Custom Color Picker for Windows Phone

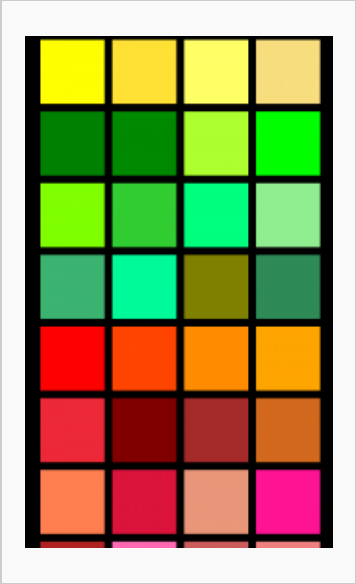
This article explains how to create a custom "discrete" colour picker page for Windows Phone.

16 Jun 2013

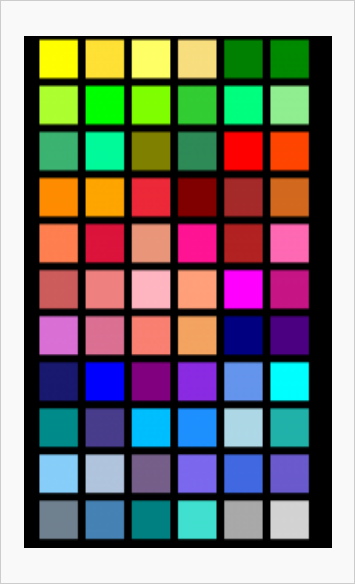
Introduction

Working with Windows Phone applications sometimes you need to have a colour picker. Sometimes a picker like that provided in the Coding4Fun Toolkit, where you can pick from any colour in the spectrum, is appropriate. In other cases it makes more sense to offer a user a discrete selection of colours.

This article explains how to create a colour picker that offers a discrete set of colours, as shown below:



Color picker page with large rectangle



Color picker page with small rectangle

Colour picker page

First of all, we need to define XAML to get small colour rectangle in rows and columns. We will use regular `ListBox` with small addition of `ItemPanelTemplate` defined as `WrapPanel`. To use `WrapPanel` you need to use Windows Phone Toolkit or `RadControls`. In this example we use controls with WP toolkit.

Note: The following code is tested on Windows Phone 8 but almost the same code would also work on Windows Phone 7.

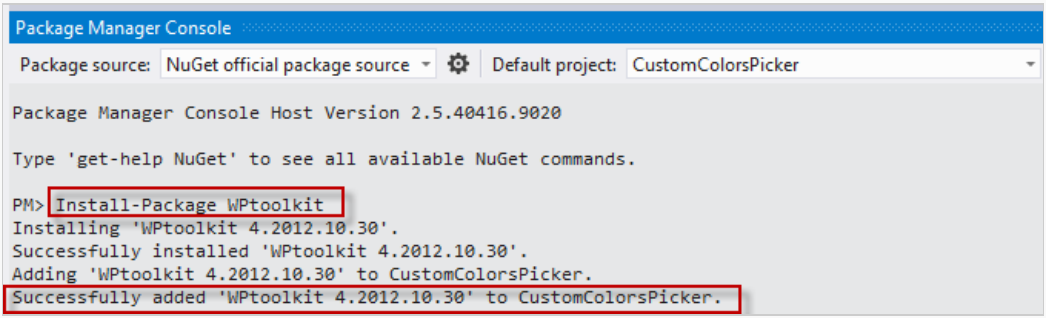
XAML code for colour picker page

```
<phone:PhoneApplicationPage
    x:Class="CustomColorsPicker.ColorPickerPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    mc:Ignorable="d"
    shell:SystemTray.IsVisible="False"
    xmlns:converters="clr-namespace:CustomColorsPicker.Converters"
```


