Александр Климов

Занимательное программирование на Visual Basic .NET

Санкт-Петербург
«БХВ-Петербург»
2005

УДК 681.3.068+800.92VB.NET
ББК 32.973.26-018.1
К49

Климов А. П.

К49 Занимательное программирование на Visual Basic. NET. — СПб.: БХВ-Петербург, 2005. — 528 с.: ил.

ISBN 5-94157-572-6

На занимательных примерах рассмотрено программирование на языке Visual Basic .NET. Описана работа с текстами и строками: от создания бегущих строк и мигающих заголовков до программирования различных текстовых эффектов. Рассмотрены примеры создания геометрических фигур и рисование различных кривых линий (от синусоиды до спирали Архимеда), а также примеры программирования градиентных заливок, геометрического преобразования объектов и сложных фигур: звезд, фигуры Инь-Янь, снежинок и др. Описана работа с импортируемыми изображениями: вращение, буксировка, наложение, плавное появление одной картинки из другой, куб с картинками на гранях и др. Рассмотрены примеры, связанные с работой клавиатуры и мыши. Показаны интересные особенности использования форм и элементов управления среды Visual Basic .NET. Приведены примеры создания игровых, шуточных, музыкальных и говорящих программ, а также различные полезные трюки и приемы программирования.

Книга сопровождается компакт-диском, содержащим все описанные примеры.

Для программистов

УДК 681.3.068+800.92VB.NET
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 25.04.05.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 42,57.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.02.953.Д.006421.11.04
от 11.11.2004 г. выдано Федеральной службой по надзору
в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-572-6

Оглавление

Введение

Дорогой читатель! Если вы купили эту книгу, надееясь найти на ее страницах примеры работы с базами данных, криптографией, офисными приложениями и тому подобными серьезными проектами, то сделали большую ошибку. Пойдите в магазин и сдайте книгу назад. Безусловно, работа с базами данных — это очень важно, а вы — весьма деловой человек, но моя книга совсем о другом. Здесь рассматривается только *занимательное* программирование.

О чем эта книга

Книги, посвященные освоению любого языка программирования, как правило, слишком серьезны и строги. Такой подход оправдан при изучении языка студентами для получения теоретических основ. Но для людей, самостоятельно изучающих язык, а также тех, кто хочет расширить свои познания, гораздо лучше изучать язык по другой методике. Общеизвестно, что даже самую скучную и нудную работу легче делать, если подойти к ней творчески. К счастью, тенденция меняется в лучшую сторону. В продаже появляются книги, в которых рассматриваются не абстрактные примеры, а интересные задачи. Вот и в этой книге я сделал попытку взглянуть на изучение Visual Basic .NET с другой стороны. Здесь собраны примеры, которые помогут лучше понять особенности языка через игры, шутки и создание красивых узоров и эффектов.

Для кого эта книга

В первую очередь книга предназначена для программистов средней и высокой квалификации. Здесь не будут объясняться азы программирования — как запустить программу, создать проект и т. д. Впрочем, начинающий программист тоже может воспользоваться этой книгой, так как я постарался подробно разобрать наиболее сложные моменты программирования. Очень хорошо, если читатель уже имел опыт программирования на Visual Basic 6.0, так как многие примеры переделаны из старых проектов. И на страницах книги мы не раз обратимся к особенностям перехода от Visual Basic 6.0 к Visual Basic .NET.

Требования, предъявляемые к читателю

Все примеры, приведенные в книге, написаны на Visual Basic .NET 2003. Но с вероятностью в 99,9 процента можно утверждать, что примеры будут работать и в Visual Basic 2002, а также в новых реализациях Visual Basic .NET, так как в примерах используются основополагающие принципы программирования, которые не подвержены кардинальным изменениям.

Дополнительные сведения

Выход новой версии любого языка программирования — это для программиста всегда головная боль. Приходится заново изучать новые возможности, предоставляемые последней версией программы. Не стал исключением и выход Visual Basic .NET. Если изменения в VB 6.0 по сравнению с VB 5.0 носили, скорее, косметический характер, то переход на версию VB .NET называют не иначе как революцией.

Анекдот в тему

Перепись населения у программиста:

– Ваш родной язык.

– Как это родной язык?

