**1. Введение в технологию WPF**

1. Описать предназначение технологии WPF. Сравнить эту технологию с технологией Windows Forms. В чем их принципиальное отличие?

**WPF (Windows Presentation Foundation) - это** программная платформа, разработанная компанией Microsoft для создания приложений, которые обладают графическим пользовательским интерфейсом (GUI-приложения).

**Предназначение WPF**

WPF предназначена для создания приложений с графическим интерфейсом, для создания современных пользовательских интерфейсов в интеллектуальных клиентских приложениях с поддержкой мультимедиа. Цель технологии – максимально облегчить и ускорить создание дружественных пользовательских интерфейсов. WPF спроектирована для использования только в контексте управляемой среды .NET и не содержит открытого API для взаимодействия из неуправляемого кода.

**Сравнение технологии с Windows Forms**

В основе WPF лежит DirectX, что делает данную технологию более производительной по сравнению с WinForms. WinForms, как и другие платформы для разработки пользовательских интерфейсов, были лишь обертками над довольно старыми библиотеками User32 и GDI32. Для повышения производительности была создана технология Microsoft DirectX. Для повышения производительности была создана технология Microsoft DirectX, которая и легла в основу WPF.

DirectX передает высокоуровневые конструкции непосредственно на видеокарту, что обеспечивает более высокую производительность. В WinForms всё выполнял CPU, что и обусловливало простой GPU. Независимо от типа интерфейса и сложности графики WPF всегда работает через DirectX, передавая основной объем работы по обработке графики GPU и освобождая CPU для других задач. Благодаря этому WPF поддерживает отображение сложных текстур, спецэффектов и трехмерной графики.

Также в WPF было проведено разделение труда программиста и дизайнера. Разметка окна приложения создается при помощи языка разметки XAML.

1. Основные преимущества WPF

* **Разделение труда программиста и дизайнера.** Раньше, до создания WPF, дизайнер разрабатывал макет, а программист решал, как реализовать этот макет программными средствами. В WPF декларативный пользовательский интерфейс полностью отделяется от бизнес-логики. **Пользовательский интерфейс создается с помощью языка** разметки **XAML**. Дизайнер может создать макет окна, потом программным способом получить разметку на языке XAML и передать её программисту. А тот в свою очередь будет писать логику приложения, работать с взаимодействием объектов в приложении.
* Аппаратное ускорение графики через **DirectX** обеспечивает **высокую производительность WPF**.
* WPF **базируется на векторной графике** и **не зависит от разрешения** устройства вывода. Векторная графика лучше растровой, т.к. векторные изображения могут быть неограниченно увеличены без потери качества.
* **Приложения** на WPF **легко масштабируются под разные мониторы с разным разрешением экрана.** При создании пользовательского интерфейса WPF использует web-подобную модель компоновки элементов управления за счет использования контейнеров.
* WPF предоставляет **богатую модель рисования** – используются базовые фигуры, блоки текста и прочие графические ингредиенты. Очень большой спектр возможностей оформления приложений.
* WPF предоставляет **более развитую текстовую модель** – стилизованный текст в любом месте интерфейса **(переносы, выравнивание, разбиение на колонки и т.п.)**
* WPF позволяет создавать множество анимаций.
* В WPF присутствует **полноценная поддержка аудио и видео**. Интеграция видео возможна в любой части пользовательского интерфейса.
* Поддержка **стилей** позволяет **стандартизировать форматирование** и многократно использовать его по всему приложению.
* Поддержка **шаблонов** позволяет изменить не только графическое представление элемента, но и его структуру. Система ресурсов позволяет **объединять шаблоны, стили, кисти, анимацию** и многие другие интерактивные элементы, что существенно **упрощает работу** с ними.
* WPF предоставляет возможности **трехмерной графики**, что позволяет создавать более яркий и интересный пользовательский интерфейс.
* Поддержка **команд** позволяет **сократить объем кода** и использовать одни и те же действия для нескольких элементов управления в различных местах программы.
* **Привязка данных** (data binding) - предоставляет WPF-приложениям **простой способ взаимодействия** с данными.

1. Краткое описание архитектуры технологии WPF.

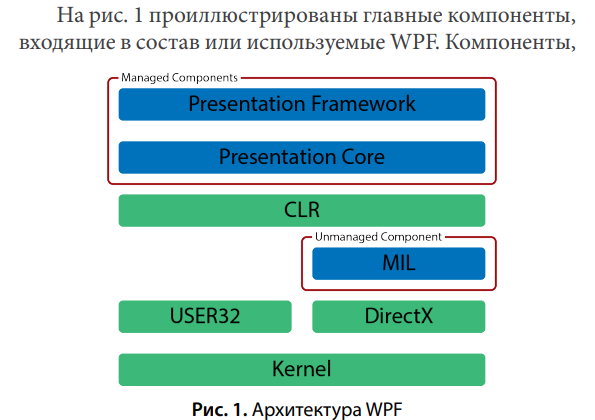
WPF разделяется на **два уровня**: **managed API** и **unmanaged API** (уровень интеграции с DirectX).

**Managed API** (управляемый API-интерфейс) содержит код, исполняемый под управлением CLR.

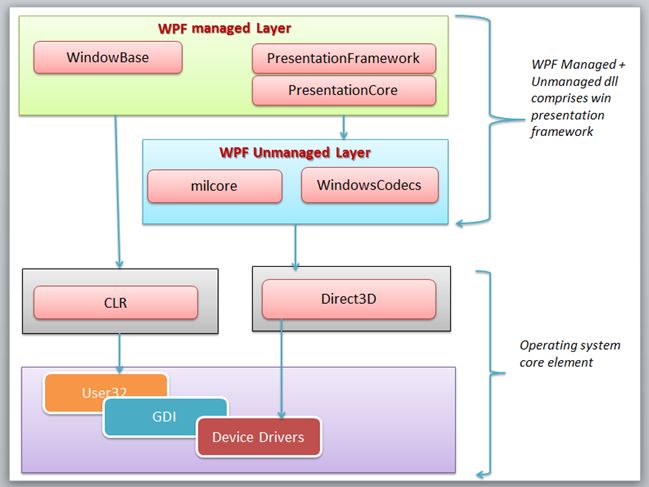
* Библиотека PresentationFramework.dll содержит высокоуровневые типы WPF (окна, элементы управления, панели, стили и т.п.)
* Библиотека PresentationCore.dll содержит базовые типы (UIElement и Visual), от которых унаследованы все фигуры и элементы управления
* Библиотека WindowsBase.dll содержит классы, которые применяются в WPF, но могут также использоваться и вне данной платформы

**Unmanaged API** используется для интеграции вышележащего уровня с DirectX

* Библиотека milcore.dll обеспечивает интеграцию компонентов WPF с DirectX, транслирует объекты .NET в текстуры и треугольники Direct3D
* Библиотека WindowsCodecs.dll предоставляет низкоуровневую поддержку для изображений в WPF



Media Integration Layer (MIL) способен получать разные виды информации (видео, масштабируемая графика, текст и т.д.) и выводить их все на одну поверхность рисования DirectX.



Еще ниже находятся компоненты операционной системы и DirectX, которые производят визуализацию компонентов приложения и выполняют прочую низкоуровневую обработку.

WPF не использует библиотеку user32.dll для рендеринга и визуализации, однако для ряда вычислительных задач, не включающих визуализацию, данная библиотека продолжает использоваться.