WPF (Windows Presentation Foundation) и Windows Forms (WinForms)

Особенности графики

- Windows Forms – для визуализации используется User32.dll и GDI/GDI+
- WPF для визуализации используется DirectX (аппаратная поддержка). В случае со старыми видеокартами используется программное вычисление эффектов.

WPF (Windows Presentation Foundation): API высокого уровня

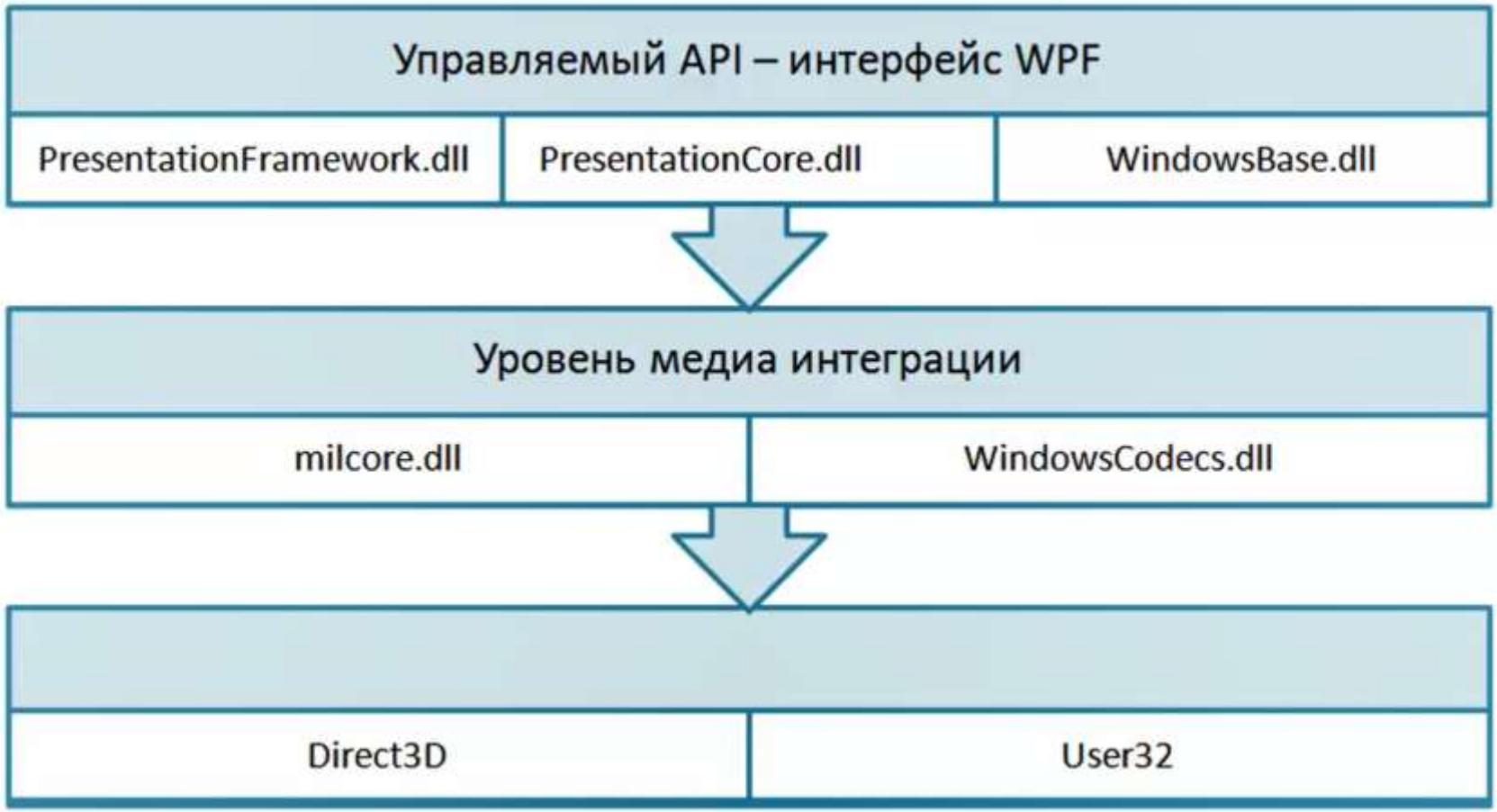
- Web подобная модель компоновки.
- Многофункциональная модель рисования.
- Форматированный текст.
- Анимация
- Аудио и Видео
- Стили и шаблоны
- Команды
- Декларативный пользовательский интерфейс
- Приложения со страничной организацией

