НИЕСН

АДАМ НАТАН

WPF 4

ПОДРОБНОЕ РУКОВОДСТВО

СИМВОЛ®

WPF 4

Unleashed

Adam Nathan

SAMS

WPF 4

Подробное руководство

Адам Натан

Санкт-Петербург — Москва
2011

Адам Натан

WPF 4. Подробное руководство

Перевод А. Слинкин

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Редактор	*Е. Тульсанова*
Корректоры	*С. Минин, О. Макарова*
Верстка	*Д. Орлова*

Натан А.

WPF 4. Подробное руководство. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 880 с., ил.

ISBN 978-5-93286-196-7

Windows Presentation Foundation (WPF) – рекомендуемая технология реализации пользовательских интерфейсов для Windows-приложений. Она позволяет создавать такие функционально насыщенные и визуально привлекательные приложения, о которых вы раньше не могли и мечтать. WPF дает возможность естественно объединять в одной программе традиционные интерфейсы, трехмерную графику, аудио и видео, анимацию, динамическую смену обложек, мультисенсорный ввод, форматированные документы и распознавание речи.

Книгу Адама Натана, известного гуру в области WPF, отличают полнота освещения, практические примеры и понятный язык. Издание содержит сведения о XAML – расширяемом языке разметки приложений; детально рассматриваются функциональные возможности WPF: элементы управления, компоновка, ресурсы, привязка к данным, стили, графика, анимация; уделено внимание новейшим средствам: мультисенсорному вводу, усовершенствованной визуализации текста, новым элементам управления, дополнениям языка XAML, программе Visual State Manager, переходным функциям в анимации; рассматриваются трехмерная графика, синтез и распознавание речи, документы и эффекты; демонстрируется создание популярных элементов пользовательского интерфейса, например галерей и экранных подсказок, а также создание более сложных механизмов организации пользовательского интерфейса, например выдвигающихся и стыкуемых панелей, как в Visual Studio; описывается, как создавать полноценные элементы управления WPF; демонстрируется создание гибридных приложений, в которых WPF сочетается с Windows Forms, DirectX и ActiveX; объясняется, как задействовать в WPF-приложении новые средства Windows 7, например списки переходов, и как обойти некоторые присущие WPF ограничения.

ISBN 978-5-93286-196-7
ISBN 978-0-672-33119-0 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 380-5007, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Подписано в печать 04.11.2011. Формат 70×100 1/16. Объем 55 печ. л.

Оглавление

Об авторе

Адам Натан – ведущий разработчик системы Microsoft Visual Studio, последняя версия которой представляет собой полноценное WPF-приложение. Ранее Адам был основателем, архитектором и разработчиком сайта Popfly, первого продукта корпорации Microsoft, построенного на базе технологии Silverlight, которая вошла в число 25 самых инновационных продуктов 2007 года по версии журнала *PCWorld Magazine*. Начав карьеру в составе коллектива разработчиков общеязыковой среды выполнения Microsoft (Common Language Runtime), Адам постоянно находился в гуще событий, связанных с созданием технологий .NET и WPF.

Многие сотрудники Microsoft и других компаний, занимающихся разработкой ПО, считают книги Адама обязательными для прочтения. Он автор бестселлера «WPF Unleashed» (Sams, 2006), который номинировался на премию Jolt Award в 2008 году, а также книг «Silverlight 1.0 Unleashed» (Sams, 2008) и «.NET and COM: The Complete Interoperability Guide» (Sams, 2002). Кроме того, Адам является одним из соавторов книг «ASP.NET: Tips, Tutorials, and Code» (Sams, 2001), «.NET Framework Standard Library Annotated Reference, Volume 2» (Addison-Wesley, 2005) и «Windows Developer Power Tools» (O'Reilly, 2006). Натан также создал сайт PINVOKE.NET и связанную с ним надстройку над Visual Studio. Связаться с Адамом можно через сайт *www.adamnathan.net* или по адресу @adamnathan в Twitter.

Посвящается

Ли ндсей, Тайлеру и Райану

Благодарности

Как всегда, я благодарю свою чудесную супругу Линдсей за невероятную поддержку и понимание. Нескончаемый процесс написания книг здорово сказывается на нашей жизни, и никто бы не удивился, если бы ее терпение наконец иссякло. Однако же никогда раньше ее поддержка не была столь ощутимой, как во время работы над этой книгой. Линдсей, что бы я без тебя делал!

Хотя создание любой книги, и этой в том числе, – по большей части глубоко личное занятие, она все же является плодом совместного труда многих талантливых и трудолюбивых людей. Не откажу себе в удовольствии назвать их поименно.

Я искренне благодарен Дуэйну Ниду (Dwayne Need), старшему менеджеру команды разработчиков WPF, – он потрясающий технический редактор. Его глубокие и проницательные рецензии на черновые варианты позволили значительно улучшить книгу. Выражаю признательность Роберту Хогу (Robert Hogue), Джо Кастро (Joe Castro) и Джордану Паркеру (Jordan Parker) за полезные отзывы. Дэвид Тейтельбаум (David Teitlebaum), специалист по трехмерной графике из команды разработчиков WPF, заслуживает самой горячей благодарности за согласие подкорректировать замечательную главу о 3D-графике, первоначально написанную Дэниелом Лехенбауэром (Daniel Lehenbauer). Ознакомиться с методологией и советами Дэниела и Дэвида – большая удача для любого читателя, подумывающего о том, чтобы заняться трехмерной графикой.

Хочется также поблагодарить следующих людей (в алфавитном порядке): Брайана Чэпмена (Brian Chapman), Беатрис де Оливейра Коста (Beatriz de Oliveira Costa), Эфианию Эчеруо (Ifeanyi Echeruo), Дэна Глика (Dan Glick), Нила

Кронлейга (Neil Kronlage), Рико Мариани (Rico Mariani), Майка Мюллера (Mike Mueller), Олега Овечкина, Лори Пирс (Lori Pearce), С. Рамини (S. Ramini), Роба Рилайи (Rob Relyea), Тима Райса (Tim Rice), Бена Ронко (Ben Ronco), Адама Смита (Adam Smith), Тима Снита (Tim Sneath), Дэвида Тредуэлла (David Treadwell) и Парамеша Вайдиянатана (Paramesh Vaidyanathan).

Я также выражаю признательность коллективу издательства Sams, а особенно Нилу Роуи (Neil Rowe) и Бетси Харрис (Betsy Harris), с которыми мне всегда приятно работать. Лучшей команды для подготовки книги не найти. Никто ни разу не сказал мне, что текст слишком длинный, или слишком короткий, или слишком отличается по стилю от типичной книги из серии «Подробное руководство». Мне предоставили свободу писать такую книгу, какую я хотел написать.

