Кисти

Рассматривая выше элементы и их свойства, были упомянуты такие свойства, как **Background** и **Foreground** и назначение им определенного цвета Background="Blue". Но если посмотреть чуть глубже, то для установки цвета нам нужен объект класса System.Windows.Media.Brush. Значение "Blue" в данном случае является свойством класса Brushes, которое инкапсулирует объект SolidColorBrush. Например, в коде мы можем установить цвет так
button1.Background=Brushes.Blue
А класс SolidColorBrush является кистью или наследником класса Brush, с помощью которого, таким образом, можно устанавливать свойства **Background**, **Foreground** и **BorderBrush**.

WPF поддерживает целый ряд кистей:

SolidColorBrush заливает содержимое сплошным цветом

LinearGradientBrush - градиентная кисть, представляет плавный переход от одного цвета к другому

RadialGradientBrush- градиентная кисть, плавно распределяющая заданные цвета от центральной точки к внешним границам.

ImageBrush в качестве заполнителя использует не цвет, а изображение

DrawingBrush с помощью свойства Drawing опеределяет рисунок, включающий, геометрические фигуры, другие элементы и т.д., служащее заполнителем.

VisualBrush в качестве заполнителя имеет какой-либо элемент управления или его часть

SolidColorBrush

Задает цвет для сплошной заливки:

```
<TextBlock Height="20" Width="160" VerticalAlignment="Top" TextAlignment="Center" Text="SolidCol
            <TextBlock.Background>
                <SolidColorBrush Color="Blue" Opacity="0.8"  />
            </TextBlock.Background>
            <TextBlock.Foreground>
                <SolidColorBrush Color="White"/>
            </TextBlock.Foreground>
</TextBlock>
```

Использование SolidColorBrush в коде:

```
button1.Background = new SolidColorBrush(Colors.Blue);
//или так - это цвет #cfffff
button1.Background = new SolidColorBrush(Color.FromRgb(207, 255, 255));
```

LinearGradientBrush

Эта кисть создает плавный переход от одного цвета к другому. Для указания цвета и точек, от которых начинается переход, используется объект **GradientStop**. Его свойство **Color** указывает на цвет, а свойство **Offset**- на точку, с которой начинается переход.

```
<Button Content="LinearGradientBrush" Canvas.Top="20" Canvas.Right="20" Width="110" Height="30">
                <Button.Background>
                    <LinearGradientBrush StartPoint="0.5,1" EndPoint="0.5,0">
                        <GradientStop Color="Black" Offset="0" />
                        <GradientStop Color="White" Offset="1" />
                    </LinearGradientBrush>
                </Button.Background>
                <Button.Foreground>
                    <LinearGradientBrush>
                        <GradientStop Color="Blue" Offset="1" />
                        <GradientStop Color="Red" Offset="0.5" />
                        <GradientStop Color="Yellow" Offset="0" />
                    </LinearGradientBrush>
                </Button.Foreground>
</Button>
```

С помощью свойств StartPoint и EndPoint можно определить направление градиента, сделать горизонтальный градиент или градиент под углом.

RadialGradientBrush

Эта кисть заполняет элемент радиальным градиентом. Объект RadialGradientBrush также имеет коллекцию объектов **GradientStop**, задающих цвет и смещение. Кроме того, он позволяет задавать центр градиента с помощью свойства |**GradientOrigin**

```
<Button Content="RadialGradientBrush" Canvas.Bottom="20" Canvas.Left="20" Width="110" Height="30
                <Button.Background>
                    <RadialGradientBrush GradientOrigin="0.4,0.1">
                        <GradientStop Color="Black" Offset="1" />
                        <GradientStop Color="White" Offset="0" />
                    </RadialGradientBrush>
                </Button.Background>
                <Button.Foreground>
                    <RadialGradientBrush Center="0.4,0.4"  SpreadMethod="Reflect">
                        <GradientStop Color="Black" Offset="1" />
                        <GradientStop Color="Yellow" Offset="0.2" />
                    </RadialGradientBrush>
                </Button.Foreground>
</Button>
```

Также RadialGradientBrush позволяет ограничить область градиента с помощью свойств **RadiusX** и **RadiusY**

```
<Ellipse Canvas.Top="90" Canvas.Left="120" Width="60" Height="60">
                <Ellipse.Fill>
                    <RadialGradientBrush RadiusX="0.6" RadiusY="0.8" GradientOrigin="0.3,0.3">
                        <GradientStop Color="Red" Offset="1" />
                        <GradientStop Color="White" Offset="0" />
                    </RadialGradientBrush>
                </Ellipse.Fill>
</Ellipse>
```