WPF: независимое разрешение

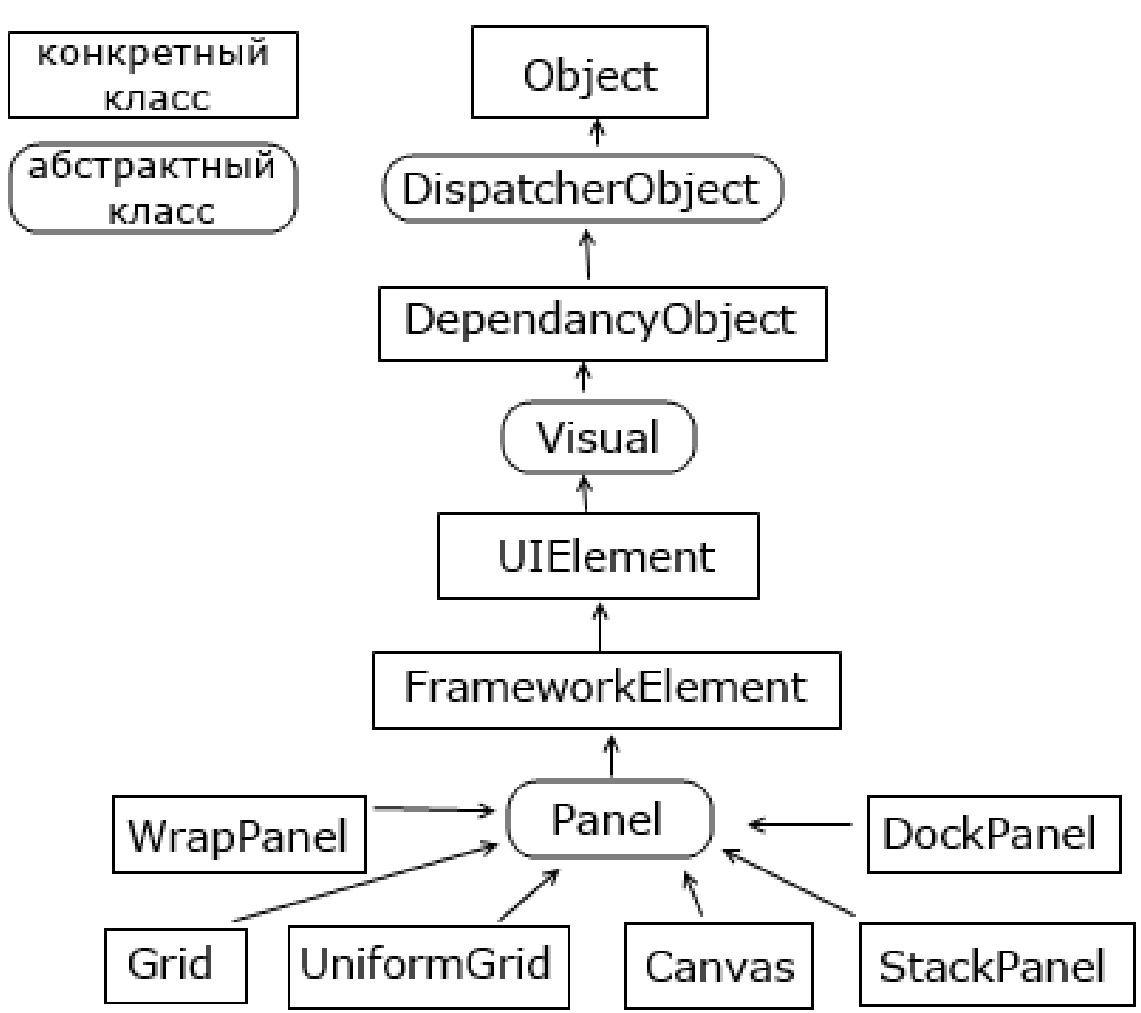
- Интерфейс в стандартных Windows приложениях не допускает масштабирования. На мониторе с высоким разрешением, размер окна уменьшается и становится менее удобным для работы. (Библиотека User32 не поддерживает полного масштабирования)

- WPF визуализирует все элементы пользовательского интерфейса самостоятельно, и если кнопка при проектировании в ширину была пять сантиметров, то при выполнении приложения на каком либо разрешении кнопка останется шириной в пять сантиметров. В основу масштабирования ставится системный параметр DPI (Dots Per Inch)

Архитектура WPF



Фундаментальные классы WPF



Фундаментальные классы WPF

System.Threading.DispatcherObject

В основе WPF лежит модель STA (Single-Thread Affinity), согласно которой за пользовательский интерфейс отвечает один поток. И чтобы пользовательский интерфейс мог взаимодействовать с другими потоками, WPF использует концепцию диспетчера - специального объекта, управляющего обменом сообщениями, через которые взаимодействуют потоки. Наследование типов от класса DispatcherObject позволяет получить доступ к подобному объекту-диспетчеру и и другим функциям по управлению параллелизмом.

