

Разработка приложений на платформе .NET

Лекция 19

Графика в WPF

Сегодня

- Кисти
- Фигуры
- Геометрии
- Трансформации

Кисть (Brush)

- Заполняет область
 - Определяет, чем заполняются внутренности элемента, фигуры, границу элемента и т.п.
- Пример свойств элементов типа **Brush**:
 - **Background** – фон элемента
 - **Foreground** – передний план элемента
 - **BorderBrush** – границы элемента
 - **OpacityMask** – прозрачность элемента (используется только установка прозрачности кисти)
 - **Fill** – фигуры изнутри
 - **Stroke** – окрашивает края фигуры
- Кисти поддерживают частичную прозрачность
- Класс **SystemBrushes** – предоставляет доступ к кистям, которые используют цвета определенные в настройках **Windows** текущего компьютера. Класс **SystemColors** содержит системные цвета

Кисти

- **SolidColorBrush** – однотонная сплошная кисть
- **LinearGradientBrush** – представляет линейный градиент
- **RadialGradientBrush** – представляет радиальный градиент
- **ImageBrush** – кисть использует изображение
- **DrawingBrush** – кисть, заданная с помощью векторного или (и) растрового изображения
- **VisualBrush** – кисть заданная другим Visual элементом
- **BitmapCacheBrush** – кэшированная кисть заданная другим Visual элементом

SolidColorBrush

- Заливка сплошным цветом
- Свойство **Color** – определяет цвет кисти
- Имеется predefined набор кистей
 - В XAML: `<Button Background="Aqua"/>`
 - В C#: `button1.Background = Brushes.Beige;`
 - `button1.Background = new SolidColorBrush(Colors.Beige);`
- Задание цвета покомпонентно:
 - В XAML: `<Button Background="#FFFF0000"/>`
 - Цвет задается в формате #AARRGGBB (AA, RR, GG, BB – шестнадцатеричные представления прозрачности, красной, зеленой и синей компоненты цвета). FF – значение компоненты прозрачности означает полностью не прозрачную кисть, 00 – полностью прозрачную кисть
 - В C#: `button1.Background = new SolidColorBrush(Color.FromArgb(255, 255, 0, 0));`
- Использование системных кистей: `button1.Background = SystemColors.ControlBrush;`
- Примеры:

```
<Button Foreground="#ADF4A523">
  <Button.Background>
    <SolidColorBrush>
      <SolidColorBrush.Color>
        <Color A="255" R="0" G="0" B="255"/>
      </SolidColorBrush.Color>
    </SolidColorBrush>
  </Button.Background>
</Button>
```