– Ну какой Вы язык с детства изучали, всю жизнь использовали?

– Basic.

– Да нет, настоящий.

– А! Настоящий! Тогда Си.

Теперь это в прошлом! Visual Basic .NET, использующий все возможности платформы .NET Framework, теперь такой же полноценный язык, как C# или C++. В новой версии VB .NET изменилось почти все — появились новые функции, методы, объекты, стиль программирования. А что делать программистам со своим багажом примеров, которыми они активно пользовались? Изучать новый язык и переделывать примеры. Вот и я столкнулся с такой проблемой. Пришлось заново пересматривать свою коллекцию исходных кодов. При написании книги я использовал примеры из своей обширной библиотеки, расположенной на сайте **http://rusproject.narod.ru/**

Основная масса примеров была написана на VB 6.0. Мне пришлось переписывать коды к примерам практически заново. Также я использовал различные источники из Интернета и книг. По ходу повествования я буду указы-

вать на эти источники, в которых вы можете почерпнуть немало дополнительных интересных сведений, также им будет посвящено *Заключение*.

Следует отметить, что приводимые примеры не являются верхом совершенства. Дело в том, что я стал собирать свою коллекцию примеров, когда сам только начинал осваивать премудрости программирования. Теперь, спустя несколько лет, я просматриваю исходные коды и замечаю, что где-то можно оптимизировать, где-то переписать заново, где-то просто удалить лишнее. Но я решил не менять радикально эти примеры. И улучшение примеров пусть останется вам в качестве домашнего задания. И мой совет — не пытайтесь механически переписывать пример. Постарайтесь придумать свой вариант, поиграть с различными параметрами. В общем, активно изучайте язык Visual Basic .NET. Только в этом случае можно рассчитывать на полное и уверенное овладение языком программирования. А я в меру своих скромных сил постараюсь вам помочь.

Как пользоваться примерами

На прилагаемом компакт-диске вы сможете найти все примеры, которые приводятся в книге. Примеры разбиты в папках по номерам глав. Например, в папке 7 находятся примеры из седьмой главы. Внутри каждой такой папки ищите файл с расширением sln. Запустив этот файл в Visual Basic .NET 2003 (или Visual Studio .NET 2003), вы загружаете тем самым все проекты выбранной главы. Чтобы запустить конкретный проект, вам необходимо выделить название проекта и в контекстном меню выбрать команду **Set as StartUp Project**.

Благодарности

Написание книги отняло у меня много сил. Мне пришлось долго и тщательно отбирать примеры, которые могли бы показаться вам интересными и занимательными. Кроме того, необходимо было найти золотую середину в стиле повествования — хотелось, чтобы книга выглядела не только как тривиальный сборник примеров, но и стала для вас увлекательным чтением. Моя личная жизнь отошла на второй план. У меня почти не оставалось времени ни на ремонт квартиры, ни на встречи с друзьями...

Существует множество сайтов по программированию, на которые есть возможность присылать свои примеры для ознакомления. Однажды кто-то назвал такие сайты свалкой файлов. Я категорически против такого подхода. Эти сайты можно сравнить с залежами полезных ископаемых. И, скачивая мегабайты кода на свой компьютер, находишь яркие и запоминающиеся проекты, изучение которых доставляет удовольствие. К сожалению, я не

могу назвать поименно всех авторов примеров, которые скопились в моей коллекции. Количество скачанных и изученных файлов не поддается никакому учету. Но всем им Большое спасибо!

Отдельно хочется выразить свою признательность руководителю проекта Игорю Шишигину и редактору Григорию Добину, которые помогли мне написать данную книгу.

Кот Рыжик любезно разрешил использовать фотографии со своим изображением в качестве иллюстраций к некоторым примерам при условии, что я угощу его удвоенной порцией любимой им печенки. Что я обязательно сделаю.

Александр Климов

Москва, 2005 год

ЧАСТЬ I

Строки и тексты

Глава 1

Эффекты со строками

Примеры научают лучше,
нежели толкования и книги.