ImageBrush

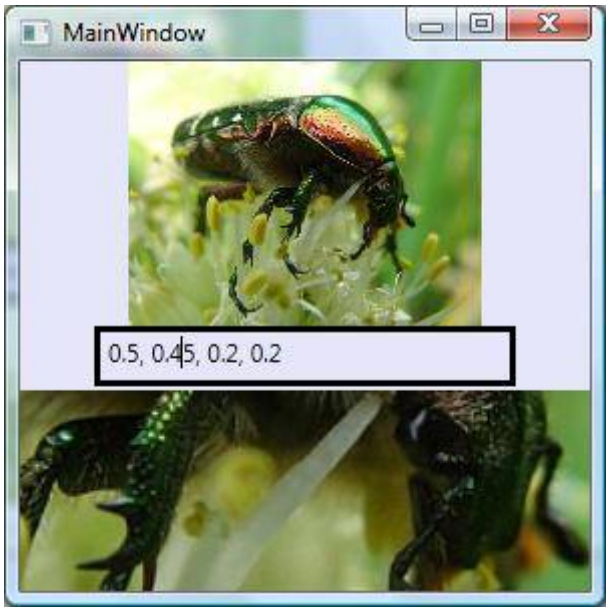
Эта кисть использует изображение в качестве фона. Источник устанавливается свойством **ImageSource**. Свойство **Stretch** задает способ заполнения элемента изображением - если оно равно **Fill** (по умолчанию), то изображение заполняет весь элемент, растягиваясь, если это нужно. Если Stretch="Uniform", то изображение масштабируется пропорционально размеру элемента и по краям могут образоваться пустые места, не заполненные изображением.

```
<ImageBrush ImageSource="D:\Images\image.jpg"
                            Stretch="Uniform" />
```

Пример использования кистей:

Среди прочих свойств ImageBrush следует отметить свойство **Viewbox**. Оно применяется для выреза какой-то части изображения. Его первый параметр служит для установки x-координаты изображения, а второй параметр - y-координаты. Они находятся в пределах от нуля до единицы, и чтобы получить реальные координаты изображения, надо умножить первый параметр на ширину, а второй параметр - на высоту изображения. Третий и четвертый параметр указывают соответственно на ширину и высоту вырезаемого изображения. Так ниже в примере, начальная точка выреза изображения имеет координаты: 0.5*ширина_изображения, 0.45 * высота_изображения. Вырезается 30% от оставшейся ширины и 20% от оставшейся длины:

```
<Grid x:Name="grid1" Background="Lavender">
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <Image Source="D:\Images\Image.jpg" Grid.Row="0"
        HorizontalAlignment="Center" />
       <Canvas Grid.Row="1" >
            <Canvas.Background>
                <ImageBrush ImageSource="D:\Images\Image.jpg"
                            Stretch="Uniform" Viewbox="0.5,0.45,0.3,0.2" />
            </Canvas.Background>
        </Canvas>
    </Grid>
```



ImageBrush также позволяет нам многократно отобразить изображение на элементе и проделывать с ним некоторые преобразования. Для этого класс ImageBrush имеет свойство **Viewport**. Оно похоже на Viewbox, также задает четрые параметра, только они указывают на координаты прямоугольника Viewbox на элементе управления. Первый и второй параметр указывают на начальную координату этого прямоугольника, а третий и четвертый - на конечную точку. Реальные координаты получаются путем умножения параметров на длину и ширину элемента.

Кроме того, свойство **TileMode** позволяет задать режим заполнения элемента изображением. Оно имеет четыре варианта:

- **Tile** - изображение многократно повторяется на элементе, пока не заполнит все пространство.
- **FlipX** - изображение повторяется по оси X, и каждый второй столбец является зеркальным отображением предыдущего
- **FlipY** - изображение повторяется по оси Y, и каждая вторая строка является зеркальным отображением предыдущей
- **FlipXY**- каждое изображение зеркально отображается как по оси X, так и по оси Y.
- **None** - создается единичное изображение (по умолчанию)

Пример

```
<Grid.Background>
            <ImageBrush ImageSource="D:\Images\Image.jpg"
                        Viewport="0,0,0.25,0.25" TileMode="FlipXY" />
        </Grid.Background>
```

DrawingBrush

DrawingBrush - более сложная кисть. Ее рисунок может содержать как отдельные изображения, так и сложные рисунки с применением различных геметрических форм.