Демонстрация

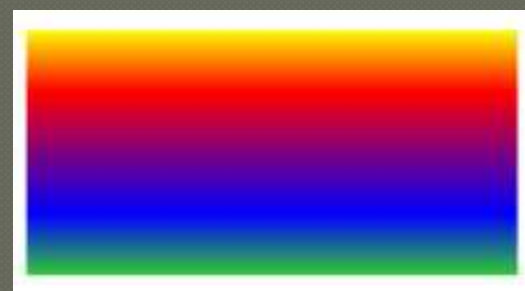
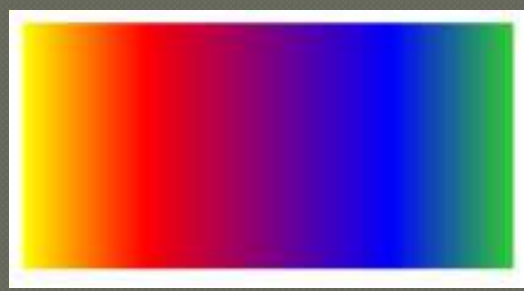
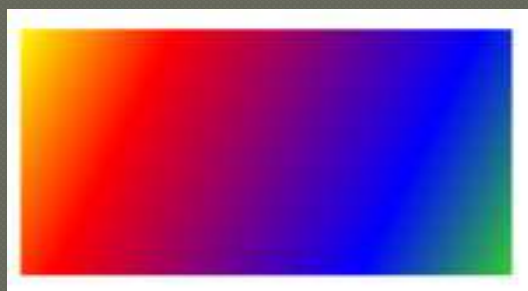
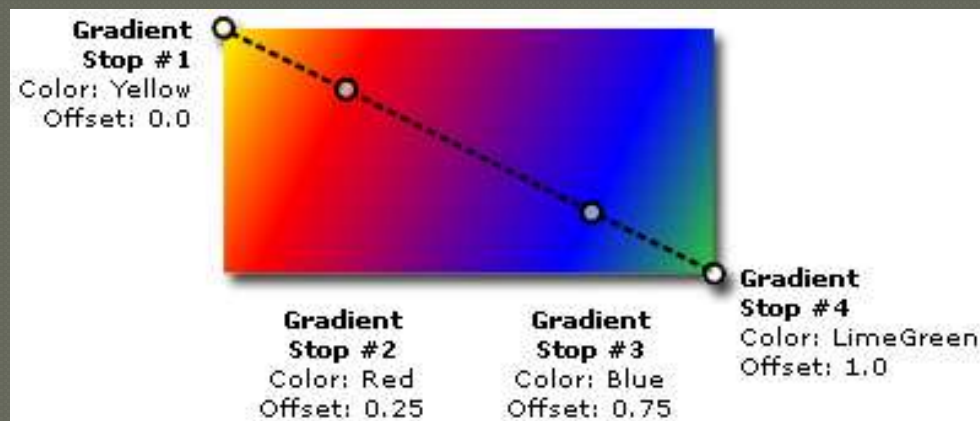
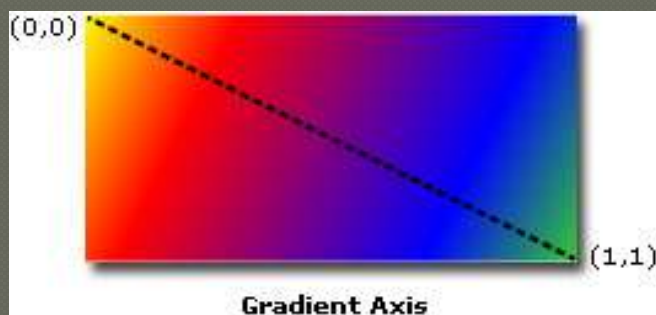
SolidColorBrush

LinearGradientBrush

- Кисть с градиентом
- Задаются несколько точек, где отображаются указанные чистые цвета
- **StartPoint, EndPoint** – точки начала и окончания градиента в относительных координатах
 - Точка (0,0) – левый верхний угол, (1,1) – правый нижний
- **GradientStops** коллекция объектов **GradientStop**, которые задают точки (смещение) в которой должен отображаться указанный цвет
 - **GradientStop.Offset** – смещение на отрезке между начальной и конечной точками
 - **GradientStop.Color** – заданный цвет

```
<Button Height="50" Width="200">  
  <Button.Background>  
    <LinearGradientBrush StartPoint="0,1" EndPoint="1,1">  
      <GradientStop Color="Red" Offset="0"/>  
      <GradientStop Color="Green" Offset="0.5"/>  
      <GradientStop Color="Violet" Offset="1"/>  
    </LinearGradientBrush>  
  </Button.Background>  
</Button>
```