System.Windows.DependencyObject

Наследование от этого класса позволяет взаимодействовать с элементами в приложении через их специальную модель свойств, которые называются свойствами зависимостей (dependency properties). Эта модель упрощает применение ряда особенностей WPF, например, привязки данных. Так, система свойств зависимостей отслеживает зависимости между значениями свойств, автоматически проверяет их и изменяет при изменении зависимости.

Фундаментальные классы WPF

System.Windows.Media.Visual

Класс Visual содержит инструкции, которые отвечают за отрисовку, визуализацию объекта.

Этот класс описывает основную логику для визуализации объекта на окне.

Также класс устанавливает связь между управляемыми библиотеками WPF и milcore.dll, которая осуществляет рендеринг

System.Windows.UIElement

Класс UIElement добавляет возможности по компоновке элемента, обработку событий и получение ввода.

Фундаментальные классы WPF

System.Windows.Controls.Control

Элемент управления – это компонент, который может взаимодействовать с пользователем. Данный класс вводит дополнительные свойства для управлением внешним видом контролов (изменение шрифта, фона и т.д.) Добавляет возможность поддержки шаблонов для быстрого задания стилей элементам управления.

System.Windows.Controls.ContentControl

Это базовый класс для всех элементов управления, которые содержат единственную порцию содержимого. Примеры некоторых производных классов: Label, Button, Window и т.д.

Фундаментальные классы WPF

System.Windows.Controls.ItemControl

Класс описывает все элементы управления, которые содержат в себе коллекцию элементов (TreeView, ListBox, ComboBox)

System.Windows.Controls.Panel

Это базовый класс для всех элементов компоновки – объекты способные хранит один и более дочерних элементов.

XAML

XAML (англ. *e**X**tensible **A**pplication **M**arkup **L**anguage* — расширяемый язык разметки приложений произносится (*зэмл*)) — основанный на XML язык разметки для декларативного программирования приложений, разработанный Microsoft.

Основы XAML

- Каждый элемент в документе XAML отображается как экземпляр класса .NET. Имя элемента в точности соответствует имени класса.Например элемент <Button> сообщает WPF что должен быть создан объект Button.

- Как и любой XML-документ код XAML допускает вложение одного элемента внутрь другого. Если вы видите элемент Button внутри элемента Grid , то пользовательский интерфейс включает Grid, содержащий внутри себя Button.

- Свойства каждого класса можно устанавливать через атрибуты. Тем не менее, в некоторых ситуациях атрибуты не достаточно мощны, чтобы справится с этой работой. В этих случаях понадобятся вложенные дескрипторы со специальным синтаксисом.

XAML

Загрузка и компиляция XAML

В общем случае существует три разных стиля кодирования которые могут применятся при создании приложения WPF:

- Только код. Это традиционный подход ,используемы в Visual Studio для приложений Windows Forms. Пользовательский интерфейс в нем генерируется операторами кода.

- Код и не компилированная разметка (XAML). Это специализированный подход, который имеет смысл в определенных сценариях, когда нужны исключительно динамичные пользовательские интерфейсы.

- Код и компилированная разметка (BAML). Это предпочтительный подход для WPF, поддерживаемый Visual Studio. Для каждого окна создается шаблон XAML, и этот код XAML компилируется в BAML, после чего встраивается в конечную сборку. Во время выполнения скомпилированный BAML извлекается и используется для регенерации пользовательского интерфейса.

Философия компоновки WPF

- Окно в WPF может содержать только один элемент.
- Размеры элементов не должны быть заданы явно.
- Элементы не отражают свое положение с помощью экранных координат.
- Контейнеры компоновки разделяют доступное пространство между своими дочерними элементами.
- Контейнеры компоновки допускают вложения.

