```
private void button2_Click(object sender, EventArgs e)
{
    MessageBox.Show(inputPanel.InputMethods.Count.ToString());
}
```

Свойство `InputPanel.InputMethodCollection.Count` возвращает число методов у клавиатуры. Обычно их три:

- `Keyboard`;
- `Letter Recognizer`;
- `Block Recognizer`.

Но тем не менее мы не должны уповать на то, что нужный нам метод имеется на устройстве. Мы можем получить реальный список методов (листинг 5.8).

Листинг 5.8

```
/// <summary>
/// Получаем коллекцию методов ввода
/// </summary>
private void GetInputMethods()
{
    this.listBox1.Items.Clear();

    // Получаем коллекцию InputMethods
    // и каждый метод выводим в список
    foreach (InputMethod im in inputPanel.InputMethods)
    {
        this.listBox1.Items.Add(im.Name);
    }
}
```

ПРИМЕЧАНИЕ

Пример InputPanelDemo.

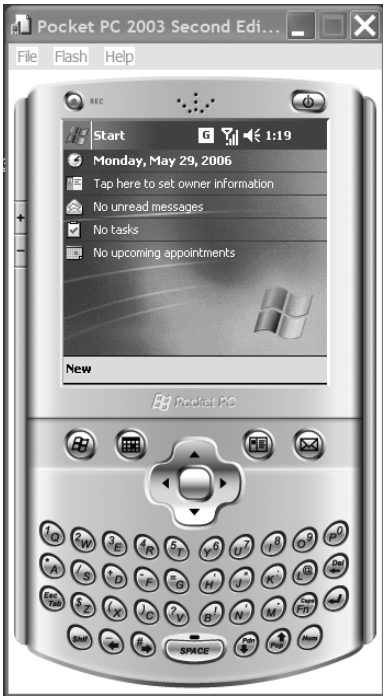

Рис. 5.2. Эмулятор устройства с клавиатурой

Рис. 5.3. Обработка нажатий клавиш навигации

Клавиатура

В последнее время стали появляться устройства с настоящей встроенной клавиатурой. Как правило, эти устройства имеют квадратный экран. Среда разработки поддерживает эмуляторы подобных моделей, например эмулятор Pocket PC 2003 SE (рис. 5.2).

Кроме того, на устройствах имеются клавиши навигации, клавиша Enter и кнопки запуска определенных приложений. Все эти клавиши могут обрабатывать стандартные события.

Клавиши навигации

Если в процессе создания приложения в режиме работы с формой щелкнуть мышью на любой из кнопок навигации, то среда разработки сгенерирует код для этих кнопок в событии `Form_KeyDown`. В листинге 5.9 приведен пример обработчика этого события.

Листинг 5.9

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
   if ((e.KeyCode == System.Windows.Forms.Keys.Up))
   {
      label1.Text = "Клавиша Вверх";
   }
```

```
   if ((e.KeyCode == System.Windows.Forms.Keys.Down))
   {
      label1.Text = "Клавиша Вниз";
   }
   if ((e.KeyCode == System.Windows.Forms.Keys.Left))
   {
      label1.Text = "Клавиша Влево";
   }
   if ((e.KeyCode == System.Windows.Forms.Keys.Right))
   {
      label1.Text = "Клавиша Вправо";
   }
   if ((e.KeyCode == System.Windows.Forms.Keys.Enter))
   {
      label1.Text = "Клавиша Enter";
   }
}
```

Как видите, приложение определяет нажатую клавишу при помощи перечисления `System.Windows.Forms.Keys`. Если открыть виртуальную клавиатуру и нажать клавиши со стрелками, то можно убедиться, что они тоже инициируют событие `Form_KeyDown` (рис. 5.3). Если протестировать пример на устройстве с настоящей клавиатурой, то можно заметить, что приложение правильно обрабатывает нажатие встроенных клавиш со стрелками.

Нажатия специальных кнопок

На мобильных устройствах имеются различные функциональные кнопки. Например, есть отдельная кнопка выключения устройства. На самом деле при нажатии этой кнопки устройство не выключается, а переходит в особый спящий режим. В мобильных устройствах программы и данные хранятся в памяти, и если устройство действительно выключить, то все приложения и данные просто пропадут. Разработчик может программно перевести устройство в спящий режим, имитируя нажатие этой кнопки выключения с помощью функции API `keybd_event`, как показано в листинге 5.10.

Листинг 5.10

