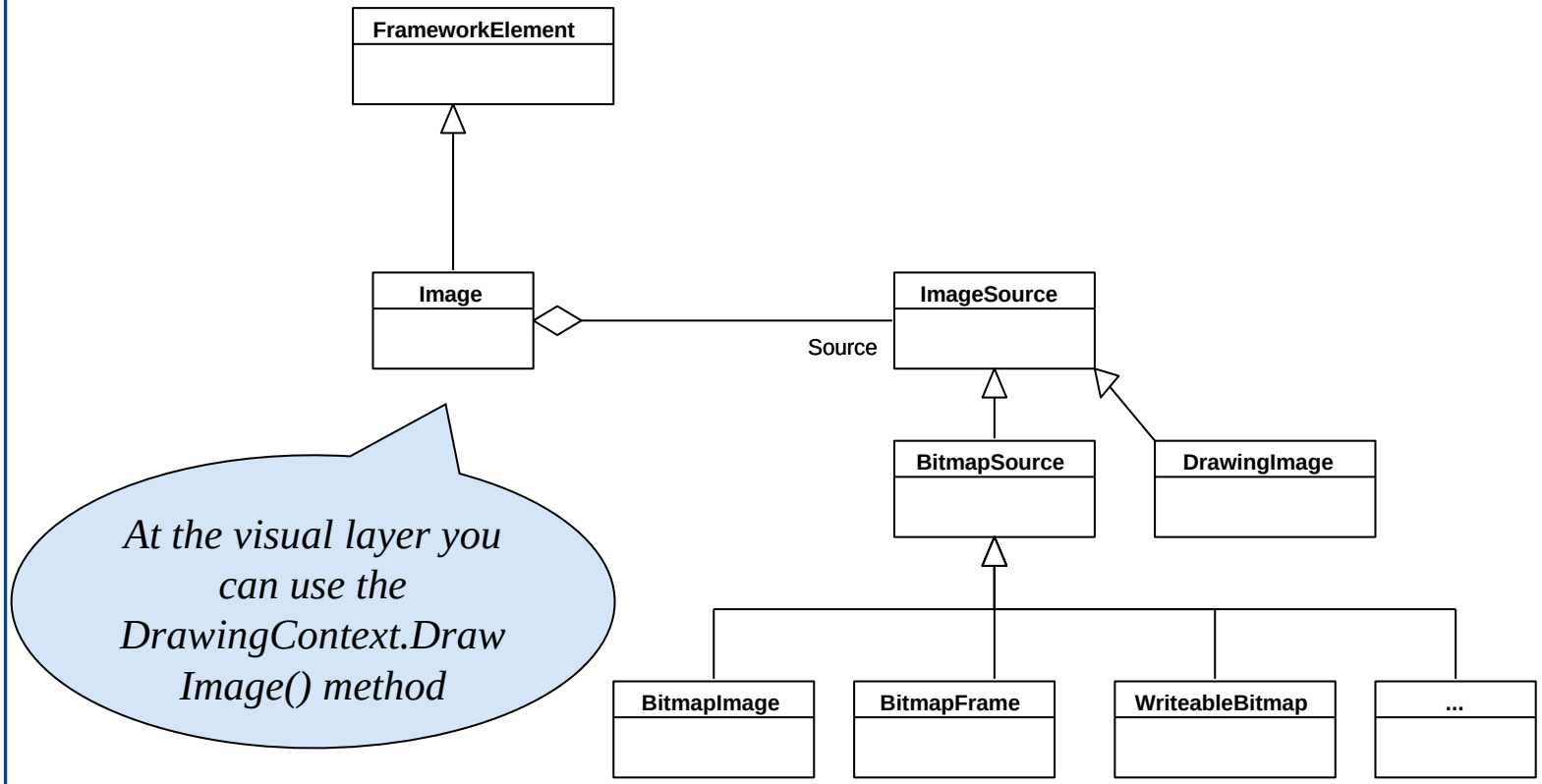
Images

Agenda

- Overview
- Image Basics

Image

- The Image class in WPF simply displays an image
- It derives from FrameworkElement (as Shape does) so you can place it in the visual tree
- It can resize the image.
  - The exact behavior depends on the layout panel and the value of the stretch property