Процесс компоновки WPF

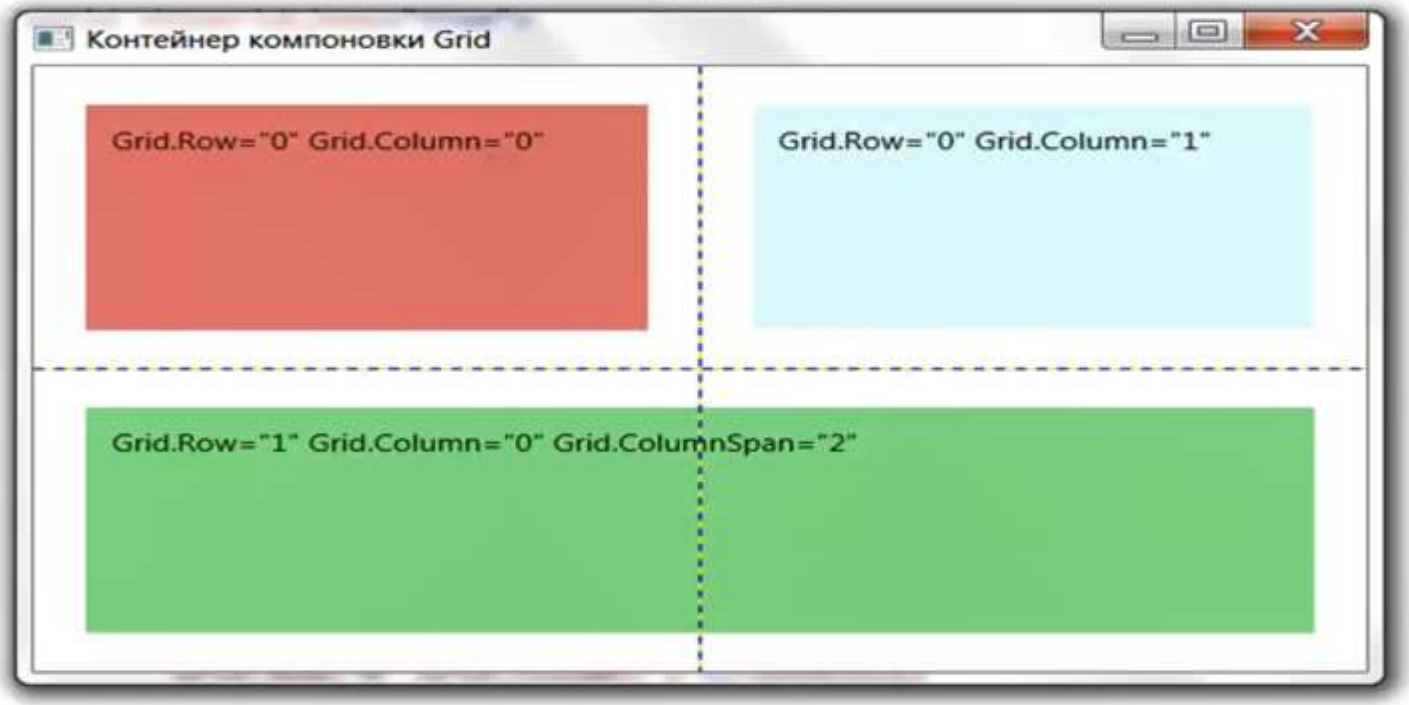
Процесс компоновки делится на две стадии:

- Измерение - контейнер компоновки просматривает свои дочерние элементы и запрашиваетy них предпочтительный размер.
- Упорядочивание – контейнер компоновки помещает элементы управления в соответствующие позиции.

WPF: Контейнеры и компоновки

Базовые контейнеры компоновки WPF

Grid



Выстраивает элементы в строки и колонки невидимой таблицы. Это один из наиболее гибких и широко используемых контейнеров компоновки.