Спасибо также маме, папе и брату, которые раскрыли передо мной мир программирования, когда я еще учился в начальной школе. Если у вас есть дети, то посвятите их в магию создания программ, когда они еще прислушиваются к вашим словам! (А WPF и Silverlight помогут превратить этот опыт в незабываемое удовольствие!)

И наконец, спасибо вам за то, что вы взяли в руки эту книгу и прочитали ее хотя бы до этого места! Надеюсь, что вы на этом не остановитесь и для вас погружение в мир WPF 4 будет таким же завораживающим, как и для меня!

Нам важно ваше мнение!

Вы, читатель этой книги, – наш самый важный критик и комментатор. Мы ценим ваше мнение и хотим знать, что мы сделали правильно, что могли бы улучшить, на какие темы нам стоило бы выпускать книги. В общем, нам интересны любые мысли, которыми вы хотели бы с нами поделиться.

Вы можете писать мне по обычной или электронной почте о том, что понравилось или не понравилось в этой книге. А также о том, что мы могли бы еще сделать, чтобы наши книги стали лучше.

Пожалуйста, имейте в виду, что я не в состоянии ответить на технические вопросы по теме данной книги и что из-за большого количества получаемой почты я не всегда имею возможность ответить на каждое сообщение.

Если будете писать, не забудьте указать название и автора книги, а также свое имя и телефон или адрес электронной почты. Я внимательно изучу ваши замечания и направлю их автору и редакторам, работавшим над книгой.

Электронная почта: feedback@samspublishing.com

Почтовый адрес: Neil Rowe
Executive Editor
Sams Publishing
800 East 96th Street
Indianapolis, IN 46240 USA

В помощь читателям

Посетите наш сайт и зарегистрируйте свой экземпляр книги по адресу *informit.com/register*, чтобы получить доступ к обновлениям, загружаемым материалам и перечню замеченных опечаток.

Введение

Благодарим за выбор книги «WPF 4. Подробное руководство»! Windows Presentation Foundation (WPF) – самая современная из предлагаемых корпорацией Microsoft технологий создания графических интерфейсов пользователя в ОС Windows, будь то простые формы, документо-ориентированные окна, анимированные изображения, видео, 3D-среды с эффектом погружения или все вышеперечисленное. Технология WPF позволяет разрабатывать самые разнообразные приложения проще, чем когда бы то ни было ранее. Кроме того, она лежит в основе технологии Silverlight, которая распространяет WPF на Сеть и мобильные устройства, например телефоны на базе ОС Windows.

С момента анонсирования WPF в 2003 году (под кодовым названием Avalon) эта технология привлекла к себе пристальное внимание благодаря революционному изменению процесса разработки ПО – особенно со стороны программистов Windows, привыкших к Windows Forms и GDI. WPF сравнительно легко позволяет создавать интересные и полезные приложения, демонстрирующие разнообразные возможности, которые трудно реализовать с помощью других технологий. В версии WPF 4, выпущенной в апреле 2010 года, существенно улучшены практически все аспекты этой технологии.

WPF знаменует собой отход от предшествующих технологий в плане модели программирования, основополагающих идей и базовой терминологии. Даже просмотр исходного кода WPF-приложения (например, путем декомпиляции его компонентов с помощью программы .NET Reflector или ей подобной) может стать источником сюрпризов, потому что интересующий вас код часто находится не там, где вы ожидаете. А если добавить сюда еще и тот факт, что любую задачу в WPF можно решить несколькими способами, то легко прийти к разделяемому многими выводу: *изучить WPF очень трудно*.

Вот тут-то и приходит на помощь эта книга. Когда WPF еще только разрабатывалась, было понятно, что не будет недостатка в книгах, посвященных этой технологии. Но лично меня беспокоило другое: смогут ли авторы соблюсти баланс между изложением самой технологии со всеми ее своеобразными идеями и демонстрацией использования ее на практике. Поэтому, работая над первым изданием этой книги, «Windows Presentation Foundation Unleashed», я ставил перед собой следующие цели:

- Познакомить читателя с базовыми концепциями в доступной форме, не покидая практическую почву

- Ответить на вопросы, возникающие у большинства изучающих технологию, и показать, как решаются типичные задачи
- Предложить авторитетный источник информации благодаря участию членов команды разработчиков WPF, которые проектировали, реализовывали и тестировали эту технологию
- Четко очертить границы применимости технологии, не делая вид, что она представляет собой решение всех проблем
- Предложить удобное справочное руководство, к которому можно возвращаться снова и снова

Успех первого издания превзошел самые смелые мои ожидания. Теперь, по прошествии четырех лет, я полагаю, что и во втором издании мне удалось достичь тех же целей, только с большей глубиной. Помимо освещения новых возможностей, появившихся в WPF 3.5, WPF 3.5 SP1 и WPF 4, я более подробно рассказываю о средствах, имевшихся еще в первой версии WPF. Надеюсь, что любой читатель – неважно, приступает он к изучению WPF впервые или имеет солидный опыт работы с этой технологией, – согласится, что книга отвечает всем заявленным критериям.

Предполагаемая аудитория

Эта книга адресована разработчикам, заинтересованным в создании пользовательских интерфейсов для Windows. Неважно, что именно вы разрабатываете: программы для бизнеса или для массового потребителя, повторно используемые элементы управления, – здесь вы найдете сведения, позволяющие извлечь максимум пользы из платформы. Книга написана так, что ее смогут понять даже читатели, совсем не знакомые с каркасом .NET Framework. Но и те, кто уверенно владеет WPF, тоже найдут интересную для себя информацию. Для них эта книга станет как минимум ценным справочным руководством.

Поскольку в основе WPF и Silverlight лежат одни и те же технология и концепции, то, прочитав эту книгу, вы заодно повысите свою квалификацию как разработчика приложений на платформе Windows Phone 7 и веб-приложений.

Хотя книга и не предназначена специально для графических дизайнеров, знакомство с ней поможет лучше понять, что на самом деле представляют собой такие продукты, как Microsoft Expression Blend.

Подведем итоги. В этой книге:

- Содержится все, что необходимо знать об основанном на XML языке eXtensible Application Markup Language (XAML) для декларативного создания пользовательских интерфейсов, допускающих применение стилей.
- Весьма детально рассматриваются различные функциональные возможности WPF: элементы управления, компоновка, ресурсы, привязка к данным, стили, графика, анимация и многое другое.

