NOKIA Developer

Image control using 9-patch rescaling for Windows Phone

Note: This is an entry in the Nokia Imaging and Big UI Wiki Competition 2013Q4.

This article explains about an alternative implementation of 9 patch images for windows phone.

Introduction



With the roll out of Windows phone update 3(GDR3), 1080p resolution have joined Lumia family. The two newly launched devices Lumia 1520 and Lumia 1320 have 6 inch displays, much bigger than the conventional 4 inch or smaller screens.

Before this we just had 4 inch or smaller screen devices. So one background image would suffice all the phones without much distortion. But for these 6 inch big screens, the auto scaling which system performs on the images to adjust with the current screen size can distort the image considerably, making the image look extra stretched, unclear etc. Android/iOS tackles this issue by supporting 9 patch images. But unfortunately windows phone do not have support for it. We will discuss about an alternative approach for windows phone in the following sections.

Why do we need it

Windows phone started with Nokia Lumia 800 with a screen size of 3.7", then we had Nokia Lumia 900 with 4.3" screen followed by Lumia 510 with 4.0", Lumia 920 with 4.5", Lumia 625 with 4.7" screen and recently Lumia 1520 and 1320 with a massive 6.0" screens. As windows phone market grows we can expect even bigger or much smaller screens.

Now the issue with having a lot of screen sizes to support is, keeping your UI consistent on all of them and the image assets that you use in your application play a bigger role. Even though windows phone OS has the capability of scaling the images to fit onto current screen. The re-scaling works for small difference in screen size but that's not always what we want. Imagine system re-scaling an image made for 3.7" screen for a 6" screen, that will most possibly distort the image. So to avoid system re-scaling your images, you have 2 options

- 1. Include image assets for all screen sizes and have mechanism for loading proper assets based on screen size.
- 2. Have some mechanism in place so that even high re-scaling do not distort your images.

In first implementation, with every new screen, you will need to add new assets and recompile your app. But second implementation will work fine without any changes with new screens.

Second implementation is what we will be discussing about in this article.

Background

What is a 9 patch image

The 9-Patch is a PNG image with some coding added that allows the system to determine how the image can be stretched and contorted to meet the specific layout constraints during use. It does this by taking a predefined PNG image, and allowing the user to define a 1-pixel border around the image in locations where stretching can occur.

9-Patch Theory



The 9-Patch gets its name from the fact that the image is broken up into nine defined regions, organized similar to tic-tac-toe. Each region has specific stretch properties:

Corner Regions (A,C,G,I) These regions are fixed and nothing inside them will stretch

Horizontal Sides (D,F) The pixels in these regions will stretch vertically when necessary

Vertical Sides (B,H) The pixels in these regions will stretch horizontally when necessary

Center (E) The pixels in this regions will stretch in both horizontal and vertical directions equally

Alternative for Windows Phone

To simulate a 9 patch image we will use a 3x3 grid(a 9 celled grid). Now we have 9 sections and we have control over how these sections are drawn on screen by specifying width and height properties in ColumnDefinition and RowDefinition respectively. Now we need to divide our background image into 9 parts so that each piece maps to the corresponding cell in grid. You have lot of options to split image into cells, You can use Nokia Imaging SDK to split the image in code using the CropFilter(more info on it in the following sections), or use split extension with GIMP, it gives you freedom of choosing unequal sized cells. Here & are details on installation and usage of the plug-in with GIMP &.