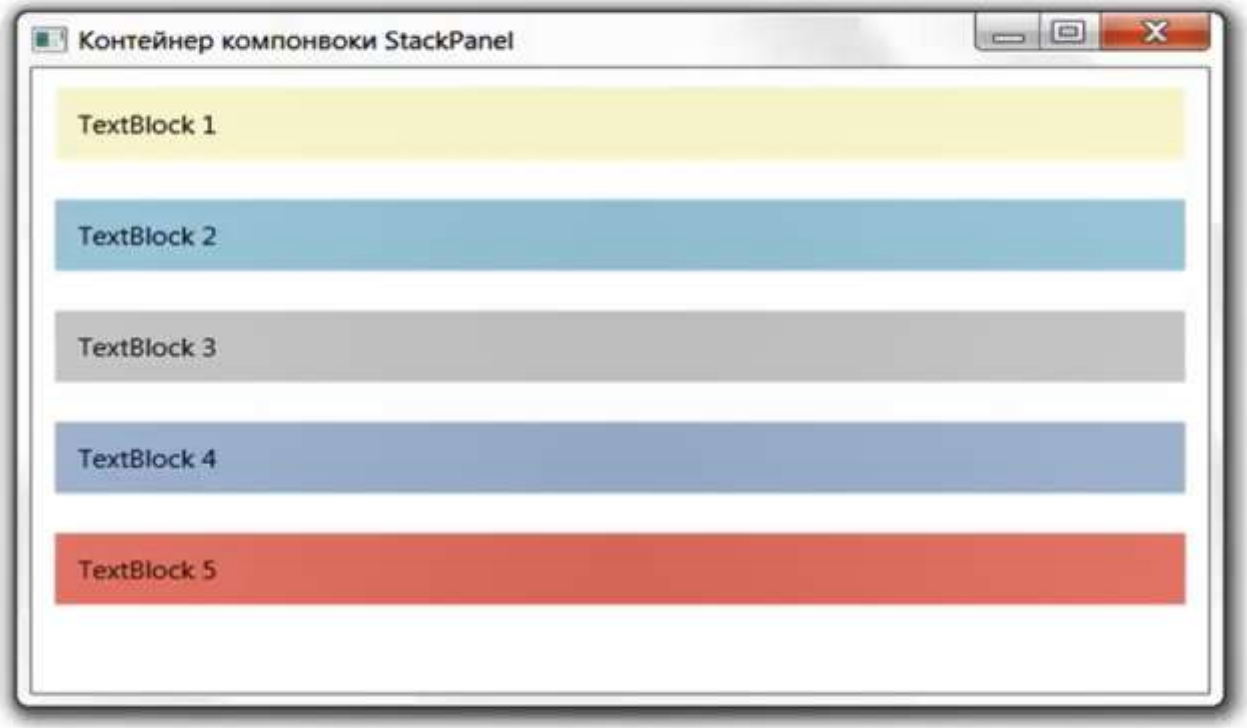
Компоновка


```
<Grid.RowDefinitions>
        <RowDefinition />
        <RowDefinition />
        <RowDefinition />
        <RowDefinition />
        <RowDefinition />
    </Grid.RowDefinitions>

<Grid.ColumnDefinitions>
        <ColumnDefinition />
        <ColumnDefinition />
        <ColumnDefinition />
        <ColumnDefinition />
    </Grid.ColumnDefinitions>
```