- Особое внимание уделено новейшим средствам, в том числе мультисенсорному вводу, усовершенствованной визуализации текста, новым элементам управления, дополнениям языка XAML, программе Visual State Manager, переходным кривым в анимации и т. д.
- Освещаются вопросы, не затрагиваемые в большинстве других книг: трехмерная графика, синтез и распознавание речи, документы, эффекты и пр.
- Демонстрируется создание популярных элементов пользовательского интерфейса, например галерей, экранных подсказок, нестандартных способов компоновки элементов.
- Демонстрируется создание более сложных механизмов организации пользовательского интерфейса, например выдвигающихся и стыкуемых панелей, как в Visual Studio.
- Объясняется, как писать и развертывать приложения любых типов, в том числе со встроенной навигацией, исполняемых в браузере и содержащих эффектные непрямоугольные окна.
- Описывается, как создавать полноценные элементы управления WPF.
- Демонстрируется создание гибридных приложений, в которых WPF сочетается с Windows Forms, DirectX, ActiveX и другими технологиями.
- Объясняется, как задействовать в WPF-приложении новые средства Windows 7, например списки переходов, и как обойти некоторые присущие WPF ограничения.

Нельзя сказать, что в этой книге описаны абсолютно все возможности WPF (в частности, вопросы спецификации XML Paper Specification (XPS) лишь слегка затронуты). Их так много, что в одной книге рассмотреть все, на мой взгляд, невозможно. Но думаю, что вам понравятся широта и глубина охвата материала.

Примеры, приведенные в книге, написаны на XAML и C#; при обсуждении вопросов интероперабельности встречается также код на C++/CLI. Повсеместное использование языка XAML объясняется рядом причин: зачастую это самый быстрый способ записать исходный код; фрагменты, написанные на XAML, можно копировать в инструментальные средства и видеть результат, не прибегая к компиляции; основанные на WPF инструменты генерируют код на XAML, а не на процедурных языках; наконец, XAML не зависит от того, на каком .NET-совместимом языке вы пишете: Visual Basic, C# или еще каком-то. В тех случаях, когда соответствие между XAML и C# неочевидно, приводятся эквивалентные представления кода на обоих языках.

Требования к программному обеспечению

В этой книге рассматриваются окончательная версия Windows Presentation Foundation 4.0, соответствующий пакет Windows SDK и Visual Studio 2010.

Должно быть установлено следующее программное обеспечение:

- Версия ОС Windows, поддерживающая .NET Framework 4.0, например: Windows XP с пакетом обновлений Service Pack 2 (включая Media Center, Tablet PC и версию x64), Windows Server 2003 с пакетом обновлений Service Pack 1 (включая версию R2), Windows Vista и более поздние версии ОС.
- Каркас .NET Framework 4.0, который устанавливается по умолчанию начиная с Windows Vista. Для предыдущих версий Windows его можно бесплатно загрузить с сайта *http://msdn.com*.

Кроме того, рекомендуется иметь следующее программное обеспечение:

- Пакет средств разработки Windows Software Development Kit (SDK) и прежде всего включенные в него средства для .NET. Его также можно бесплатно загрузить с сайта *http://msdn.com*.
- Visual Studio 2010 или более позднюю версию; подойдет и бесплатная версия Express, имеющаяся на сайте *http://msdn.com*.

Для поддержки графического дизайна в среде WPF очень полезно иметь комплект программ Microsoft Expression (конкретно Expression Blend).

Некоторые из включенных в книгу примеров ориентированы на системы Windows Vista, Windows 7 или компьютер с поддержкой мультисенсорного ввода, но в большинстве своем примеры будут работать во всех перечисленных выше версиях Windows.

Примеры кода

Исходный код всех примеров, встречающихся в этой книге, можно загрузить со страницы *http://informit.com/title/9780672331190* или *http://adamnathan.net/wpf*.

Организация материала

Книга состоит из шести частей, в которых последовательно излагается материал, необходимый для эффективного использования WPF. Но если вам не терпится забежать вперед и сразу перейти к конкретной теме, например трехмерной графике или анимации, то можно читать и не по порядку. Ниже кратко описано содержание каждой части.

Часть I «Базовые сведения»

Эта часть состоит из следующих глав:

- Глава 1 «Почему именно WPF и как насчет Silverlight?»
- Глава 2 «Все тайны XAML»
- Глава 3 «Основные принципы WPF»

В главе 1 WPF сопоставляется с альтернативными технологиями, чтобы вам было проще решить, отвечает ли она вашим нуждам. В главе 2 подробно рассматривается язык XAML с целью заложить фундамент для понимания

XAML-кода, который встретится вам в этой книге и в реальной практике. В главе 3 освещаются уникальные особенности модели программирования WPF, выходящие за пределы того, что уже известно программистам, работающим с .NET.

Часть II «Создание WPF-приложения»

Эта часть состоит из следующих глав:

- Глава 4 «Задание размера, положения и преобразований элементов»
- Глава 5 «Компоновка с помощью панелей»
- Глава 6 «События ввода: клавиатура, мышь, стилус и мультисенсорные устройства»
- Глава 7 «Структурирование и развертывание приложения»
- Глава 8 «Особенности Windows 7»

В части II вы узнаете, как собрать и развернуть традиционное приложение (хотя затрагиваются и некоторые дополнительные механизмы, например преобразования, непрямоугольные окна и технология Aero Glass). В главах 4 и 5 обсуждается компоновка элементов управления (и других элементов) в пользовательском интерфейсе программы. Глава 6 посвящена событиям ввода, в том числе поддержке новых устройств с мультисенсорным вводом. В главе 7 рассматриваются различные способы пакетирования и развертывания пользовательских интерфейсов на базе WPF для получения законченного приложения. В последней главе этой части речь пойдет об использовании некоторых возможностей Windows 7, позволяющих создавать приложения с современным внешним видом.

Часть III «Элементы управления»

Эта часть состоит из следующих глав:

- Глава 9 «Однодетные элементы управления»
- Глава 10 «Многодетные элементы управления»
- Глава 11 «Изображения, текст и другие элементы управления»

Часть III представляет собой обзор элементов управления, встроенных в WPF. Среди них много хорошо знакомых, но есть и несколько неожиданных. Две категории элементов управления – однодетные и многодетные[1] – настолько важные и глубокие темы, что заслуживают отдельных глав. Прочие элементы управления рассматриваются в главе 11.

---

[1] Термины «однодетный» и «многодетный элемент управления» (content control и items control) могут показаться непривычными, однако же какой-то эквивалент предложить необходимо. Термин «content control» буквально означает «элемент управления со свойством `Content`», а «items control» – «элемент управления со свойством `Items`». Вариант «элемент управления содержимым», встречающийся в локализованных продуктах Microsoft, совершенно не отражает сути дела. – *Прим. перев.*

Часть IV «Средства для профессиональных разработчиков»

Эта часть состоит из следующих глав:

- Глава 12 «Ресурсы»
- Глава 13 «Привязка к данным»
- Глава 14 «Стили, шаблоны, обложки и темы»

Средства, рассматриваемые в части IV, не относятся к активно используемым в WPF-приложениях, но их применение может существенно повысить качество процесса разработки. Они незаменимы для профессиональных разработчиков, серьезно относящихся к созданию надежных и удобных для сопровождения приложений или компонентов. Речь идет не столько о результатах, видимых конечному пользователю, сколько о рекомендуемых способах достижения желаемого результата.