LinearGradientBrush



Демонстрация

LinearGradientBrush

RadialGradientBrush

- Кисть с радиальным градиентом
- Задаются нескольких точек, где отображаются указанные чистые цвета
- **GradientOrigin** – задает центр градиента (в относительных координатах), от которого расходитя радиальный градиент
 - Точка (0,0) – левый верхний угол, (1,1) – правый нижний
- **RadiusX, RadiusY** – границы градиента
- **GradientStops** коллекция объектов **GradientStop**, которые задают точки (смещение) в которой должен отображаться указанный цвет
 - **GradientStop.Offset** – смещение на отрезке между начальной и конечной точками
 - **GradientStop.Color** – заданный цвет

```
<Ellipse Height="200" Width="200">  
  <Ellipse.Fill>  
    <RadialGradientBrush GradientOrigin="0.7, 0.3" RadiusX="1" RadiusY="1" >  
      <GradientStopCollection>  
        <GradientStop Color="White" Offset="0"/>  
        <GradientStop Color="Black" Offset="1"/>  
      </GradientStopCollection>  
    </RadialGradientBrush>  
  </Ellipse.Fill>  
</Ellipse>
```

- **SpreadMethod** – задает способ заполнения цветом при выходе за границы градиента: **Pad**, **Reflect**, **Repeat**. Задаются относительно центра **Center**

Демонстрация

RadialGradientBrush

ImageBrush

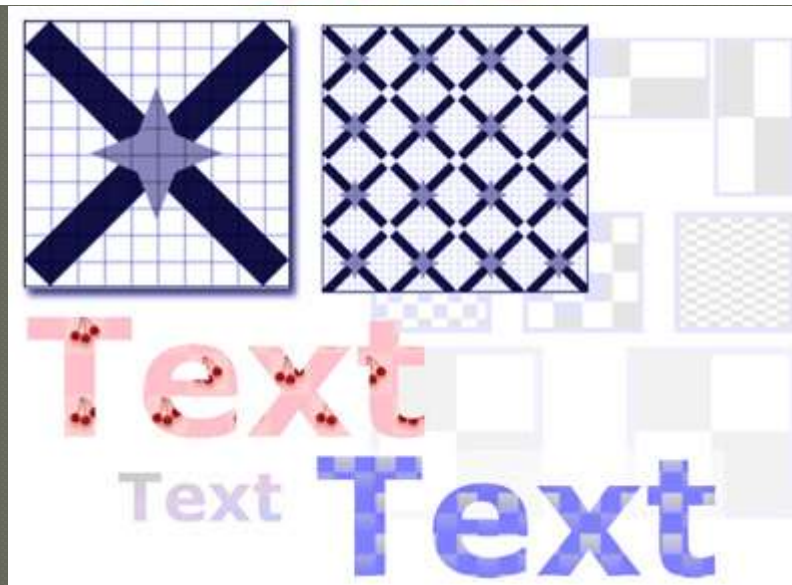
- Заполняет область растровым изображением
- **ImageSource** – задает изображение
- **Stretch** – указывает способ заполнения области изображением (None, Uniform, UniformToFill, Fill)
- **Viewbox** – обрезает изображение, а оставшуюся часть растягивает

```
<Ellipse Height="200" Width="200">  
  <Ellipse.Fill>  
    <ImageBrush ImageSource="Hydrangeas.jpg" Stretch="None"  
      Viewbox="0.4,0.4,0.6,0.6"/>  
  </Ellipse.Fill>  
</Ellipse>
```

- Задание мозаичности
 - **TileMode** – задает мозаичность заполнения области (Tile, Non, FlipX, FlipY, FlipXY)
 - **Viewport** – задает тиражируемую область

```
<Ellipse Height="200" Width="200">  
  <Ellipse.Fill>  
    <ImageBrush ImageSource="Hydrangeas.jpg" ViewportUnits="Absolute"  
      Viewport="0,0, 100,100" TileMode="Tile"/>  
  </Ellipse.Fill>  
</Ellipse>
```

ImageBrush, DrawingBrush



Демонстрация

ImageBrush

VisualBrush

- Заполняет область отображением любого Visual объекта, т.е. любого элемента в визуальном дереве
- Изображение всегда актуальное, т.е. изменяется с изменением отображения исходного элемента
- Изображение не интерактивное, т.е. не поддерживает взаимодействие с пользователем
- **Visual** – задает отображаемый элемент

```
<TextBox Name="tb" Text="Привет"/>
```

```
<Rectangle>
```

```
  <Rectangle.Fill>
```

```
    <VisualBrush Visual="{Binding ElementName=tb}"/>
```

```
  </Rectangle.Fill>
```

```
</Rectangle>
```