Базовые контейнеры компоновки WPF

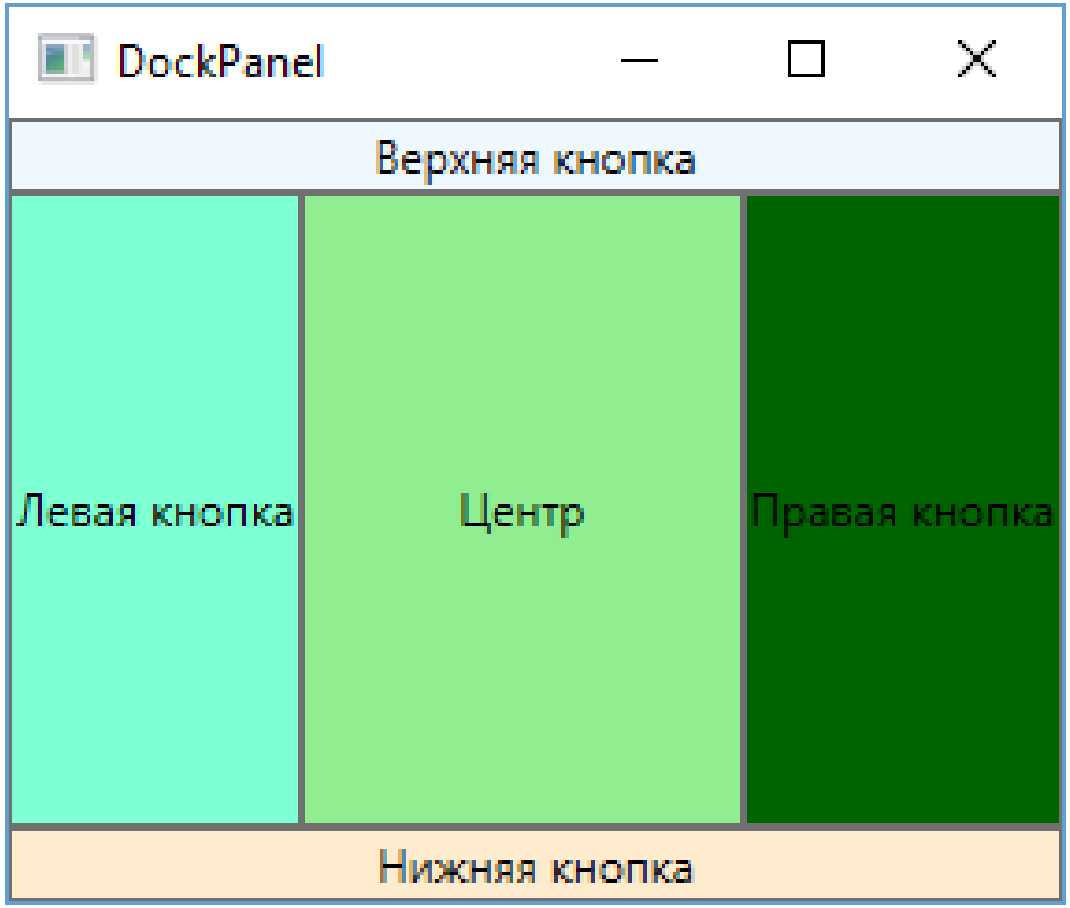
StackPanel



Размещает элементы в горизонтальном или вертикальном стеке. Этот контейнер компоновки обычно используется в небольших разделах крупного и более сложного окна.

WPF: Контейнеры и компоновки

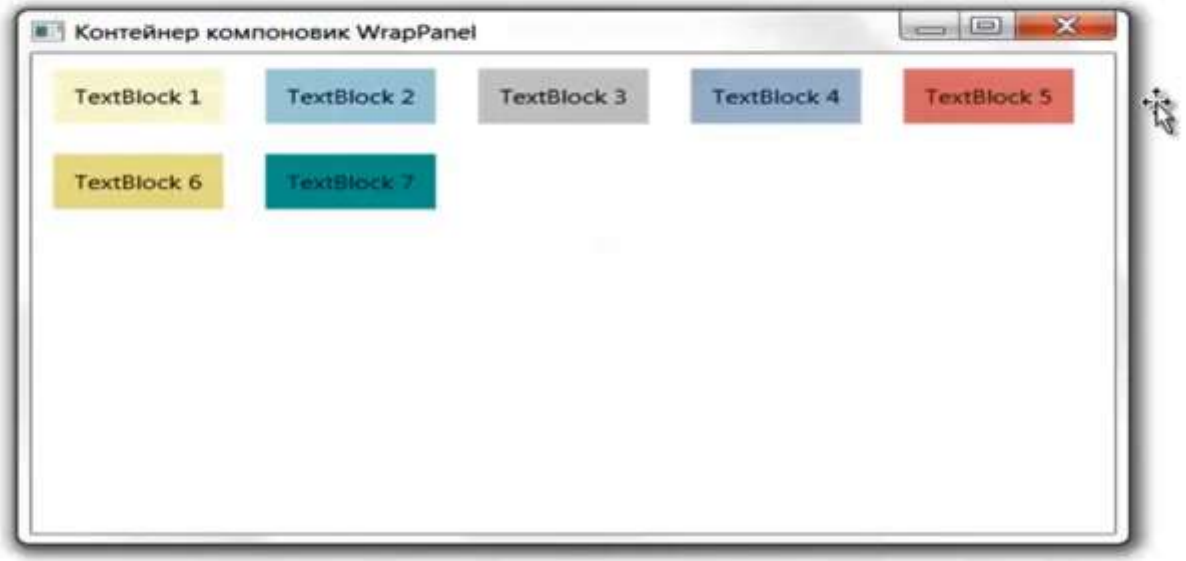
Базовые контейнеры компоновки WPF




```
<DockPanel LastChildFill="True">
    <Button DockPanel.Dock="Top"
Background="AliceBlue" Content="Верхняя
кнопка" />
    <Button DockPanel.Dock="Bottom"
Background="BlanchedAlmond"
Content="Нижняя кнопка" />
    <Button DockPanel.Dock="Left"
Background="Aquamarine" Content="Левая
кнопка" />
    <Button DockPanel.Dock="Right"
Background="DarkGreen" Content="Правая
кнопка" />
    <Button Background="LightGreen"
Content="Центр" />
  </DockPanel>
```