Часть V «Мультимедиа»

Эта часть состоит из следующих глав:

- Глава 15 «Двумерная графика»
- Глава 16 «Трехмерная графика»
- Глава 17 «Анимация»
- Глава 18 «Аудио, видео и речь»

В этой части рассматриваются те возможности WPF, которые обычно вызывают наибольший интерес. Поддержка двумерной и трехмерной графики, анимации, видео и пр. позволяет создавать приложения, поражающие воображение пользователя. Именно эти средства наряду со способами их использования и отличают WPF от предшествующих технологий. WPF снижает барьеры, стоящие на пути включения такого содержимого в приложения, позволяя браться за задачи, о которых раньше вы и помыслить не могли!

Часть VI «Дополнительные вопросы»

Эта часть состоит из следующих глав:

- Глава 19 «Интероперабельность с другими технологиями»
- Глава 20 «Пользовательские и нестандартные[1] элементы управления»
- Глава 21 «Компоновка с помощью нестандартных панелей»

В части VI рассматриваются вопросы, интересные для разработчиков более сложных WPF-приложений и элементов управления. Уже имеющиеся элементы управления WPF допускают применение стилей в очень широких пределах, поэтому потребность в создании дополнительных элементов не так насущна.

---

[1] В терминологии Microsoft – «настраиваемые». – *Прим. перев.*

Типографские соглашения

В этой книге новые термины и иные специальные элементы выделяются с помощью шрифтов, а именно:

Шрифт	Назначение
Курсив	Используется при первом определении термина и иногда для акцентирования внимания, а также для имен файлов и интернет-адресов
Моноширинный	Используется для записи выводимых на экран сообщений, листингов и команд кода. В листингах *курсивное моноширинное начертание* применяется для обозначения заменяемого текста

В книге встречаются следующие виды врезок.

FAQ

Что такое врезка FAQ?

В такой врезке формулируется вопрос, который может возникнуть у читателя в данном месте, и дается краткий ответ на него.

КОПНЕМ ГЛУБЖЕ

Врезки «Копнем глубже»

В такой врезке представлена более подробная информация по теме в дополнение к содержащейся в основном тексте. Можно сказать, что это добавочные сведения для особо любознательных.

СОВЕТ

Это описание приемов, которые могут пригодиться на практике, например самый быстрый способ достижения цели или альтернативный подход, дающий более качественный результат либо позволяющий решить задачу скорее и проще.

ПРЕДУПРЕЖДЕНИЕ

Такие врезки привлекают внимание к действию или условию, способному привести к неожиданному либо непредсказуемому результату, – с объяснением того, как избежать подобных последствий.

I

Базовые сведения

- Взгляд в прошлое
- Появление WPF
- Эволюция WPF
- Что такое Silverlight

1

Почему именно WPF и как насчет Silverlight?

В кино и на телевидении главные герои обычно не похожи на обычных людей, которые встречаются нам в повседневной жизни. Они внешне более привлекательные, обладают мгновенной реакцией и почему-то всегда точно знают, что делать дальше. То же самое можно сказать и про компьютерные программы, которые показывают в фильмах.

Впервые меня это поразило в 1994 году при просмотре фильма «Disclosure» (Разоблачение) с Майклом Дугласом и Деми Мур. Почтовая программа, которой они пользовались, выглядела совершенно не так, как Microsoft Outlook! По ходу фильма мы дивились различным визуальным эффектам: вращающееся трехмерное «е»; сообщения, которые разворачиваются при открытии и комкаются при удалении; намеки на поддержку рукописного ввода и симпатичная анимация при распечатке сообщения. (Эта почтовая программа еще не самая нереалистичная из встречающихся в фильме. Достаточно лишь вспомнить «базу данных виртуальной реальности».)

Голливуд уже давно говорит нам, что реальные программы вовсе не такие впечатляющие, какими должны быть, и речь здесь идет отнюдь не о функциональности. Вы, наверное, и сами сможете вспомнить несколько примеров забавных и фантастичных программ из известных фильмов и сериалов. Однако в последние годы реальные программы стали подтягиваться к голливудским стандартам! Это наблюдается и в традиционных операционных системах (да, и в Windows тоже), и в веб-приложениях, и в ПО для таких устройств, как iPhone, iPad, Zune, TiVo, Wii, Xbox, Windows Phone и многих-многих других. Пользователи ожидают от программ большего, а компании-производители тратят массу времени и денег, чтобы превзойти конкурентов в области разработки пользовательского интерфейса. И это касается не только программ, рассчитанных на массового потребителя. Даже бизнес-приложения

и инструменты для внутреннего использования могут здорово выиграть от улучшения интерфейса.

Однако при возрастании требований к пользовательскому интерфейсу традиционного подхода и старых технологий разработки приложений часто оказывается недостаточно. Современные программы обычно нуждаются в быстром и кардинальном изменении интерфейса по инициативе различных сторон: профессиональных дизайнеров, проектировщиков пользовательских интерфейсов или начальства, которое хочет, чтобы приложение выглядело более эффектно и включало анимацию. Но для этого необходима технология, позволяющая естественным образом отделить пользовательский интерфейс от реализации приложения, а визуальное поведение – от внутренней программной логики. У разработчиков должна быть возможность создавать внешне аскетичные, но вместе с тем полнофункциональные приложения, которые впоследствии могут быть красиво оформлены дизайнерами без привлечения программистов. Однако присущий Win32 стиль программирования, при котором элементы управления содержат код собственной визуализации, как правило, сильно затрудняет быструю смену интерфейса.

В 2006 году корпорация Microsoft выпустила в свет технологию, которая позволила разработчикам создавать приложения XXI века, отвечающие возросшим требованиям. Она называется Windows Presentation Foundation (WPF). С выходом версии WPF 4 в 2010 году эта технология позволила добиваться еще более впечатляющих результатов при разработке практически любых программ. Всего через десять лет после того, как Том Круз поспособствовал популяризации идеи компьютера с мультисенсорным интерфейсом ввода в фильме «Особое мнение» (Minority Report), и после реализации такого интерфейса в самых разных устройствах (из которых наиболее известен iPhone), WPF 4 и Windows 7 его применение стало массовым. Голливуду пора придумывать что-нибудь новенькое!