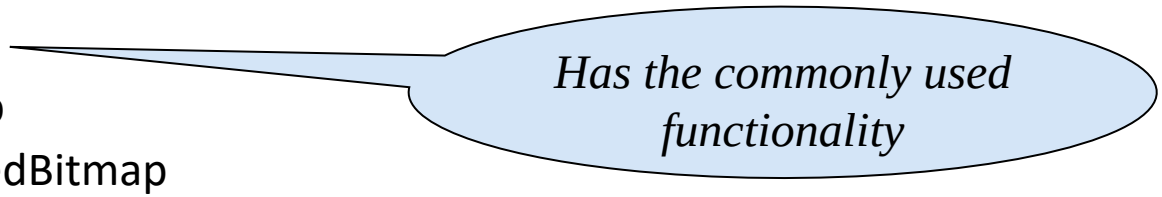


ImageSource

- ImageSource is an abstract class used to represent an image

2 classes derive from ImageSource:

- DrawingImage
  - Wraps a resolution-independent drawing object so it can be used as an image (vector graphics)
- BitmapSource
  - Is an abstract base class for various bitmap sources
    - BitmapFrame
    - **BitmapImage**
    - CachedBitmap
    - ColorConvertedBitmap
    - …
    - TransformedBitmap
    - WriteableBitmap



BitmapSource

- For bitmap decoding scenarios, **BitmapSource** uses automatic codec discovery, based on the installed codecs on the user's system
- Standard supported codecs:
  - BMP
  - GIF
  - ICO
  - JPEG
  - PNG
  - TIFF
  - HD Photo Specification 1.0 (Windows Media Photo)
- Maximum image size is four giga pixels and the minimum image size is 1x1.

Image Basics

Showing an Image

- The Image class is really a viewer for an ImageSource object
- BitmapImage is an ImageSource object used to load common raster image file formats (jpg, gif, png …)

```
<!– showing an image in XAML -->
<Image Source="Sunset.jpg">
```

```
// showing an image in C#
Image img = new Image();
img.Source = new BitmapImage(new URI("Sunset.jpg",
                                     UriKind.Relative));
```

Stretch

- Images have an natural size associated with them
  - (and optionally some meta data)
- When the Image Control displays an image it has several possible solutions to fit the image to the available space
  - You specify this with the stretch property

Source Image

Strecth None

Strecth Fill

Strecth Uniform

Strecth UniformToFill

Image Properties

- The Image control has several properties you can use to adjust the display of an image:
  - Stretch
  - RenderTransform
  - LayoutTransform
  - Effect
  - …
- But many types of image transformations is done by use of an ImageSource pipeline

ImageSource Pipeline

- ImageSource descents may be connected to build a pipeline of commands to be executed on an image as it is decoded

```
BitmapFrame frame = BitmapFrame.Create(new URI("Water
                                                lilies.jpg"));
CroppedBitmap crop = new CroppedBitmap();
crop.BeginInit();
crop.Source = frame;
crop.SourceRect = new Int32Rect(100, 150, 400, 250);
crop.EndInit();
FormatConvertedBitmap color = new FormatConvertedBitmap();
color.BeginInit();
color.Source = crop;
color.DestinationFormat = PixelFormats.BlackWhite;
color.EndInit();
img.Source = color;
```

When The Build-In Manipulation Is Not Enough

- If you want to do serius image enhangement / manipulation you have two options
  - Extract the raw image data and do some math
  - Use (create) a PixelShader (code that runs on the GPU)

THE WRITEABLEBITMAP CLASS

The WriteableBitmap Class

- The WriteableBitmap is wellsuited to applications that need to manipulate individual pixels like
  - a fractal generator
  - a sound analyzer
  - a visualization tool for scientific data
  - an application that processes raw image data from an external hardware device