Базовые контейнеры компоновки WPF

WrapPanel



Размещает элементы в последовательностях строк с переносом. В горизонтальной ориентации WrapPanel располагает элементы в строке слева направо, затем переходит к следующей строке. В вертикальной ориентации WrapPanel располагает элементы сверху вниз, используя дополнительные колонки для дополнения оставшихся элементов

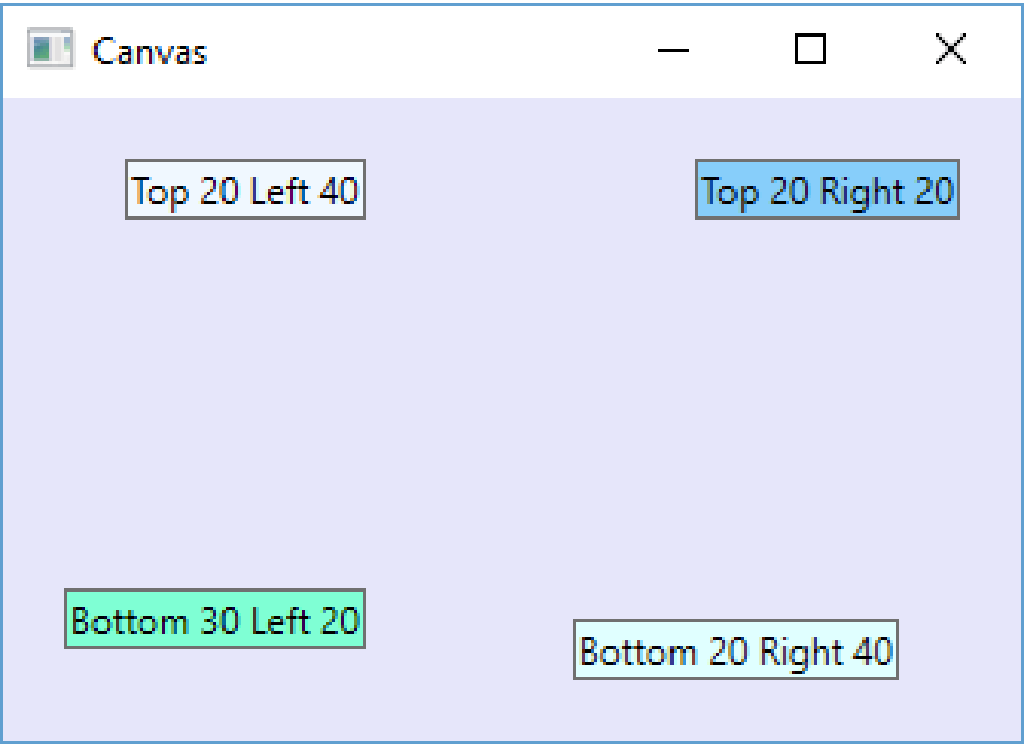
Базовые контейнеры компоновки WPF

UniformGrid

Помещает элементы в невидимую таблицу, устанавливая одинаковый размер для всех ячеек. Данный контейнер компоновки используется нечасто.