Взгляд в прошлое

Базовые технологии большинства интерфейсов в Windows – интерфейс графического устройства (Graphics Device Interface, GDI) и подсистема USER – появились в Windows 1.0 еще в 1985 году. В мире технологий это смело можно назвать доисторическим периодом! В начале 1990-х годов компания Silicon Graphics разработала ставшую популярной графическую библиотеку OpenGL для двумерной и трехмерной графики как в Windows, так и в других системах. Она была с восторгом принята компаниями, работающими в сфере создания систем автоматизированного проектирования, программ визуализации научных данных и игр. Технология Microsoft DirectX, представленная в 1995 году, обеспечила высокоскоростную альтернативу для 2D-графики, ввода, сетевого взаимодействия, работы со звуком, а со временем и 3D-графики (которая стала возможной с версией DirectX 2, вышедшей в 1996 году).

Впоследствии и в GDI, и в DirectX было внесено много существенных улучшений. Например, технология GDI+, представленная в Windows XP, добавила

поддержку прозрачности и градиентные кисти. Однако ввиду большой сложности и в отсутствие аппаратного ускорения она работает медленнее, чем GDI. Что касается технологии DirectX (кстати, используемой в Xbox), то постоянно выходят новые версии, раздвигающие пределы возможностей компьютерной графики. После появления каркаса .NET и управляемого кода (в 2002 году) разработчики получили очень продуктивную модель для создания Windows- и веб-приложений. Включенная в нее технология Windows Forms (основанная на GDI+) стала основным способом создания пользовательских интерфейсов в Windows для разработчиков на C#, Visual Basic и (в меньшей степени) C++. Она пользовалась успехом и оказалась весьма продуктивной, но имела фундаментальные ограничения, уходящие корнями в GDI+ и подсистему USER.

Начиная с версии DirectX 9 Microsoft стала поставлять эту систему для управляемого кода (подобно тому, как в прошлом поставлялись библиотеки специально для Visual Basic), которая впоследствии была заменена каркасом XNA Framework. Хотя это и позволило разработчикам на C# использовать DirectX без многих проблем, связанных с интероперабельностью .NET и COM, однако работать с управляемыми каркасами оказалось не намного проще по сравнению с неуправляемыми альтернативами. Исключение составляет только разработка игр в среде XNA Framework, поскольку она включает в себя специализированные для этой цели библиотеки и работает с такими мощными инструментами, как XNA Framework Content Pipeline и XNA Game Studio Express.

Поэтому, хотя разработать в Windows почтовую программу с 3D-эффектами (как в фильме «Разоблачение») можно было уже в середине 90-х годов с помощью альтернативных GDI технологий (фактически комбинируя DirectX или OpenGL с GDI), на практике этот способ очень редко применялся даже и десять лет спустя. На то было несколько причин: аппаратное обеспечение, позволяющее достичь нужных результатов, было не так распространено вплоть до последнего времени; работать с альтернативными технологиями на порядок сложнее; и к тому же использование GDI считалось «вполне приемлемым».

Графические подсистемы компьютеров продолжали совершенствоваться и дешеветь, ожидания потребителей росли, но до появления WPF проблеме сложности построения выразительных пользовательских интерфейсов не уделяли должного внимания. Некоторые разработчики самостоятельно брались за ее решение, стремясь сделать свои приложения и элементы управления более привлекательными. Простым примером тут является использование растровых изображений вместо стандартных кнопок. Однако мало того что подобные нестандартные решения было трудно реализовывать, они еще зачастую оказывались ненадежными. Основанные на них приложения не всегда доступны людям с ограниченными возможностями, плохо адаптируются к высокому разрешению и имеют другие визуальные огрехи.

Появление WPF

Корпорация Microsoft понимала, что необходимо нечто совершенно новое, свободное от ограничений GDI+ и подсистемы USER, но не менее продуктивное и удобное в использовании, чем Windows Forms. И с учетом роста популярности кроссплатформенных приложений, основанных на HTML и JavaScript, Windows отчаянно нуждалась в столь же простой технологии, которая при этом еще и позволяла бы задействовать все возможности локального компьютера. И Windows Presentation Foundation (WPF) дала в руки разработчиков ПО и графических дизайнеров тот инструмент, который был необходим для создания современных решений и не требовал освоения сразу нескольких сложных технологий. И хотя слово Presentation (представление) – всего лишь высокопарный синоним привычного «пользовательского интерфейса», возможно, оно лучше отражает более высокий уровень визуального совершенства, которого ждут от современных приложений, равно как и обширную новую функциональность, включенную в WPF!

Перечислим основные возможности, которые предоставляет WPF.

- **Широкая интеграция.** До WPF разработчикам в Windows, которые хотели использовать одновременно 3D-графику, видео, речь и форматированные документы в дополнение к обычной двумерной графике и элементам управления, приходилось изучать несколько независимых технологий, плохо совместимых между собой и не имеющих встроенных средств сопряжения. А в WPF все это входит в состав внутренне согласованной модели программирования, поддерживающей композицию и визуализацию разнородных элементов. Одни и те же эффекты применимы к различным видам мультимедийной информации, а один раз освоенная техника может использоваться и для других целей.
- **Независимость от разрешающей способности.** Только представьте себе мир, в котором переход к более высокому разрешению экрана или принтера не означает, что все уменьшается. Вместо этого графические элементы и текст только становятся более четкими! Представьте себе пользовательский интерфейс, который прекрасно выглядит и на маленьком нетбуке, и на полутораметровом экране телевизора! WPF все это обеспечивает и дает возможность уменьшать или увеличивать элементы на экране независимо от его разрешения. Это стало возможным благодаря тому, что WPF основана на использовании векторной графики.
- **Аппаратное ускорение.** Поскольку WPF основана на технологии Direct3D, то все содержимое в WPF-приложении, будь то двумерная или трехмерная графика, изображения или текст, преобразуется в трехмерные треугольники, текстуры и другие объекты Direct3D, а потом отрисовывается аппаратной графической подсистемой компьютера. Таким образом, WPF-приложения задействуют все возможности аппаратного ускорения графики, что позволяет добиться более качественного изображения и одновременно повысить производительность (поскольку часть работы перекладывается с центральных процессоров на графические). При этом от применения

новых графических ускорителей и их драйверов выигрывают все WPF-приложения (а не только высококлассные игры). Но WPF *не требует* обязательного наличия высокопроизводительной графической аппаратуры. В ней имеется и собственный программный конвейер визуализации. Это позволяет использовать возможности, которые пока еще не поддерживаются аппаратно (например, осуществлять высокоточное отображение любого содержимого на экране). Программная реализация используется и как запасной вариант в случае отсутствия аппаратных ресурсов (например, если в системе стоит устаревшая графическая карта, или карта современная, но GPU не хватает ресурсов, скажем, видеопамяти).