Н. И. Лобачевский

А начну я свою книгу с самого простого — со строк. Казалось бы, что можно сделать со строками? И, тем не менее, можно придумать несколько интересных эффектов. Прежде всего, надо помнить, что строка состоит из отдельных элементов — символов. Можно, используя возможности таймера, выводить требуемую строку не целиком, а по мере надобности. Действуя так, мы получим анимационные эффекты печатающейся строчки, бегущей строки и т. д. Давайте перейдем к реализации наших планов.

1.1. Печатающаяся строка

Создадим новый проект `TypingText` (листинг 1.1). Расположим на форме надпись `Label1`, две кнопки `Button1` и `Button2`, а также таймер `Timer1`.

Совет

В наших примерах мы часто будем оставлять имена элементов управления по умолчанию. В реальных же проектах настоятельно рекомендуется использовать говорящие имена, например для кнопки можно использовать имя `butAniString` (Анимированная строка).

Свойству `Text` метки `Label1` присвоим значение `Занимательное программирование`, свойству `Text` кнопки `Button1` — значение `Старт`, а свойству `Interval` таймера `Timer1` присвоим значение `500`. У кнопки `Button2` присвоим свойству `Text` значение `Кнопка`. Можете добавить код по своему усмотрению для обработки щелчка данной кнопки. Свойству `Text` нашей формы `Form1` мы присвоим значение `Печатающаяся строка`. Данный текст будет отображаться в заголовке формы, указывая на цель наших упражнений. Остальные свойства оставляем по умолчанию. Теперь дважды щелкаем на

кнопке Button1, чтобы написать код, который будет выполняться при щелчке мыши на кнопке.

Листинг 1.1. Пример печатающейся строки

```
'----------------------------------------
' Печатающаяся строка © 2004 Александр Климов
'----------------------------------------
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        If Button1.Text = "Старт" Then
            Timer1.Enabled = True
            Button1.Text = "Стоп"
        Else
            Timer1.Enabled = False
            Button1.Text = "Старт"
        End If
    End Sub


    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
        ' Анимация слева направо
        Dim title As String = "Занимательное программирование"
        Static c As Integer = 0
        Label1.Text = title.Substring(0, c)
        Me.Text = title.Substring(0, c)
        c += 1
        If c = title.Length() + 1 Then c = 0

        ' Анимация справа налево
        Dim title2 As String = "Кнопка"
        Static i As Integer = title2.Length()
        Button2.Text = title2.Substring(0, i)
        i -= 1
        If i = -1 Then i = title2.Length()
    End Sub
```

Разберем, что мы написали. Кнопка с надписью **Старт** запускает таймер, при этом надпись **Старт** поменяется на **Стоп**. При повторном нажатии надпись **Стоп** заменяется обратно на **Старт**. При этом происходит включение и

выключение таймера. Вся работа по созданию эффекта происходит в событии таймера `Tick`. Далее идет объявление строковой переменной `title`, которой мы присвоили сразу же значение `Занимательное программирование`. Затем мы создаем статическую переменную `c`, которая служит счетчиком (counter). С помощью метода `Substring` класса `String` мы как бы наращиваем строку, начиная с первого символа и заканчивая последним. После чего обнуляем счетчик и все повторяем снова. Как видите, ничего сложного в реализации эффекта нет, а результат получился достаточно интересным. Вы можете в любой момент прервать анимацию строки, а повторным нажатием на кнопку возобновить вывод строки с остановленной позиции. Чтобы пример выглядел более интересным, я добавил анимированную строку и в заголовок формы:

```
Me.Text = title.Substring(0, c)
```

Если вам хочется не наращивать текст, а наоборот, уменьшать его, то данный пример легко переделать. В этом случае придется создать еще один счетчик `i`. Начальное значение счетчика равно длине заданной строки. При каждом срабатывании таймера будет вычитаться один символ из строки. При достижении счетчиком значения 0 снова присвоим ему старое значение длины строки и применим данный код не к метке, а ко второй кнопке. Ну, вот, мы и написали первый простой пример, который показывает, что и простыми методами можно добиваться интересных эффектов. Далее мы разберем более сложные примеры.

1.2. Мигающий заголовок

Усложним немного пример. Главной его особенностью будет отказ от использования визуальных средств при создании таймера. Умение программно создавать объекты, не прибегая к помощи панели инструментов, полезно при создании проектов без помощи пакета Visual Studio. Да-да, создавать программы можно даже при помощи обычного Блокнота.