Демонстрация

VisualBrush

Прозрачность

- С помощью кистей также можно задавать неоднородную прозрачность любых элементов
- **Opacity** – задает сплошной процент прозрачности. 1 – полностью не прозрачный элемент, 0 – полностью прозрачный элемент
- **OpacityMask** – задает кисть, по которой определяется прозрачность. Используется только компонента прозрачности кисти

```
<Button Height="50" Width="200">  
  <Button.OpacityMask>  
    <LinearGradientBrush StartPoint="0,1" EndPoint="1,1">  
      <GradientStop Color="#FF000000" Offset="0"/>  
      <GradientStop Color="#11000000" Offset="0.5"/>  
    </LinearGradientBrush>  
  </Button.OpacityMask>  
</Button>
```

Демонстрация

Прозрачность

Сегодня

- Кисти
- Фигуры
- Геометрии
- Трансформации

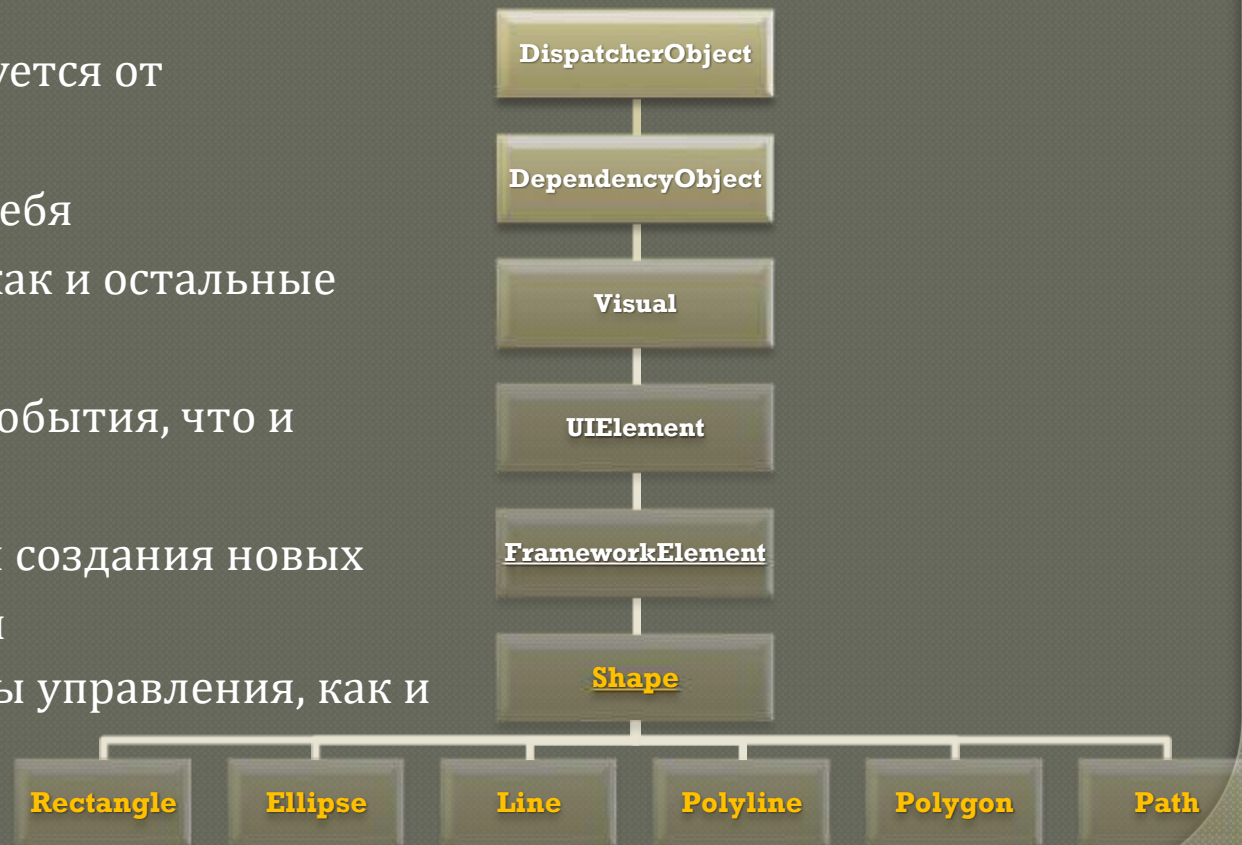
Фигуры

- Все фигуры наследуются от **System.Windows.Shapes**
- Представляют собой геометрические примитивы

Поскольку **Shape** наследуется от **FrameworkElement**, то:

- Фигуры рисуют сами себя
- Организованы также как и остальные элементы
- Поддерживают те же события, что и остальные элементы
- Могут применяться для создания новых элементов управления

Т.е. это такие же элементы управления, как и остальные



Shape

○ Свойства класса Shape:

- **Fill** – объект Brush, окрашивающий фигуру изнутри
- **Stroke** – объект Brush, окрашивающий границы фигуры
- **StrokeThickness** – задает толщину границы
- **Stretch** – указывает на заполнение фигурой занимаемого ею пространства. Значения: **None**, **Uniform**, **Fill**, **UniformToFill**
- **Width**, **Height**, **Margin** – аналогичные
- Система компоновки такая же, как и у обычных элементов

Фигуры

◉ **Rectangle** – прямоугольник

- `<Rectangle Height="100" Width="200" Fill="Blue"/>`
- Прямоугольник с закругленными углами
- `<Rectangle Height="100" Width="200" Fill="Blue" RadiusX="10" RadiusY="10"/>`