How to Generate a WriteableBitmap

- To generate a bitmap you must supply a few key pieces of information:
  - its width and height in pixels
  - its DPI resolution in both dimensions
  - the image format

```
WriteableBitmap wb = new WriteableBitmap(width, height, dpiX,
                                         dpiY, PixelFormats.Bgra32, null);
```

- Define the update square (which is as big as the entire image)

```
Int32Rect rect = new Int32Rect(0, 0, width, height);
```

- Create array for pixels

```
byte[] pixels = new byte[width * height *
                         wb.Format.BitsPerPixel / 8];
```

How to Generate a WriteableBitmap - 2

- Calculate stride

```
int stride = (wb.PixelWidth * wb.Format.BitsPerPixel) / 8;
```

- Write the pixels

```
wb.WritePixels(rect, pixels, stride, 0);
```

- Show the bitmap in an Image element

```
img.Source = wb;
```

How to Create a new BitmapSource

```
// Define parameters used to create the BitmapSource.
PixelFormat pf = PixelFormats.Bgr32;
int width = 200;
int height = 200;
int rawStride = (width * pf.BitsPerPixel + 7) / 8;
byte[] rawImage = new byte[rawStride * height];

// Initialize the image with data.
Random value = new Random();
value.NextBytes(rawImage);

// Create a BitmapSource.
BitmapSource bitmap = BitmapSource.Create(width, height, 96, 96, pf, null,
                                          rawImage, rawStride);

// Create an image element;
Image myImage = new Image();
myImage.Width = 200;
// Set image source.
myImage.Source = bitmap;
```

EFFECTS

Media.Effects Namespace

- WPF provides visual effects that you can apply to any element
- The Effect-derived classes:
  - **`BlurEffect`**
    - Blurs the content in your element
  - **`DropShadowEffect`**
    - Adds a rectangular drop shadow behind your element
  - **`ShaderEffect`**
    - Applies a pixel shader, which is a ready-made effect that's defined in High Level Shading Language (HLSL) and already compiled

```
<TextBlock FontSize="20" Margin="5"
           Text="BlurEffect with Radius = 5">
    <TextBlock.Effect>
        <BlurEffect Radius="5"/>
    </TextBlock.Effect>
</TextBlock>
```

PixelShader Useage

- To use a pixelshader is very simple:

```
<Image Name="image1" Stretch="Uniform" Source="bouquet.png">
    <Image.Effect>
        <shaders:DesaturateEffect x:Name="SaturationEffect“
                                  Saturation="1.0">
        </shaders:DesaturateEffect>
    </Image.Effect>
</Image>
```

You can find many pixelshaders on the Internet

PixelShader Creation

- Write the Shader in the HLSL language:

```
float4 DesaturatePS(float2 uv : TEXCOORD0) : COLOR {
float3  LuminanceWeights = float3(0.299,0.587,0.114);
    float4srcPixel = tex2D(implicitSampler, uv);
    floatluminance = dot(srcPixel,LuminanceWeights);
    float4dstPixel = lerp(luminance,srcPixel,Saturation);
    //retain the incoming alpha
    dstPixel.a = srcPixel.a;
    return dstPixel;
}
```

- Write a C# wrapper:

```
public class DesaturateEffect : ShaderEffect {

        public DesaturateEffect()
        {
            PixelShader = _pixelShader;

            UpdateShaderValue(TextureProperty);
            UpdateShaderValue(SaturationProperty);
        } . . .
```

References And Links

- **Imaging Overview**
  http://msdn.microsoft.com/en-us/library/ms748873.aspx

- ImageMagic - WPF Image Sharpening
  http://www.codeproject.com/Articles/44115/ImageMagic-WPF-Image-Sharpening

- ImageMagick® is a software suite to create, edit, compose, or convert bitmap images.
  http://www.imagemagick.org/script/index.php
  ImageMagick.NET : http://imagemagick.codeplex.com/

- Image processing application
  http://www.paint.net/