```
/// <summary>
/// Функция имитирует нажатия клавиш на клавиатуре
/// </summary>
/// <param name="bVk">Виртуальный код клавиши для имитации
/// нажатия и отпускания клавиши</param>
/// <param name="bScan">Зарезервировано - установлено в
/// 0</param>
/// <param name="dwFlags">Флаг</param>
/// <param name="dwExtraInfo">Дополнительная информация</param>
 [DllImport("coredll.dll", CharSet = CharSet.Unicode)]
public static extern void
keybd_event(byte bVk, byte bScan, int dwFlags, int dwExtraInfo);
// константа для кнопки выключения устройства
public const int VK_OEM_8 = 0xDF;
private void butOff_Click(object sender, EventArgs e)
```

продолжение ⇩

Листинг 5.10 (продолжение)

```
{
   // Имитируем нажатие кнопки выключения устройства
   keybd_event(VK_OEM_8, 0, 0, 0);
}
```

Раз уж мы заговорили о функциональных клавишах, то рассмотрим еще один пример. На устройствах Windows Mobile Professional имеется кнопка с изображением красного телефона, с помощью которого можно прервать телефонный разговор. Эта кнопка соответствует клавише F4. Таким образом, если вам необходимо программно нажать данную клавишу в приложении, то пишем следующий код с использованием той же функции keybd_event (листинг 5.11).

Листинг 5.11

```
[DllImport("coredll.dll", CharSet = CharSet.Unicode)]
public static extern void
keybd_event(byte bVk, byte bScan, int dwFlags, int dwExtraInfo);

// константа для кнопки End
public const int VK_F4  =  0x73;

// Флаг отпускания клавиши
public const int KEYEVENTF_KEYUP = 0x0002;