Предыдущий пример можно было сделать с помощью DrawingBrush:

```
<DrawingBrush TileMode="FlipXY" Viewport="0,0,0.25,0.25">
            <DrawingBrush.Drawing>
                 <ImageDrawing  ImageSource="D:\Images\Image.jpg" Rect="0,0,1,1" />
            </DrawingBrush.Drawing>
</DrawingBrush>
```

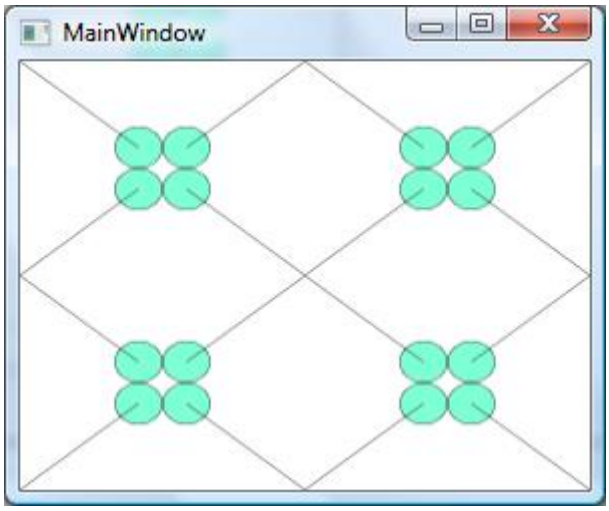
DrawingBrush использует те же свойства, что и ImageBrush - Viewport, Viewbox.

DrawingBrush имеет несколько вариантов рисунка:

- **ImageDrawing** - заполнителем кисти является изображение.
- **GeometryDrawing** - кисть формируется на основе рисунка, составленного каким-нибудь геометрическим примитивом (прямоугольником, линией, эллипсом)
- **VideoDrawing** - кисть формируется на основе видеоресурса.
- **GlyphRunDrawing**

При необходимости сочетания нескольких вариантов, используется свойство DrawingGroup класса Drawing.

```
<DrawingBrush TileMode="FlipXY" Viewport="0,0,0.25,0.25">
         <DrawingBrush.Drawing>
               <DrawingGroup>
                     <GeometryDrawing Brush="Aquamarine">
                         <GeometryDrawing.Pen>
                             <Pen Brush="Black" />
                         </GeometryDrawing.Pen>
                         <GeometryDrawing.Geometry>
                             <EllipseGeometry RadiusX="30" RadiusY="30" Center="150,125" />
                         </GeometryDrawing.Geometry>
                     </GeometryDrawing>
                     <GeometryDrawing Brush="Aquamarine">
                         <GeometryDrawing.Pen>
                             <Pen Brush="Black" />
                         </GeometryDrawing.Pen>
                         <GeometryDrawing.Geometry>
                             <LineGeometry EndPoint="150,125" />
                         </GeometryDrawing.Geometry>
                     </GeometryDrawing>
                </DrawingGroup>
         </DrawingBrush.Drawing>
</DrawingBrush>
```



VisualBrush

Эта кисть при помощи свойства **Visual** создает привязку к определенному элементу, копируя весь его фон или его часть.

VisualBrush, как и кисти DrawingBrush и ImageBrush, обладает свойствами Viewport, Viewbox и TileMode, позволяющие проводить все те же преобразования, что были рассмотрены для этих кистей:

```
<Grid x:Name="grid1" Background="Lavender">
        <Button Name="button1" Content="VisualBrush" Background="Black" FontWeight="Black"
              Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Top"
                Width="100" Height="30"/>
        <TextBlock HorizontalAlignment="Center" VerticalAlignment="Center"
                   Width="120" Height="35">
            <TextBlock.Background>
                <VisualBrush Visual="{Binding ElementName=button1}" />
            </TextBlock.Background>
        </TextBlock>
        <TextBlock HorizontalAlignment="Center" VerticalAlignment="Bottom"
                   Width="140" Height="50">
            <TextBlock.Background>
                <VisualBrush Visual="{Binding ElementName=button1}" Viewbox="0.1,0.1,0.3,0.7" />
            </TextBlock.Background>
        </TextBlock>
    </Grid>
```