Примечание

В состав .NET Framework входит бесплатный компилятор проектов. Написание кода в Блокноте тоже достаточно занимательное занятие, хотя, наверное, здесь больше подходит термин *экстремальное программирование*. Но в некоторых случаях это умение может быть востребовано. Например, у вас есть доступ к компьютеру с установленным .NET Framework, но без пакета Visual Studio .NET. Тогда вам ничего не остается, как написать программу в текстовом редакторе.

Но перейдем к примеру. Мы немного видоизменим способ анимации текста в заголовке формы. Теперь строка будет не только увеличиваться, но еще и мигнет несколько раз, привлекая внимание пользователя. Данный пример

взят из моей коллекции примеров, написанных на VB 6.0, и переделан с учетом новых возможностей VB .NET 2003. Откроем новый проект и назовем его `FlashText`. Вставим следующий код в редакторе кода (листинг 1.2).

Листинг 1.2. Мигающая анимированная строка

```
'---------------------------------------
' Мигающая анимированная строка © 2004 Александр Климов
'---------------------------------------
    Dim title As String = "Занимательное программирование"
    Dim b As Integer
    Dim p As Integer
    Dim I As Integer

    'Создадим два таймера программным путем
    Dim tmr As New System.Timers.Timer
    Dim tmr2 As New System.Timers.Timer

    Private Sub Form1_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        AddHandler tmr.Elapsed, AddressOf OnTimedEvent
        tmr.Interval = 200
        tmr.Enabled = True
        AddHandler tmr2.Elapsed, AddressOf OnTimedEvent2
        tmr2.Interval = 100
        tmr2.Enabled = False
        CheckAgain()
    End Sub

    Public Sub OnTimedEvent(ByVal source As Object, ByVal e As
System.Timers.ElapsedEventArgs)
        Dim t As String
        t = Microsoft.VisualBasic.Left(title, b)
        MyClass.Text = t
        b = b + 1
        If b > I Then
            b = 0
            tmr.Enabled = False
            tmr2.Enabled = True
```

```
            p = 0
        End If
    End Sub

   Public Sub OnTimedEvent2(ByVal source As Object, ByVal e As
System.Timers.ElapsedEventArgs)
        p = p + 1
        If p Mod 2 = 0 Then
            MyClass.Text = title
        Else
            MyClass.Text = ""
        End If
        If p = 10 Then
            tmr2.Enabled = False
            tmr.Enabled = True
        End If
    End Sub

    Sub CheckAgain()
        I = Len(title)
        b = 0
        p = 0
    End Sub
```

Разберем написанный код. Сначала мы объявили на уровне класса несколько переменных и два таймера. Обратите внимание, что мы не пользовались услугами визуального редактора, чтобы использовать элементы управления (в данном случае таймеры) в своем проекте. При загрузке формы таймеры включаются путем добавления двух обработчиков событий `Elapsed`. Также при загрузке формы я присвоил этим таймерам некоторые свойства и включил процедуру `CheckAgain`, о которой речь впереди. Для первого таймера обработчик события достаточно простой и практически совпадает с кодом предыдущего примера с самопечатающей строкой. Единственное отличие — мы подключаем второй таймер, который обеспечивает мигание заголовка формы. Процедура `CheckAgain` просто устанавливает старое значение для длины заголовка и обнуляет значения переменных `b` и `p`. Как видите, на самом деле сложного тут ничего нет. Сложным я назвал этот пример потому, что техника использования таймеров программным путем и написание обработчика событий поначалу вызывают некоторые неудобства у программистов, которые только недавно начали переходить от VB 6.0 к VB .NET.

1.3. Бегущая строка

Вероятно, вам приходилось видеть бегущую строку на электронных табло, где периодически повторяется одно и то же рекламное сообщение. Попробуем воспроизвести данный эффект в проекте `RunTitle` (листинг 1.3). Для примера достаточно поместить на форму только таймер.