◉ **Ellipse** – ЭЛЛИПС

- `<Ellipse Height="200" Width="200" Fill="Red"/>`

◉ **Line** – ЛИНИЯ

- `<Line Stroke="Aqua" X1="0" Y1="0" X2="100" Y2="300"/>`
- `StartLineCap`, `EndLineCap` – границы линии
- `StrokeDashArray`, `StrokeDashCap`, `StrokeDashOffset` – задают пунктирность линии

◉ **Polyline** – ломаная линия. Если есть внутреннее содержимое, окрашивает его кистью, заданной в `Fill`

- `<Polyline Stroke="Green" Points="100, 200, 200, 200, 200, 100, 300, 200, 300, 100"/>`

◉ **Polygon** – ломаная линия, но в отличие от `Polyline`, соединяет первую и последнюю точки. Окрашивает внутреннее содержимое кистью, заданной в `Fill`

- `<Polygon Stroke="Green" Fill="Beige" Points="100, 200, 200, 200, 200, 100, 300, 200, 300, 100"/>`

◉ **Path** – сложная форма. Объединяет коллекцию классов `Geometry`

Демонстрация

Фигуры

Сегодня

- Кисти
- Фигуры
- Геометрии
- Трансформации

Geometry

- **Path** – сложная форма. Объединяет коллекцию классов **Geometry** и добавляет обработку событий.
- Класс **Geometry** содержит данные для отображения, но не поддерживает обработку событий.
- Задаются в независимых от разрешения координатах
- Не являются элементами. Поэтому не могут сами отображаться
- Задают геометрию, а не отображение
- Наследники **Geometry**:
 - **RectangleGeometry** – прямоугольник
 - `<RectangleGeometry Rect="100,100,50,30"/>`
 - **EllipseGeometry** – эллипс
 - `<EllipseGeometry Center="100,50" RadiusX="30" RadiusY="50" />`
 - **LineGeometry** – линия
 - `<LineGeometry StartPoint="10,20" EndPoint="100,200"/>`
 - **PathGeometry** – сложная геометрия
 - **CombinedGeometry** – логические операции над фигурами
 - **GeometryGroup** – группа объектов

```
<Path>
```

```
  <Path.Data>
```

```
    <EllipseGeometry RadiusX="50" RadiusY="100" Center="200 ,200"/>
```

```
  </Path.Data>
```

```
</Path>
```

