1 ВВЕДЕНИЕ В WPF

1.1 История

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 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.

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.

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 ненадежны-

ми. Основанные на них приложения не всегда доступны людям с ограниченными возможностями, плохо адаптируются к высокому разрешению и имеют другие визуальные огрехи.

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.

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.

• **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 подси-

стемой 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 WPFприложения (а не только высококлассные игры). Но WPF не требует обязательного наличия высокопроизводительной графической аппаратуры.

• **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). 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.

• **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).