Листинг 1.3. Бегущая строка

```
'--------------------------------------
' Бегущая строка © 2004 Александр Климов
'--------------------------------------
    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
        Dim StrTemp As String
        StrTemp = Me.Text
        StrTemp =  Microsoft.VisualBasic.Right(_
      StrTemp, Len(StrTemp) - 1) & Microsoft.VisualBasic.Left(StrTemp, 1)
        Me.Text = StrTemp
    End Sub

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As Sys-
tem.EventArgs) Handles MyBase.Load
        ' Необходимо добавить несколько пробелов,
        ' чтобы строки не слипались
        Me.Text = "Занимательное программирование       "
    End Sub
```

В этом примере мы отбрасываем первый символ из строки и добавляем его в конец нашей строки. Чтобы создать пространство между бегущими строками, пришлось добавить несколько пробелов в выводимый текст.

1.4. Эффект "Матрицы"

Вот еще один эффект со строкой, который я назвал *эффектом "Матрицы"*. Тот, кто смотрел этот фильм, помнит мелькающие буквы в его начале. Я попробовал реализовать этот эффект в проекте `Matrix` (листинг 1.4). Для примера понадобится таймер `Timer1` с интервалом, равным `100`, и свойством `Enable = True`.

Листинг 1.4. Эффект "Матрицы"

```
'--------------------------------------
' Эффект "Матрицы" © 2004 Александр Климов
'--------------------------------------
    Dim FXCounter As Integer = 0
    Dim FXCounter2 As Integer
    Dim SingleChr As String

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick

        ' Образец - текст, который должен появиться в конечном итоге
        Dim StrTemp As String
        StrTemp = "MATRIX"

        ' Проходим в цикле по всем символам
        ' Справа налево
        If FXCounter = 0 Then
            FXCounter2 = 31
            FXCounter = 1
        End If

        ' Позиция для замены символа
        Dim iStart As Integer
        iStart = Len(StrTemp) - (FXCounter - 1)

        Try
            If FXCounter2 = 31 Then SingleChr = Mid$(StrTemp, iStart, 1)
            FXCounter2 = FXCounter2 + 1
            Mid(Me.Text, iStart, 1) = Chr(FXCounter2)
            If FXCounter2 = Asc(SingleChr) Then
                FXCounter2 = 31
                FXCounter = FXCounter + 1
            End If

        Catch ex As Exception When iStart = 0
            'Обрабатываем ошибку
            Timer1.Enabled = False
```

```
        End Try
    End Sub

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Me.Text = "Kitten"
    End Sub
```

1.5. Заключение

В примерах этой главы мы рассматривали строки как набор символов. Но можно ведь рассматривать их как графические объекты. В этом случае мы можем разукрасить буквы разными цветами, задать градиентную заливку, вращать буквы вокруг своей оси и вдоль какой-нибудь кривой. Но все это будет рассмотрено в следующих главах.

1.6. Советы и рекомендации

Попробуйте придумать новые собственные эффекты для строк. Вот несколько рекомендаций.

- Измените пример из *разд. 1.3* таким образом, чтобы бегущая строка бежала слева направо.
- В этом же примере из *разд. 1.3* используются старые функции `Left` и `Right`, оставшиеся от VB 6.0 в целях совместимости. Попробуйте переписать пример с использованием методов класса `System.String`.

Глава 2

Игры с текстами

В этой главе мы поговорим о том, что можно сделать с текстами. Читатель спросит: "А чем, собственно, отличаются строки от текстов?" Вопрос справедливый — деление на строки и тексты в данном случае условно. Просто мне хотелось разъединить две области применения текстов в программе. Операции, которые мы проводили в *главе 1*, можно использовать, например, в заголовках формы. Согласитесь, что в заголовке формы мы не можем менять размеры шрифтов, цвет символов или вращать символы в разных направлениях. Теперь же у нас уже больше возможностей для создания красивых эффектов с использованием разных размеров символов и цветов.

2.1. Использование узорных и градиентных кистей

Самый простой способ добиться некоторого эффекта при работе с текстом — это использование штриховых и градиентных кистей. Более подробно о кистях я расскажу в *главе 3*, посвященной графике, а пока приведу небольшой пример — проект `UsingPens` (листинг 2.1).