Группирование

- Класс **GeometryGroup**
- Объединение нескольких геометрий в одну

```
<Path Fill="Gray">  
  <Path.Data>  
    <GeometryGroup FillRule="Nonzero">  
      <EllipseGeometry RadiusX="100" RadiusY="20" Center="200, 200"/>  
      <RectangleGeometry Rect="180, 200, 40, 100"/>  
    </GeometryGroup>  
  </Path.Data>  
</Path>
```

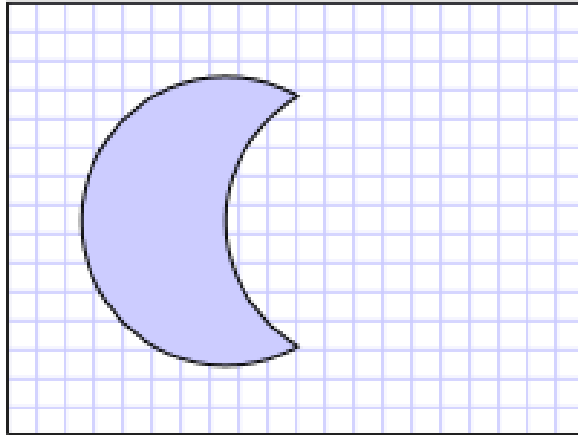
Комбинирование

- Класс **CombinedGeometry**
- Комбинирование двух геометрий в одну.
- Применяются правила комбинирования
 - Задается свойством **GeometryCombineMode**

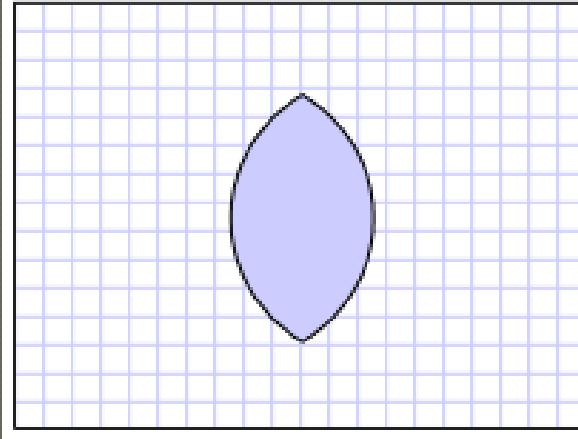
```
<Path Fill="Gray">
  <Path.Data>
    <CombinedGeometry GeometryCombineMode="Xor">
      <CombinedGeometry.Geometry1>
        <EllipseGeometry RadiusX="100" RadiusY="100" Center="100,100"/>
      </CombinedGeometry.Geometry1>
      <CombinedGeometry.Geometry2>
        <EllipseGeometry RadiusX="100" RadiusY="100" Center="200,100"/>
      </CombinedGeometry.Geometry2>
    </CombinedGeometry>
  </Path.Data>
</Path>
```