(C) Copyright Nokia 2014. All rights reserved.

```
    xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    toolkit:TiltEffect.IsTiltEnabled="True">

    <phone:PhoneApplicationPage.Resources>
        <converters:ColorToBrushConverter x:Key="ColorToBrushConverter"/>
    </phone:PhoneApplicationPage.Resources>

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">

            <ListBox Name="listBox" SelectionChanged="lstColor_SelectionChanged">
                <ListBox.ItemsPanel>
                    <ItemsPanelTemplate>
                        <toolkit:WrapPanel/>
                    </ItemsPanelTemplate>
                </ListBox.ItemsPanel>

                <ListBox.ItemTemplate>
                    <DataTemplate>
                        <StackPanel x:Name="item" Orientation="Horizontal" Margin="12, 6
0, 6">
                            <Rectangle Fill="{Binding Color, Converter={StaticResource
ColorToBrushConverter}}"
                                Width="100" Height="100" />
                        </StackPanel>
                    </DataTemplate>
                </ListBox.ItemTemplate>
            </ListBox>

        </Grid>
    </Grid>

</phone:PhoneApplicationPage>
```

To use toolkit you need to install it first because it is not part of the SDK for Windows Phone 8. The best way to install it is to use NuGet (tool that help you to install packages into your Visual Studio project) from Visual Studio: **Tools/Library Package Manager/Package Manager Console**, and the use next command: Install-Package WPtoolkit



NuGet Installed

If you don't have NuGet I strongly suggest you to install it in you Visual Studio environment (**Tools/Extensions and Updates NuGet Package Manager**).

(C) Copyright Nokia 2014. All rights reserved.

When you install WP toolkit, you need to include it in the XAML. I also add `TiltEffect` into the page to make it obvious what colour is selected (you'll see some small animation when you touch one of the rectangles filled with colour). Finally, in order to colour the rectangle, we used `ColorToBrushConverter` to fill the rectangle with the selected colour. Fill controls need `SolidColorBrush` type of object and we use converter to convert from `Color` type to `SolidColorBrush` type.

The size of the rectangle can change using `Width` and `Height` property of the rectangle. For small size we use 65 px and for large size we use 100 px. You can use size depending how many colours you want on your colour picker page.

When we have definition of our colour picker page, we need to fill these rectangle with colours. So, the next step is code behind explanation of our `ColorPickerPage`.

Code behind for color picker page

First we define `colorNames` and `uintColors` for definition of our custom colours:

```
static string[] colorNames =
{
 "Yellow","BananaYellow","LaserLemon","Jasmine","Green","Emerald",
 "GreenYellow","Lime","Chartreuse","LimeGreen","SpringGreen","LightGreen",
 "MediumSeaGreen","MediumSpringGreen","Olive","SeaGreen","Red","OrangeRed",
 "DarkOrange","Orange","ImperialRed","Maroon","Brown","Chocolate",
 "Coral","Crimson","DarkSalmon","DeepPink","Firebrick","HotPink",
 "IndianRed","LightCoral","LightPink","LightSalmon","Magenta","MediumVioletRed",
 "Orchid","PaleVioletRed","Salmon","SandyBrown","Navy","Indigo",
 "MidnightBlue","Blue","Purple","BlueViolet","CornflowerBlue","Cyan",
 "DarkCyan","DarkSlateBlue","DeepSkyBlue","DodgerBlue","LightBlue","LightSeaGreen",
 "LightSkyBlue","LightSteelBlue","Mauve","MediumSlateBlue","RoyalBlue","SlateBlue",
 "SlateGray","SteelBlue","Teal","Turquoise","DarkGrey","LightGray"
};

static uint[] uintColors =
{
 0xFFFFFF00,0xFFFFE135,0xFFFFFF66,0xFFF8DE7E,0xFF008000,0xFF008A00,
 0xFFADFF2F,0xFF00FF00,0xFF7FFF00,0xFF32CD32,0xFF00FF7F,0xFF90EE90,
 0xFF3CB371,0xFF00FA9A,0xFF808000,0xFF2E8B57,0xFFFF0000,0xFFFF4500,
 0xFFFF8C00,0xFFFFA500,0xFFED2939,0xFF800000,0xFFA52A2A,0xFFD2691E,
 0xFFFF7F50,0xFFDC143C,0xFFE9967A,0xFFFF1493,0xFFB22222,0xFFFF69B4,
 0xFFCD5C5C,0xFFF08080,0xFFFFB6C1,0xFFFFA07A,0xFFFF00FF,0xFFC71585,
 0xFFDA70D6,0xFFDB7093,0xFFFA8072,0xFFF4A460,0xFF000080,0xFF4B0082,
 0xFF191970,0xFF0000FF,0xFF800080,0xFF8A2BE2,0xFF6495ED,0xFF00FFFF,
 0xFF008B8B,0xFF483D8B,0xFF00BFFF,0xFF1E90FF,0xFFADD8E6,0xFF20B2AA,
 0xFF87CEFA,0xFFB0C4DE,0xFF76608A,0xFF7B68EE,0xFF4169E1,0xFF6A5ACD,
 0xFF708090,0xFF4682B4,0xFF008080,0xFF40E0D0,0xFFA9A9A9,0xFFD3D3D3
};
```