Листинг 2.1. Использование узорных и градиентных кистей

```
'---------------------------------------
' Использование кистей © 2004 Александр Климов
'---------------------------------------
Imports System.Drawing.Drawing2D
Dim f As Font = New Font("Tahoma", 72)
Dim sBrick As String = "Кирпичики"
Dim sGradient As String = "Градиент"

    Private Sub Form1_Paint(ByVal sender As Object, ByVal e As
System.Windows.Forms.PaintEventArgs) Handles MyBase.Paint
        Dim g As Graphics = e.Graphics
```

```
        ' Создаем кисть с узором кирпича
        Dim hbr As New HatchBrush(HatchStyle.HorizontalBrick, _
            Color.White, Color.Tomato)
        ' Выводим строку, закрашенную кирпичами
        g.DrawString(sBrick, f, hbr, 0, 0)

        Dim rect As New Rectangle(10, 50, _
                     ClientSize.Width, ClientSize.Height)
        ' Создаем градиентную кисть
        Dim lgrb As New LinearGradientBrush(rect, _
                     Color.Violet, Color.SkyBlue, _
                     LinearGradientMode.BackwardDiagonal)
        g.DrawString(sGradient, f, lgrb, 0, 100)

    End Sub
```

В этом примере я применяю две разновидности кистей: первая кисть использует узор кирпича, вторая — градиентную заливку. Результат программы представлен на рис. 2.1.

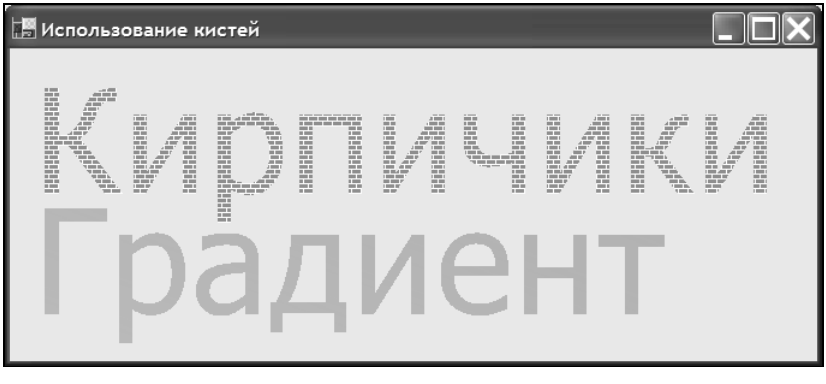

Рис. 2.1. Использование кистей для текстов

2.2. Объемный текст

Зачастую на форму выводится обычный текст, который выглядит не слишком красиво. Но можно придать текстам объемность. Причем существуют два эффекта объемности — выпуклый и вдавленный вид. Достигаются подобные эффекты простым смещением заданного текста с другим цветом,

имитирующим тень. Этот способ широко распространен и с успехом используется в самых различных областях. Но пора переходить к практике. Проект назовем `3DText` и поместим на форму одну кнопку. Для начала реализуем эффект выпуклости. Для этого нам понадобятся два вызова метода `DrawString`. При первом вызове метода на форме рисуется заданная строка серым цветом, при втором вызове — строка белого цвета со смещением в 1 или 2 пиксела (листинг 2.2).

Листинг 2.2. Создание выпуклого текста

```
'--------------------------------------
' Объемный текст © 2004 Александр Климов
'--------------------------------------
     Private Sub Form1_Paint(ByVal sender As Object, ByVal e As
System.Windows.Forms.PaintEventArgs) Handles MyBase.Paint
        Dim sb As New SolidBrush(Color.Gray)
        Dim f As New Font("Tahoma", 48, FontStyle.Bold)
        Dim g As Graphics = e.Graphics
        g.DrawString("Котенок", f, sb, 10, 10)
        sb.Color = Color.White
        g.DrawString("Котенок", f, sb, 8, 8)
    End Sub
```

Запустите проект и убедитесь, что выводимый текст выглядит выпукло (рис. 2.2).

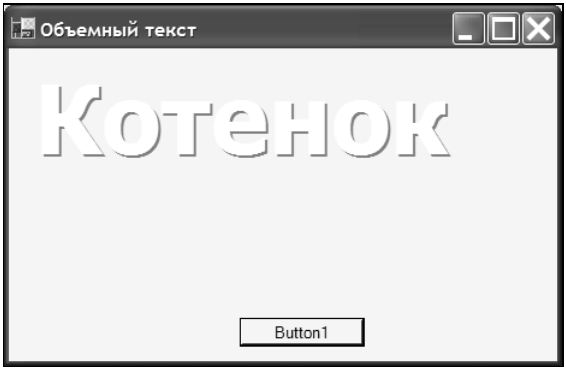

Рис. 2.2. Выпуклый текст

Для эффекта вдавленного текста необходимо сместить вторую строку ниже первой. Этот пример реализован через кнопку `Button1` (листинг 2.3).