9 patch implementation

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
 <Grid.RowDefinitions>
  <RowDefinition Height="50" />
 <RowDefinition Height="*"/>
  <RowDefinition Height="50" />
 </Grid.RowDefinitions>
 <Grid.ColumnDefinitions>
 <ColumnDefinition Width="50" />
 <ColumnDefinition Width="*" />
 <ColumnDefinition Width="50" />
 </Grid.ColumnDefinitions>
 <Image Name="A" Grid.Row="0" Grid.Column="0" Stretch="Fill" />
 <Image Name="B" Grid.Row="0" Grid.Column="1" Stretch="Fill" />
 <Image Name="C" Grid.Row="0" Grid.Column="2" Stretch="Fill"/>
 <Image Name="D" Grid.Row="1" Grid.Column="0" Stretch="Fill"/>
 <Image Name="E" Grid.Row="1" Grid.Column="1" Stretch="Fill"/>
 <Image Name="F" Grid.Row="1" Grid.Column="2" Stretch="Fill"/>
 <Image Name="G" Grid.Row="2" Grid.Column="0" Stretch="Fill"/>
 <Image Name="H" Grid.Row="2" Grid.Column="1" Stretch="Fill"/>
 <Image Name="I" Grid.Row="2" Grid.Column="2" Stretch="Fill"/>
</Grid>
```

Note: The Width and Height values, currently 50, are not fixed and its entirely upto you to choose a proper value based on

your image. Also Stretch attribute must be set to Fill.

Unscaled	Scaled in X axis	Unscaled
Scaled in Y axis	Scaled in Both Axis	Scaled in Y axis
Unscaled	Scaled in X axis	Unscaled

Above grid provides you with an ideal 9 patch image configuration wherein you have fixed corner regions, vertical sides, horizontal sides and flexible center. This configuration can be used for button backgrounds but it barely suits for page background images. In most cases you need either center, top, left or right part of your background image to be fixed and others to be scaled. Let us see how can we achieve it.

Power in your hands

With the 9 patch image, you don't have any control over behavior of the 9 sections. Those are predefined as explained in section above. But with a 9 celled grid approach you have the whole power of defining behavior of the 9 sections. You may make center of the image to never stretch or only right side of your image stretch in both directions etc. It can achieved by assigning proper attributes to the RowDefinition and ColumnDefinition. We will discuss some of these below.

9 patch variants

Fixed center



The original image shown above has its important content placed in its center, if we use ideal 9 patch configuration, center will stretched in both horizontal and vertical directions making the image look distorted.

This is how image look when center is stretched in horizontal or vertical direction.

Stretch Type	Image
Horizontally	



The images looks pretty distorted. right?

To avoid image distortion, what ideally we would need is fixed center configuration for this particular image. where in center of the image will be fixed while other portion will scale.

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
 <Grid.RowDefinitions>
  <RowDefinition Height="*" />
  <RowDefinition Height="230"/>
  <RowDefinition Height="*" />
 </Grid.RowDefinitions>
 <Grid.ColumnDefinitions>
  <ColumnDefinition Width="*" />
                                                                         Fixed Center properties
  <ColumnDefinition Width="208" />
  <ColumnDefinition Width="*" />
 </Grid.ColumnDefinitions>
 <Image Name="A" Grid.Row="0" Grid.Column="0" Stretch="Fill" />
 <Image Name="B" Grid.Row="0" Grid.Column="1" Stretch="Fill" />
 <Image Name="C" Grid.Row="0" Grid.Column="2" Stretch="Fill"/>
 <Image Name="D" Grid.Row="1" Grid.Column="0" Stretch="Fill"/>
 <Image Name="E" Grid.Row="1" Grid.Column="1" Stretch="Fill"/>
 <Image Name="F" Grid.Row="1" Grid.Column="2" Stretch="Fill"/>
 <Image Name="G" Grid.Row="2" Grid.Column="0" Stretch="Fill"/>
 <Image Name="H" Grid.Row="2" Grid.Column="1" Stretch="Fill"/>
<Image Name="I" Grid.Row="2" Grid.Column="2" Stretch="Fill"/>
</Grid>
```

Note: Set Width and Height of ColumnDefinition and RowDefinition equal to Width and Height of your images center part

respectively.

Now only the edges are stretched in horizontal or vertical direction and center part remains fixed.

Stretch Type Image



Note: Observer the difference between ideal 9 patch and fixed centred configuration stretched images.