GeometryCombineMode

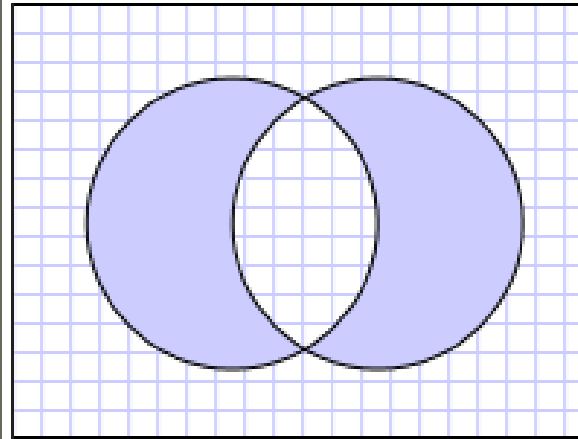
Exclude



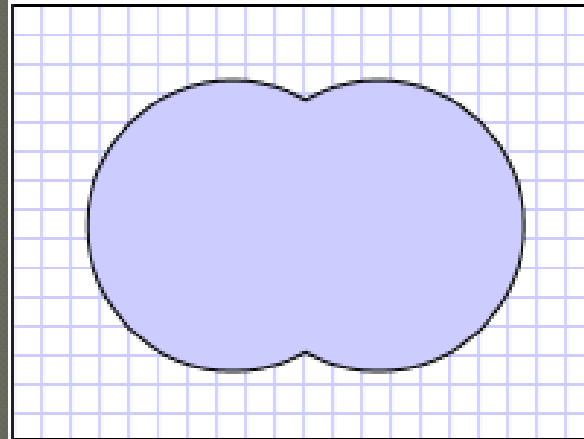
Intersect



Xor



Union



Демонстрация

Геометрии

Ограничение элементов

- Свойство **Clip** любого элемента управления принимает геометрию и обрезает элемент управления в соответствии с этой геометрией

```
<Button Height="50" Width="200">  
  <Button.Clip>  
    <EllipseGeometry RadiusX="30" RadiusY="30" Center="100,25"/>  
  </Button.Clip>  
  <Ellipse Fill="Red" Width="100" Height="40"/>  
</Button>
```

Демонстрация

Круглая кнопка

Сегодня

- ◎ Кисти
- ◎ Фигуры
- ◎ Геометрии
- ◎ Трансформации

Преобразования

- Преобразование (**Transform**) преобразует двумерные координаты при помощи матрицы 3*3
- Влияет на размещение и отрисовку
- Применимо к любому элементу

- Базовый класс всех трансформаций
 - `System.Windows.Media.Transform`

- Виды преобразований
 - **RenderTargetTransform** - влияет только на отображение
 - **LayoutTransform** - влияет еще и на размещение

```
<Button Height="50" Width="200">
```

```
  <Button.RenderTransform>
```

```
    ....
```

```
  </Button.RenderTransform>
```

```
</Button>
```

```
<Button Height="50" Width="200">
```

```
  <Button.LayoutTransform >
```

```
    ....
```

```
  </Button.LayoutTransform >
```

```
</Button>
```

Типы преобразований

○ **RotateTransform**

- Поворот вокруг (CenterX, CenterY) на Angle градусов по часовой стрелке

○ **ScaleTransform**

- Масштабирование вокруг (CenterX, CenterY) на (ScaleX, ScaleY)

○ **TranslateTransform**

- Перенос на вектор (TranslateX, TranslateY)

○ **SkewTransform**

- Преобразование перекоса (AngleX, AngleY)

○ **MatrixTransform**

- Преобразование с произвольной матрицей (Matrix)

○ **TransformGroup**

- Суперпозиция преобразований. Преобразуется WPF в MatrixTransform

Примеры преобразований

```
<Button Height="50" Width="200">  
  <Button.RenderTransform>  
    <RotateTransform CenterX="100" CenterY="25"  
      Angle="45"/>  
  </Button.RenderTransform>  
</Button>
```

Группа преобразований:

```
<Button Height="50" Width="200">  
  <Button.RenderTransform>  
    <TransformGroup>  
      <RotateTransform CenterX="100" CenterY="25" Angle="45"/>  
      <SkewTransform AngleX="45" AngleY="0" CenterX="100"  
        CenterY="25"/>  
    </TransformGroup>  
  </Button.RenderTransform>  
</Button>
```

Демонстрация

Трансформации