Листинг 2.3. Создание вдавленного текста

```
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim sb As New SolidBrush(Color.Gray)
        Dim f As New Font("Tahoma", 48, FontStyle.Bold)
        Dim g As Graphics = CreateGraphics()
        g.Clear(BackColor)
        g.DrawString("Котенок", f, sb, 10, 10)
        sb.Color = Color.White
        g.DrawString("Котенок", f, sb, 12, 12)
        g.Dispose()
    End Sub
```

Снова запустите проект и нажмите на кнопку. Вы должны увидеть, что текст теперь стал вдавленным (рис. 2.3).

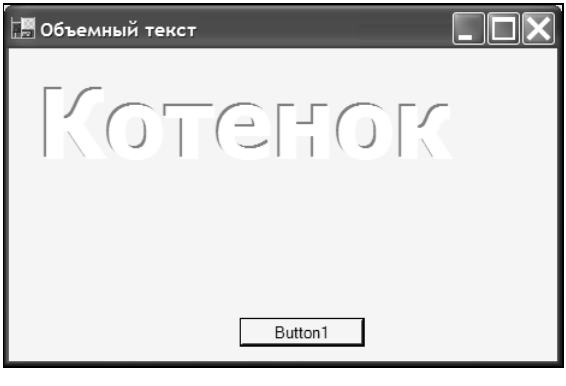

Рис. 2.3. Вдавленный текст

2.3. Контурный текст

Текст, выводимый на экран, по умолчанию заполнен каким-то цветом. Но если вам нужно вывести только контур символов, то можно воспользоваться классом `DrawPath`. Данный класс позволяет создавать различные контуры из

фигур и текстов. В проекте `Outline` (листинг 2.4) мы рассмотрим, как можно вывести на экран контуры символов заданного текста.

Листинг 2.4. Создание контурного текста

```
'--------------------------------------
' Контур © 2004 Александр Климов
'--------------------------------------
Imports System.Drawing.Drawing2D

    Private Sub Form1_Paint(ByVal sender As Object, ByVal e As
System.Windows.Forms.PaintEventArgs) Handles MyBase.Paint
        ' Создаем траекторию
        Dim pth As New GraphicsPath

        ' Добавляем строку
        pth.AddString("Кошкин дом", _
                      New FontFamily("Tahoma"), _
                      0, 70, New Point(30, 30), _
                      StringFormat.GenericDefault)

        ' Создаем синее перо
        Dim p As New Pen(Color.Blue, 2)

        ' Выводим контурный текст
        e.Graphics.DrawPath(p, pth)

        ' Очистим траекторию
        pth.Reset()

        ' Добавляем новый текст
        pth.AddString("Кошки-мышки", _
                      New FontFamily("Verdana"), _
                      0, 60, New Point(30, 120), _
                      StringFormat.GenericTypographic)

        ' Заливаем траекторию кистью
        e.Graphics.FillPath(Brushes.Peru, pth)
```

```
        ' Выводим на экран
        e.Graphics.DrawPath(p, pth)

        ' Освобождаем ресурсы
        pth.Dispose()
    End Sub
```

Пример достаточно прост для понимания. Создается контур `pth`, в который добавляется первый текст. На экране будут выводиться очертания символов текстовой строки. Для сравнения второй текст выводится, заполненный цветом `Peru` (рис. 2.4). Тем не менее вы все равно можете видеть контур символов.