Top-Right Stretch

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
<Grid.RowDefinitions>
<RowDefinition Height="*" />
<RowDefinition Height="200"/>
<RowDefinition Height="200" />
</Grid.RowDefinitions>
<Grid.ColumnDefinitions>
<ColumnDefinition Width="200" />
<ColumnDefinition Width="200" />
<ColumnDefinition Width="200" />
<ColumnDefinition Width="200" />
<ColumnDefinition Width="*" />
</Grid.ColumnDefinitions>
</Grid.ColumnDefinitions>
```

Stretch Type

Image type to be applied against



This configuration makes only top and right side of the image to stretch, keeping all other cells fixed. This configuration is usable in cases where bottom left part of your image should be kept fixed as with the image in the table above.

Bottom-Left Stretch

```
_____
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
 <Grid.RowDefinitions>
  <RowDefinition Height="200" />
  <RowDefinition Height="200"/>
  <RowDefinition Height="*" />
 </Grid.RowDefinitions>
 <Grid.ColumnDefinitions>
                                                                                         묘
  <ColumnDefinition Width="*" />
                                                                        Bottom Left Stretch
                                                                        properties
  <ColumnDefinition Width="200" />
  <ColumnDefinition Width="200" />
 </Grid.ColumnDefinitions>
</Grid>
```

Stretch Type	Image type to be applied against
Bottom-Left Stretch	

This configuration makes only top and right side of the image to stretch, keeping all other cells fixed. This configuration is usable in cases where top right part of your image should be kept fixed as with the image in the table above.

On the similar lines we can have Top-Bottom, Left-Right stretches.

Right Only Stretch

	Printed on 2014-03-0
<grid grid.row="1" margin="12,0,12,0" x:name="ContentPanel"></grid>	Unscaled Unscaled
<grid.rowdefinitions></grid.rowdefinitions>	
<rowdefinition height="200"></rowdefinition>	
<rowdefinition height="200"></rowdefinition>	Unscaled Unscaled
<rowdefinition height="200"></rowdefinition>	
	Unscaled Unscaled
<grid.columndefinitions></grid.columndefinitions>	
<columndefinition width="200"></columndefinition>	Right only stretch
<columndefinition width="200"></columndefinition>	properties
<columndefinition width="*"></columndefinition>	

This configuration will make only right side(3rd column) of the image to be stretched, keeping all other cells fixed.

Similarly we can have Left Only, Bottom Only and Top Only configurations for keeping the respective edge stretchable.

Splitting image using Nokia Imaging SDK

Now that we know about the 9-patch and its variants, let us look at how to split the image in code. You can use Nokia Imaging SDK's cropFilter to split the image into 9 parts. Below code will split image into 9 equally sized cells and assigns them to the corresponding 9-patch sections.

```
async void SlicePhoto(Stream photoStream)
{
if (photoStream != null)
 using (var source = new StreamImageSource(photoStream))
  {
  var inf = await source.GetInfoAsync();
  Windows.Foundation.Rect rect_E;
   //get the middle rect
   rect_E = new Windows.Foundation.Rect(inf.ImageSize.Width / 3, inf.ImageSize.Height /
3, inf.ImageSize.Width / 3, inf.ImageSize.Height / 3);
   {
   //Get rect for A cell
   var rect_A = new Windows.Foundation.Rect(0, 0, rect_E.Left, rect_E.Top);
   //extract bitmap for A cell from the image and assign it to A cell of the grid
   A.Source = await extract(source, rect_A);
   }
   {
   //Get rect for B cell
   var RectB = new Windows.Foundation.Rect(rect_E.Left, 0, rect_E.Width, rect_E.Top);
   //extract bitmap for B cell from the image and assign it to B cell of the grid
   B.Source = await extract(source, RectB);
   }
   {
    //Get rect for C cell
   var rect_C = new Windows.Foundation.Rect(rect_E.Right, 0, inf.ImageSize.Width -
```

Page 7 of 10