WPF: Контейнеры и компоновки

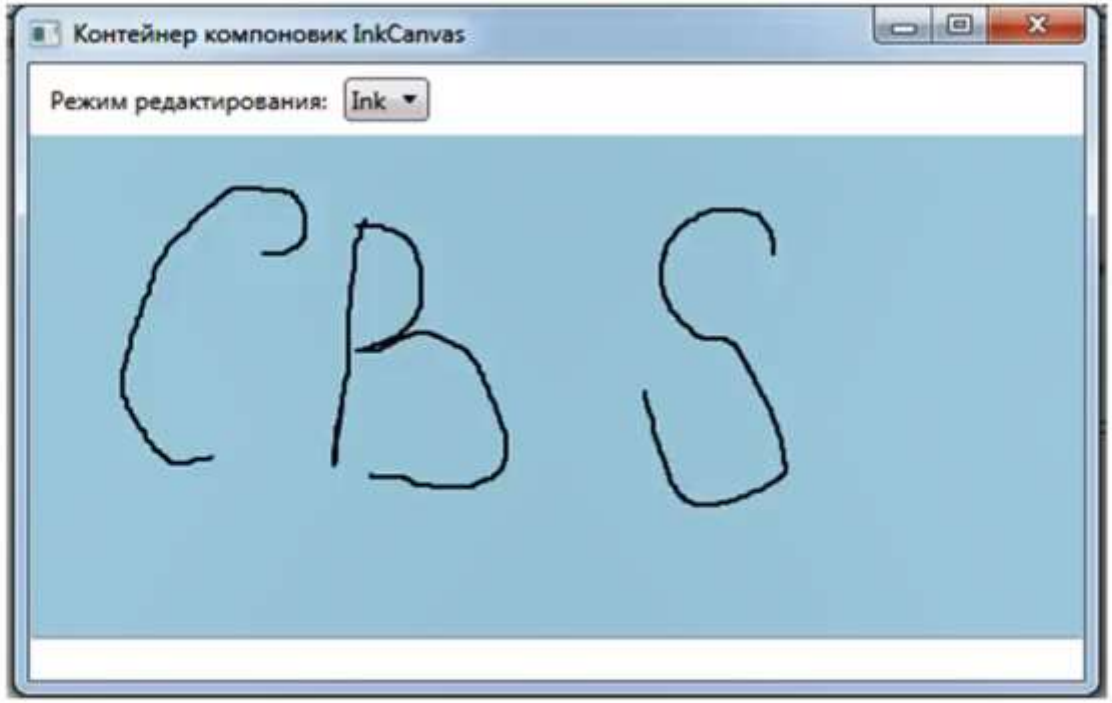
Canvas -контейнер компоновки WPF




```
<Grid>
    <Canvas Background="Lavender">
        <Button Background="AliceBlue"
Content="Top 20 Left 40" Canvas.Top="20"
Canvas.Left="40" />
        <Button Background="LightSkyBlue"
Content="Top 20 Right 20" Canvas.Top="20"
Canvas.Right="20"/>
        <Button Background="Aquamarine"
Content="Bottom 30 Left 20"
Canvas.Bottom="30" Canvas.Left="20"/>
        <Button Background="LightCyan"
Content="Bottom 20 Right 40"
Canvas.Bottom="20" Canvas.Right="40"/>
    </Canvas>
</Grid>
```

Базовые контейнеры компоновки WPF

InkCanvas



В WPF также имеется элемент InkCanvas, главное предназначение которого заключается в обеспечении перьевого ввода, а также считывания жестов пользователя.

Свойства InkCanvas

Имя	Описание
Ink	InkCanvas позволяет пользователю рисовать аннотаци. Когда поьлзователь рисует мышью или пером,появляются штрихи.
GestureOnly	В этом режиме InkCanvas не позволяет пользователю рисовать аннотации,но считывает жесты произведенные пользователем с помощью мыши или пера.
InkAndGesture	InkCanvas позволяет пользователю рисовать штриховые аннотации и также распознает предопределенные жесты.
EraseByStroke	InkCanvas удаляет весь штрих при щелчке.
EraseByPoint	InkCanvas удаляет часть штриха(точку штриха) при щелчке на соответствующей его части.
Select	InkCanvas позволяет пользователю выбирать элементы, хранящиеся в коллекции Children. Как только элемент выбран, его можно перемещать, изменять размер или удалять.
None	InkCanvas игнорирует ввод с помощью мыши или пера.

Вложение контейнеров компоновки

Панели StackPanel, WrapPanel и DockPanel редко используются сами по себе. Вместо этого они применяются для формирования частей интерфейса. Например панель DockPanel можно использовать для размещения разных контейнеров StackPanel и WrapPanel в соответствующих областях окна.

Вложение контейнеров компоновки

Панели StackPanel, WrapPanel и DockPanel редко используются сами по себе. Вместо этого они применяются для формирования частей интерфейса. Например панель DockPanel можно использовать для размещения разных контейнеров StackPanel и WrapPanel в соответствующих областях окна.

WPF: Контейнеры и компоновки

Свойства для компоновки

Имя	Описание
HorizontalAligment	Определяет как дочерние элементы позиционируются внутри контейнера когда есть дополнительное пространство по горизонтали Значения: Left, Right, Center, Stretch
VerticalAligment	Определяет как дочерние элементы позиционируются внутри контейнера когда есть дополнительное пространство по вертикали Значения: Left, Right, Center, Stretch
Margin	Добавление пустого пространства вокруг элемента. В коде это экземпляр структуры System.Windows.Thickness
MinWidth и MinHeight	Минимальные значения по высоте и ширине
MaxWidth и MaxHeight	Максимальное значения по высоте и ширине
Width и Height	Явный размер