- **Декларативное программирование.** Декларативное программирование не является уникальной особенностью WPF, поскольку в программах на платформе Win16/Win32 сценарии описания ресурсов для определения компоновки диалоговых окон и меню применяются вот уже 25 лет. И в .NET-приложениях часто используются декларативные атрибуты наряду с конфигурационными и ресурсными XML-файлами. Однако в WPF применение декларативного программирования вышло на новый уровень благодаря языку XAML (eXtensible Application Markup Language – расширяемый язык разметки *приложений*) (произносится «заммел»). Сочетание WPF и XAML аналогично использованию HTML для описания пользовательского интерфейса, но с гораздо более широкими выразительными возможностями. И эта выразительность выходит далеко за рамки описания интерфейса. В WPF язык XAML применяется в качестве формата документов, для представления 3D-моделей и многого другого. В результате дизайнер может непосредственно влиять на внешний вид приложения и некоторые особенности его поведения; раньше для этого, как правило, приходилось писать код. В следующей главе мы будем рассматривать XAML подробно.
- **Богатые возможности композиции и настройки.** В WPF элементы управления могут сочетаться немыслимыми ранее способами. Можно создать комбинированный список, содержащий анимированные кнопки, или меню, состоящее из видеоклипов! Конечно, сама мысль о таком интерфейсе может привести в ужас, но важно то, что для оформления элемента способом, о котором его автор и не помышлял, не понадобится писать никакой (!) код (и в этом коренное отличие от предшествующих технологий, где отрисовка элементов жестко задавалась создателем кода). Кроме того, отметим, что WPF позволяет безо всякого труда радикально изменять обложку (скин) приложения (мы рассмотрим этот вопрос в главе 14 «Стили, шаблоны, обложки и темы»).

Короче говоря, цель WPF – соединить в себе все лучшее, имеющееся в DirectX (трехмерная графика и аппаратное ускорение), Windows Forms (продуктивность разработки), Adobe Flash (развитая поддержка анимации) и HTML (декларативная разметка). Надеюсь, эта книга убедит вас в том, что WPF повышает производительность труда, дает больше возможностей и более увлекательна, чем любая другая технология, с которой вам доводилось работать прежде!

КОПНЕМ ГЛУБЖЕ

GDI и аппаратное ускорение графики

На самом деле в технологии GDI в Windows XP тоже использовалось аппаратное ускорение графики. Модель драйвера видеокарты явно поддерживает ускорение наиболее распространенных операций GDI. В Windows Vista реализована новая модель драйвера видеокарты без аппаратного ускорения примитивов GDI. Вместо этого применяется программная реализация устройства канонического отображения для поддержки операций GDI в унаследованных драйверах. Однако в Windows 7 восстановлено частичное аппаратное ускорение для примитивов GDI.

FAQ

Позволяет ли WPF сделать что-то, чего нельзя было сделать ранее?

Если быть совсем точным, то нет. Точно так же ни C#, ни .NET не позволяют сделать ничего, что нельзя было бы реализовать на языке ассемблера. Вопрос лишь в том, сколько времени и сил потребуется для достижения желаемого результата!

Если вы попытаетесь создать эквивалент WPF-приложения с нуля без использования WPF, то придется не только заниматься отрисовкой пикселов на экране и взаимодействием с устройствами ввода, но также проделать массу дополнительной работы для поддержки доступности и локализации, тогда как в WPF все это уже встроено. Кроме того, WPF обеспечивает простой способ задействовать все возможности Windows 7, например определить списки перехода с помощью коротенького кода на XAML (см. главу 8 «Особенности Windows 7»).

Поэтому я полагаю, что с учетом времени и финансовых затрат большинство людей утвердительно ответят на поставленный вопрос.

FAQ

Когда следует использовать DirectX вместо WPF?

DirectX больше подходит для разработчиков зрелищных игр или приложений со сложными 3D-моделями, в которых требуется максимальная производительность. Отметим, однако, что очень легко написать такое приложение DirectX, которое будет работать гораздо медленнее аналогичного WPF-приложения.

DirectX – это низкоуровневый интерфейс к графическому оборудованию, который делает явными все особенности конкретного графического процессора. DirectX можно назвать «языком ассемблера в мире графики»: вы можете делать все, что поддерживает данный GPU, но учитывать капризы разнообразной аппаратуры придется самостоятельно. Это трудно, зато такой низкоуровневый интерфейс позволяет опытным разработчикам достичь желаемого компромисса между высоким качеством графики и скоростью работы. Кроме того, DirectX позволяет работать с последними достижениями в области GPU, а они появляются гораздо быстрее, чем новые версии WPF.

С другой стороны, WPF обеспечивает более высокий уровень абстракции. Вы передаете системе описание сцены, а она уже сама решает, как оптимально визуализировать ее с учетом имеющихся аппаратных ресурсов. (Это система, работающая в *режиме запоминания*, а не в *режиме непосредственной визуализации*.) В WPF основное внимание уделено двумерной графике, а трехмерная графика ограничена сценариями визуализации данных и интеграцией с 2D без претензии на полную поддержку всех возможностей DirectX.

Однако, отдавая предпочтение DirectX, следует иметь в виду потенциально астрономическое увеличение стоимости разработки. По большей части затраты связаны с необходимостью тестировать приложение для всех возможных комбинаций драйверов и GPU, которые вы намереваетесь поддерживать. Одно из основных преимуществ построения приложения на базе WPF состоит в том, что Microsoft уже провела такое тестирование за вас! Вы же можете сосредоточиться на измерении производительности, для чего достаточно относительно дешевой системы. А тот факт, что WPF-приложение способно использовать установленный на компьютере пользователя GPU даже в условиях частичного доверия, – еще один аргумент в пользу выбора этой технологии.

Отметим также, что WPF и DirectX можно использовать совместно в одном приложении. В главе 19 «Интероперабельность с другими технологиями» описано, как это делается.

Эволюция WPF

Как ни странно, WPF 4 действительно является четвертой основной версией WPF. Странность в том, что первый выпуск имел номер 3.0! Он увидел свет в ноябре 2006 года и получил название WPF 3.0, потому что вошел в состав каркаса .NET Framework 3.0. Второй выпуск – WPF 3.5 – состоялся почти год спустя (если быть точным, то на день раньше). Третья основная версия вышла еще через год (в августе 2008 года). Она была включена в пакет обновлений Service Pack 1 (SP1) для .NET 3.5, но в части WPF это был не обычный пакет обновлений, поскольку появилось много новых возможностей и улучшений.

Кроме основных версий в августе 2008 года на сайте *http://wpf.codeplex.com* Microsoft представила набор инструментов WPF Toolkit, который содержит многочисленные инструментальные средства и примеры использования и обновляется несколько раз в год. WPF Toolkit предназначен для ускоренного внедрения новых возможностей, правда, в экспериментальной форме (и зачастую вместе с исходным кодом). Нередко средства, впервые появившиеся в WPF Toolkit, со временем включаются в новые версии WPF – в зависимости от мнения пользователей об их желательности и степени готовности.