private void butKeyEnd_Click(object sender, EventArgs e)
{
    // имитируем нажатие и отпускание кнопки End
    keybd_event(VK_F4, 0, 0, 0);
    keybd_event(VK_F4, 0, KEYEVENTF_KEYUP, 0);
}
```

ПРИМЕЧАНИЕ

Пример KeysPress.

Разработка приложений без использования пера

В этой главе мы рассмотрели несколько примеров использования пера и клавиатуры в приложениях. Но важно понимать, как использовать устройства ввода информации в своих программах. В последнее время Microsoft стала уделять этому вопросу повышенное внимание, пытаясь разработать определенные стандарты в этом вопросе. Сейчас прослеживается четкая тенденция стирания граней между смартфонами и карманными компьютерами. Потребители все больше склоняются к выбору гибридных устройств, имеющих возможности телефона и компьютера. По новой классификации от Microsoft такие устройства относятся к группе Windows Mobile Professional. В таких устройствах основной приоритет отдается мобильной телефонии. Для удобства пользователя работа с телефоном должна быть доступна пальцами одной руки. Это стало особенно актуально с появлением очень модного телефона iPhone, в котором управление ведется именно пальцами на сенсорном экране. Представьте себе человека в офисе, который одновременно

держит телефон в одной руке и ручку в другой, делая необходимые пометки. Понятно, что такой пользователь не будет доставать перо, чтобы нажать кнопку ответа на телефонный звонок или кнопку отбоя. Таким образом, перед разработчиком стоит задача разработки приложений, которыми можно управлять с помощью одной руки. Естественно, в данном случае мы не рассматриваем случаи разработки игр и других программ, где действуют свои правила. Однако вернемся к нашей теме.

Основные принципы

Итак, в современных устройствах под управлением Windows Mobile присутствуют две клавиши Soft Key 1 и Soft Key 2. К этим клавишам привязываются элементы меню. Пользователь должен иметь возможность управлять программой одной рукой, перемещаясь от одного элемента управления к другому при помощи еще одной клавиши, Action, которая находится в центре устройства. Если устройство имеет клавиатуру, то пользователь должен иметь возможность пользоваться клавишей Tab с такой же функциональностью. Кроме того, желательно, чтобы пользователь также мог нажать на элемент, имеющий фокус, своим пальцем вместо пера — нужно позаботиться об удобных размерах элементов управления.

Теперь рассмотрим таблицу, в которой собраны основные операции при работе с различными клавишами (табл. 5.1).

Таблица 5.1. Поведение клавиш Action и клавиш клавиатуры

Тип клавиши	Название клавиши	Описание
Action Key	Righ	Перемещает фокус на следующий элемент. Если элемент имеет редактируемый текст, то перемещает каретку в этом тексте
Action Key	Left	Перемещает фокус на предыдущий элемент в tab-order. Если элемент содержит редактируемый текст, то перемещает каретку в этом тексте
Action Key	Down	Перемещает фокус на следующий элемент в tab-order. Если элемент содержит список значений, то перемещает выделение по списку (ComboBox и ListBox)
Action Key	Up	Перемещает фокус на предыдущий элемент в tab-order. Если элемент содержит список значений, то перемещает выделение по списку (combo box и list)
Action Key	Center	Активирует событие Click. Если элемент имеет выпадающий список, то данное событие должно разворачивать или сворачивать его (combo box)
Keyboard	Tab	Перемещает фокус на следующий элемент
Keyboard	Shift+Tab	Перемещает фокус на предыдущий элемент
Keyboard	Enter	Активирует событие Click

Рассмотрим эти правила подробнее. Создадим пример, иллюстрирующий наши правила. Проверьте, что передача фокуса от элемента к элементу идет в правильном порядке. Для этого надо проверить значение `TabIndex` у каждого элемента. Более удобный способ —

включить режим показа порядка перехода фокуса (меню View | Tab Order). В этом случае вы сможете визуально просматривать и редактировать порядок передачи фокуса. Возможны ситуации, когда не требуется, чтобы элемент управления получал фокус. В этом случае присвойте его свойству `TabStop` значение `False`. В принципе, наши первые приготовления не сильно отличаются от принципов дизайна формы для настольных приложений. Использование клавиши Tab на клавиатуре или на панели ввода SIP не представляет трудностей. Далее рассмотрим вопросы применения клавиш навигации (Action Key). Большинство мобильных устройств не имеет встроенной клавиатуры, чтобы пользоваться клавишей Tab. Использовать кнопку Tab на виртуальной клавиатуре — значит, доставать перо и отказаться от затеи создания программ для одной руки. Но на всех устройствах Windows Mobile имеются кнопки навигации с пятью действиями: вверх, вниз, влево, вправо, Enter.

Если вы попробуете запустить проект и понажимать на клавиши навигации, то увидите, что не всегда переход фокуса происходит как задумывалось. Здесь требуется определенная доработка.

Очень хорошая статья «Developing Stylus-Free Windows Mobile Professional Applications» на данную тему имеется на сайте MSDN по адресу http://msdn2.microsoft.com/en-us/library/bb985500.aspx.

Дополнительные материалы

Если вы хотите узнать о клавиатуре еще больше, то стоит обратить внимание на блог Алекса Яхнина, который можно найти по адресу blog.opennetcf.org/ayakhnin. Там можно найти статью «Keyboard hook in the CF v2». В данной статье рассказывается о перехвате всех сообщений, которые посылаются при нажатии любых кнопок устройства. Также может быть полезна статья «Custom SIP Control for CF». Автор статьи предлагает собственную реализацию элемента `InputControl`, который содержит свою виртуальную клавиатуру. Этот пример может пригодиться при создании приложения, в котором не используется стандартная панель ввода SIP, но при этом пользователь должен иметь возможность ввода информации.