```
Page 8 of 10
rect_E.Right, rect_E.Top);
                                                                                Printed on 2014-03-09
    //extract bitmap for C cell from the image and assign it to C cell of the grid
   C.Source = await extract(source, rect_C);
   }
   {
    //Get rect for D cell
    var rect_D = new Windows.Foundation.Rect(0, rect_E.Top, rect_E.Left, rect_E.Height);
   //extract bitmap for D cell from the image and assign it to D cell of the grid
   D.Source = await extract(source, rect_D);
   }
   {
    //extract bitmap for E cell from the image and assign it to E cell of the grid
   E.Source = await extract(source, rect_E);
   }
   {
    //Get rect for F cell
    var rect_F = new Windows.Foundation.Rect(rect_E.Right, rect_E.Top,
inf.ImageSize.Width - rect_E.Right, rect_E.Height);
    //extract bitmap for F cell from the image and assign it to F cell of the grid
   F.Source = await extract(source, rect_F);
  }
    //Get rect for G cell
    var rect_G = new Windows.Foundation.Rect(0, rect_E.Bottom, rect_E.Left,
inf.ImageSize.Height - rect_E.Bottom);
    //extract bitmap for G cell from the image and assign it to G cell of the grid
   G.Source = await extract(source, rect_G);
  }
   {
    //Get rect for H cell
    var rect_H = new Windows.Foundation.Rect(rect_E.Left, rect_E.Bottom, rect_E.Width,
inf.ImageSize.Height - rect_E.Bottom);
    //extract bitmap for H cell from the image and assign it to H cell of the grid
   H.Source = await extract(source, rect_H);
  }
   Ł
    //Get rect for I cell
    var rect_I = new Windows.Foundation.Rect(rect_E.Right, rect_E.Bottom,
inf.ImageSize.Width - rect_E.Right, inf.ImageSize.Height - rect_E.Bottom);
    //extract bitmap for I cell from the image and assign it to I cell of the grid
   I.Source = await extract(source, rect_I);
  }
  }
}
}
```

```
//method to extract a rectangular portion from the image
async Task<WriteableBitmap> extract(IImageProvider source, Windows.Foundation.Rect rect)
{
  var bmp = new WriteableBitmap((int)rect.Width, (int)rect.Height);
  using (var filter = new FilterEffect(source))
  using (var renderer = new WriteableBitmapRenderer(filter, bmp))
  {
    filter.Filters = new IFilter[] { new CropFilter(rect) };
    return await renderer.RenderAsync();
  }
}
```

Reusable UserControl

You can find a reusable UserControl named *FixedCenter9CellGrid* for FixedCentre variant in the attached Source Code. You need to provide it with following data

- FixedX : X coordinate for center cell
- FixedY : Y coordinate for center cell
- FixedWidth : Width of the center cell
- FixedHeight : Height of the center cell
- ImageUri : String Uri of the image

Depending on the values you provide the image is divided into 9 parts and each part is assigned to the corresponding 9-patch section. Center part(E part) is kept fixed and others scale to fit the screen size. If you provide only the ImageUri, the default behaviour is to divide the image into 9 equal sized cells.

By tweaking in little bit in the code, you shall be able to create reusable UserControl for other 9-patch variants too.

Note: The FixedX, FixedY, FixedWidth, FixedHeight and ImageUri are all DependencyProperty, you can target them for data

binding.

Note: Credits to Yan for the image splitting Code

Summary

In this article, we learnt to

- Implement 9 patch styled background images for windows phone using Grid control.
- Design variants of 9 patch, based on the nature of background image.
- Slice an image into 9 parts using Nokia Imaging SDK.
- Create a reusable UserControl for Fixed Center 9-patch variant.

References

- http://adilsoomro.blogspot.in/2012/11/android-how-to-use-9-patch-png.html IP
- http://wiresareobsolete.com/wordpress/2010/06/9-patches/ IP
- http://www.arakne.es/en/dessign/gimp-websplit-split-the-image-using-paths/

Page 10 of 10 Printed on 2014-03-09