На момент выпуска первой версии для WPF не существовало практически никакой инструментальной поддержки. В последующие месяцы были написаны примитивные WPF-расширения для Visual Studio 2005, а затем состоялся первый выпуск Expression Blend для публичного ознакомления. Теперь же среда Visual Studio 2010 включает не только полноценную поддержку WPF, но и сама была существенно переписана и ныне является WPF-приложением!

Программа Expression Blend, полностью написанная средствами WPF, также получила множество новых функций для проектирования и создания прототипов замечательных пользовательских интерфейсов. Кроме того, за последние несколько лет немало WPF-приложений было выпущено такими компаниями, как Autodesk, SAP, Disney, Blockbuster, Roxio, AMD, Hewlett Packard, Lenovo и многими другими. Сама корпорация Microsoft, конечно же, тоже разработала многочисленные приложения, основанные на WPF (Visual Studio, Expression, Test and Lab Manager, Deep Zoom Composer, Songsmith, Surface, Semblio, Robotics Studio, LifeCam, Amalga, Games for Windows LIVE Marketplace, Office Communicator Attendant, Active Directory Administrative Center, Dynamics NAV, Pivot, PowerShell ISE и многие другие).

СОВЕТ

Чтобы узнать, какие WPF-элементы применяются в конкретном WPF-приложении, можно воспользоваться программой Snoop, представленной на сайте *http://snoopwpf.codeplex.com*.

Давайте более подробно рассмотрим, как WPF менялась со временем.

Усовершенствования в WPF 3.5 и WPF 3.5 SP1

В версиях WPF 3.5 и 3.5 SP1 произошли следующие существенные изменения:

- **Интерактивная 3D-графика** – интеграция двумерной и трехмерной графики улучшилась благодаря базовому классу `UIElement3d`, который позволил 3D-элементам принимать данные, получать фокус клавиатуры и события; классу со странным названием `Viewport2DVisual3D`, который позволил помещать любой интерактивный 2D-элемент управления на 3D-сцену; и другим нововведениям. Подробнее см. в главе 16 «Трехмерная графика».
- **Полноценная интероперабельность с DirectX** – ранее WPF-приложение могло взаимодействовать с DirectX только на уровне общей для обеих технологий платформы Win32. Теперь с помощью класса `D3DImage` можно работать непосредственно с поверхностью Direct3D, а не с ее описателями `HWND`. Среди прочего это позволяет размещать WPF-содержимое поверх DirectX-содержимого и наоборот. См. главу 19.
- **Улучшенная привязка к данным** – в WPF появилась поддержка технологии привязки XLINQ, улучшены способы контроля данных и отладки, а также форматирование выводимых строк средствами XAML, что позволяет в ряде случаев обойтись без написания процедурного кода. См. главу 13 «Привязка к данным».
- **Улучшенные спецэффекты** – уже самая первая версия WPF поставлялась с полезными спецэффектами для рисунков (размывание, тени, внешнее свечение, выдавливание и выпуклость), но пользоваться ими не рекомен-

довалось из-за крайне низкой производительности. Теперь все изменилось – добавлен новый набор эффектов с поддержкой аппаратного ускорения, а также реализована совершенно иная архитектура, позволяющая подключать собственные эффекты с аппаратным ускорением с помощью пиксельных построителей текстур (pixel shaders). См. главу 15 «Двумерная графика».

- **Высокопроизводительное произвольное рисование** – раньше в WPF не имелось хорошего механизма произвольного рисования, масштабируемого на тысячи точек или фигур, поскольку даже примитивы самого низкого уровня были для этого слишком медленны. Теперь класс `WriteableBitmap` модернизирован таким образом, что при рисовании можно указывать изменившиеся области, а не обновлять весь растр в каждом кадре! Впрочем, поскольку этот класс позволяет только задавать отдельные пикселы, то рисованием такой процесс можно назвать с большой натяжкой.
- **Улучшения в области обработки текста** – повышена производительность, улучшена поддержка интернационализации (усовершенствован редактор методов ввода (IME) и улучшена поддержка индийских языков). Модернизации подверглись также классы `TextBox` и `RichTextBox`. См. главу 11 «Изображения, текст и другие элементы управления».
- **Модернизация приложений с частичным доверием** – .NET-приложениям с частичным доверием стало доступно гораздо больше функциональности, например возможность обращаться к WCF (Windows Communication Foundation) для вызова веб-служб (с привязкой `basicHttpBinding`) и возможность читать и записывать файлы cookie. Кроме того, технология XAML Browser Applications (XBAPs) – основной механизм запуска WPF-приложений с частичным доверием, – ранее доступная только для Internet Explorer, теперь распространена и на браузер Firefox (но необходимая для этого надстройка больше не устанавливается по умолчанию).
- **Улучшенное развертывание приложений и каркаса .NET** – проявляется в различных формах: ускорение процедуры установки и уменьшение объема .NET за счет внедрения технологии «клиентского профиля», которая позволяет исключить части .NET, нужные только для серверов (например, ASP.NET); новый компонент-«загрузчик», который обрабатывает зависимости .NET, ранее установленные компоненты и появившиеся обновления и позволяет осуществлять установку нестандартных дистрибутивов; а также разнообразные новые возможности в технологии ClickOnce.
- **Улучшенная производительность** – в WPF и в общеязыковой среде исполнения реализован ряд изменений, которые существенно повысили скорость исполнения WPF-приложений без каких либо изменений в коде. Например, заметно уменьшилось время загрузки приложения (особенно первой), анимация (особенно медленная) стала гораздо более плавной, привязка данных в некоторых ситуациях начала работать быстрее, а полупрозрачные окна (описанные в главе 8) теперь поддерживают аппаратное ускорение. Имеются и другие усовершенствования в области производительности, которые пользователь должен включать самостоятельно из-за ограничений

совместимости, например улучшенная виртуализация и отложенная прокрутка в многодетных элементах управления (см. главу 10 «Многодетные элементы управления»).