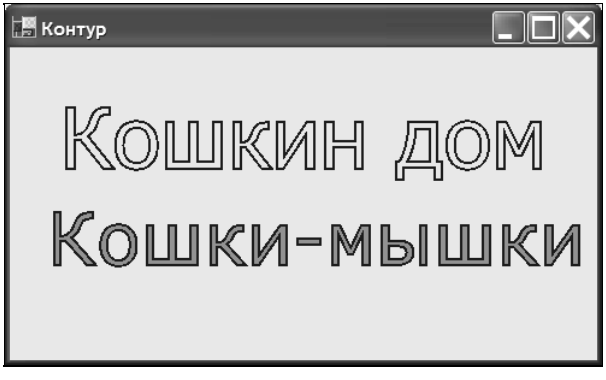

Рис. 2.4. Контуры текста

2.4. Раскаленный текст

Очень неплохо смотрится на экране эффект нагретого до высокой температуры материала или электрического ореола. Способ создания напоминает методику создания объемности. Идею эффекта я почерпнул из примера, расположенного на сайте Роберта Пауэлла (Robert W. Powell) **http://www.bobpowell.net/**. Добавим в проект `ElectricFX` графический объект `PictureBox1`. Местоположение и другие свойства данного элемента несущественны, поэтому оставим все свойства по умолчанию. Весь код содержится в событии `Paint` формы `Form`. В начале кода необходимо добавить ссылку на пространство имен `System.Drawing.Drawing2D` (листинг 2.5).

Листинг 2.5. Создание электрического эффекта

```
'--------------------------------------
' Электрический эффект © 2004 Александр Климов
'--------------------------------------
Imports System.Drawing.Drawing2D

    Private Sub Form1_Paint(ByVal sender As Object, ByVal e As
System.Windows.Forms.PaintEventArgs) Handles MyBase.Paint

        Me.BackColor = Color.Black

        ' Создадим рисунок в заданной пропорции
        Dim bm As New Bitmap(CInt(Me.ClientSize.Width / 5), _
                             CInt(Me.ClientSize.Height / 5))

        ' Создадим объект GraphicsPath
        Dim pth As New GraphicsPath

        ' Добавим строку в заданном стиле
        pth.AddString("Электрический эффект", _
                      New FontFamily("Tahoma"), _
                      CInt(FontStyle.Bold), 48, _
                      New Point(10, 20), _
                      StringFormat.GenericTypographic)

        ' Получим объект Graphics
        Dim g As Graphics = Graphics.FromImage(bm)

        ' Сформируем матрицу для создания эффекта
        Dim mx As Matrix
        mx = New Matrix(1.0F / 5, 0, 0, _
                        1.0F / 5, -(1.0F / 5), _
                       -(1.0F / 5))

        ' Выберем режим сглаживания
        g.SmoothingMode = SmoothingMode.AntiAlias

        ' Преобразуем объект Graphics
        g.Transform = mx
```

```
        ' Создадим перо
        Dim p As New Pen(Color.Tomato, 3)

        ' Рисуем вокруг созданного пути
        g.DrawPath(p, pth)

        ' и заполняем для лучшего эффекта
        g.FillPath(Brushes.Yellow, pth)

        ' Освобождаем ресурсы
        g.Dispose()

        ' Установим режим сглаживания для контура
        e.Graphics.SmoothingMode = SmoothingMode.AntiAlias
        e.Graphics.InterpolationMode = _
                     InterpolationMode.HighQualityBicubic

        ' и расширяем картинку для создания размытости краев
        e.Graphics.DrawImage(bm, ClientRectangle, 0, 0, _
                    bm.Width, bm.Height, GraphicsUnit.Pixel)

        ' Перерисовываем оригинальный текст
        e.Graphics.FillPath(Brushes.Black, pth)

        ' Освобождаем ресурсы
        pth.Dispose()
    End Sub
```

Дадим некоторые пояснения к коду. Чтобы создать ореол вокруг текста, я воспользовался свойством `InterpolationMode`, которое позволяет установить режим размытости текста. Техника создания эффекта ореола основана на двойном наложении текста. Сначала создается уменьшенная копия рисунка, которая будет растянута с применением режима интерполяции. Необходимо установить некоторый коэффициент сжатия картинки. В примере установлен коэффициент 1:5. И вот как это работает. Сначала формируется картинка в заданной пропорции. Затем создается траектория и устанавливается желаемый текст. Теперь можно получить объект `Graphics` из графического объекта `PictureBox` и создать матрицу, которая сжимает картинку. Заполняем траекторию желаемым цветом с помощью пера и для лучшего эффекта заливаем ее еще выбранной кистью. Растягиваем рисунок для получения