Усовершенствования в WPF 4

В версию WPF 4 дополнительно внесены следующие изменения:

- **Поддержка мультисенсорного ввода** – на компьютерах с поддержкой мультисенсорного ввода, работающих под управлением ОС Windows 7 или более поздней, элементы WPF могут получать различные события ввода: низкоуровневые данные, простые преобразования (например, поворот и масштабирование) или высокоуровневые (в том числе нестандартные) жесты. Все встроенные в WPF элементы управления модернизированы для работы с мультисенсорными устройствами ввода. Разработчики WPF воспользовались результатами работы над проектом Microsoft Surface (который, в свою очередь, основан на WPF). В итоге поддержка мультисенсорного ввода в WPF 4 совместима с версией 2 Surface SDK, что стало отличной новостью для всех, кто собирался разрабатывать приложения для Windows и Surface. См. главу 6 «События ввода: клавиатура, мышь, стилус и мультисенсорные устройства».
- **Полноценная поддержка прочих возможностей Windows 7** – мультисенсорный ввод – конечно, ценное добавление в Windows 7, но есть и много других, не требующих наличия специального оборудования и, следовательно, доступных широкому кругу пользователей. WPF обеспечивает оптимальный способ интеграции приложений с такими новыми возможностями панели задач, как списки переходов (Jump List) и многослойные значки (icon overlays), а также с обновленными стандартными диалоговыми окнами и многими другими нововведениями. См. главу 8.
- **Новые элементы управления** – WPF 4 включает такие элементы управления, как `DataGrid`, `Calendar` и `DatePicker`, которые первоначально появились в WPF Toolkit. См. главу 11.
- **Новые функции для анимации переходов** – появилось одиннадцать новых классов анимации, в том числе `BounceEase`, `ElasticEase` и `SineEase`, которые позволяют декларативно задавать замысловатые траектории анимации с настраиваемым ускорением и замедлением. Эти «переходные функции» и поддерживающая их инфраструктура впервые были представлены в Silverlight 3 и только потом вошли в WPF 4.
- **Улучшенная стилизация с помощью Visual State Manager** – менеджер визуальных состояний впервые появился в Silverlight 2; он предлагает новый способ организации визуальных элементов и их интерактивного поведения: в виде «визуальных состояний» и «переходов между состояниями». Это упрощает дизайнерам работу с элементами управления в таких инструментальных средах, как Expression Blend, а также позволяет создавать общие шаблоны для WPF и Silverlight.

- **Улучшенное поведение на границе пикселов** – WPF всегда стремилась соблюдать баланс между независимостью от разрешающей способности устройства (для чего требуется игнорировать границы физических пикселов) и обеспечением четкости визуальных элементов (что предполагает привязку к границам пикселов, особенно для мелких элементов). С самого начала WPF поддерживала свойство `SnapsToDevicePixels`, позволяющее принудительно осуществлять «привязку элементов к пикселам». Но использовать его было довольно сложно, а в некоторых случаях оно не давало никакого эффекта. Тогда в Silverlight был сделан шаг назад, в направлении обычной чертежной доски, и реализовано свойство `UseLayoutRounding`, которое работало более естественным образом. Теперь это свойство появилось и в WPF 4. Если задать его равным `true` для корневого элемента, то координаты этого элемента и всех его потомков будут округляться (с недостатком или с избытком), так чтобы они совпали с границами ближайших пикселов. В результате пользовательский интерфейс сохраняет способность к масштабированию, оставаясь при этом четким!
- **Более четкий текст** – стремление WPF обеспечить независимость от разрешающей способности устройства и масштабируемость интерфейса всегда терпело неудачу при отображении небольших фрагментов текста, которые превалируют в традиционных интерфейсах на экранах с разрешением 96 точек на дюйм (DPI). Это очень огорчало многих пользователей и разработчиков. Я даже говорил, что всегда смогу отличить интерфейс, созданный с помощью WPF, просто по размытости текста. В WPF 4 разработчики наконец-то решили эту проблему, предложив альтернативный способ визуализации текста, при котором текст выглядит так же четко, как выведенный с помощью GDI, но с сохранением почти всех преимуществ WPF. Например, этот режим используется в Visual Studio 2010 для отображения текстовых документов. Но поскольку новый способ визуализации имеет ряд ограничений, то включать этот режим следует явно. См. главу 11.
- **Усовершенствования в области развертывания приложений** – теперь клиентский профиль .NET может устанавливаться на одной машине с полным каркасом .NET и использоваться почти во всех сценариях, характерных для WPF-приложений. На самом деле проекты для .NET 4.0 в Visual Studio 2010 по умолчанию ориентированы на более компактный клиентский профиль.
- **Дальнейшее повышение производительности** – для максимального ускорения векторной графики WPF может кэшировать результаты рендеринга в виде растровых изображений и затем использовать их повторно. Этим поведением можно управлять с помощью свойства `CacheMode`; см. главу 15. Стимулом для многочисленных улучшений в области производительности послужило то, что WPF активно используется в Visual Studio 2010, но ощутить эффект теперь могут все WPF-приложения.

FAQ

Что будет добавлено в WPF после версии 4?

Во время написания этой книги никаких анонсов еще не было, но без особого риска можно предположить, что повышение производительности и дальнейшее сближение с Silverlight по-прежнему будут оставаться в центре внимания разработчиков WPF. Дополнительным источником информации о том, что может быть включено в ядро, служит WPF Toolkit. Речь может идти об элементах управления для построения графиков, а также об элементах `BreadcrumbBar`, `NumericUpDown` и др.

FAQ

Зависит ли поведение WPF от версии Windows?

Среди прочего в WPF реализованы API, которые относятся только к Windows 7 (или более поздним версиям), в частности мультисенсорный ввод и другие функции, описанные в главе 8. Кроме того, WPF ведет себя несколько иначе при запуске в Windows XP (самой старой версии Windows, которую поддерживает WPF). Например, не производится сглаживание для 3D-объектов.

И, конечно же, для элементов управления WPF по умолчанию применяются разные темы – в зависимости от операционной системы (Aero в Windows Vista и Windows 7, Luna в Windows XP).

Кроме того, в Windows XP используется старая модель драйверов, что может негативно сказаться на работе WPF-приложений. Модель драйверов в последних версиях Windows подразумевает виртуализацию и распределение ресурсов графических процессоров, что повышает общую производительность системы при исполнении нескольких программ, активно работающих с графикой. Запуск нескольких приложений WPF или DirectX в Windows XP может затормозить систему, но в более поздних версиях Windows таких проблем быть не должно.

Что такое Silverlight

Silverlight – это компактная, облегченная версия каркаса .NET, рассчитанная на насыщенные веб-приложения (в качестве альтернативы Adobe Flash и Flex). Подход Silverlight к созданию пользовательских интерфейсов такой же, как WPF, что дает массу преимуществ. Первая версия Silverlight вышла в 2007 году, а теперь, как и WPF, она находится на уровне четвертой версии. Silverlight 4 была выпущена в апреле 2010 года, через несколько дней после выхода WPF 4.

Взаимоотношения между WPF и Silverlight выглядят несколько запутанно, как и вопрос о том, когда применять одну технологию, а когда другую. Еще больше осложняет ситуацию тот факт, что WPF-приложения можно запускать в браузере (с помощью технологии XBAPs), то есть они практически «готовы