You can find list of colours in alphabetical order on Wiki page List of colors: A-M and List of colors: N-Z. For `uintColors` variable use Hex(RGB) value from Wiki page but add FF value after 0x so the format for `uintColors` is for example for yellow color: 0xFFFFFF00.

When `ColorPickerPages` is loaded we need to use all the colours from our static variable and use it as `ItemSource` for `ListBox` control:

```
private void ColorPickerPage_Loaded(object sender, RoutedEventArgs e)
{
    List<ColorItem> item = new List<ColorItem>();
    for (int i = 0; i < 66; i++)
    {
```


(C) Copyright Nokia 2014. All rights reserved.

```
        item.Add(new ColorItem() { Text = colorNames[i], Color =
ConvertColor(uintColors[i]) });
    };

    listBox.ItemsSource = item; //Fill ItemSource with all colors
}

private Color ConvertColor(uint uintCol)
{
    byte A = (byte)((uintCol & 0xFF000000) >> 24);
    byte R = (byte)((uintCol & 0x00FF0000) >> 16);
    byte G = (byte)((uintCol & 0x0000FF00) >> 8);
    byte B = (byte)((uintCol & 0x000000FF) >> 0);

    return Color.FromArgb(A, R, G, B); ;
}
```

As our definition of our colours is in Hexadecimal format we use `ConvertColor()` function to split hex value into A,R,G,B component and use `Color.FromArgb` to get `Color` type that we could use in our app.

To store color item (text and color value) we use `ColorItem` class with two properties:

```
public class ColorItem
{
    public string Text { get; set; }
    public Color Color { get; set; }
}
```

And finally, when one of the color is selected `SelectionChanged` is triggered:

```
private void lstColor_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (e.AddedItems.Count > 0)
    {
        (Application.Current as App).CurrentColorItem = ((ColorItem)e.AddedItems[0]);
        this.NavigationService.GoBack();
    }
}
```

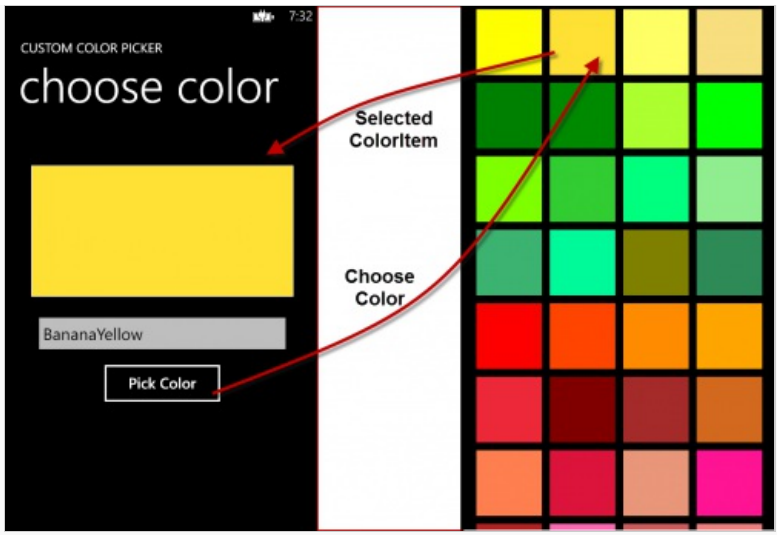
We use global `CurrentColorItem` property to store selected `ColorItem`. `CurrentColorItem` is defined in `App.xaml.cs` class as:

```
public ColorItem CurrentColorItem { set; get; }
```

How to call ColorPickerPage

When we have `ColorPickerPage` defined, we need to call it and receive colour and name of the colour. For that purpose we use **MainPage.xaml**. Let say, just for presentation purpose, that we want to fill a rectangle on the first page of our app with some custom colours and the name of the same colour in text box. The main purpose of main page is to show selected colour and name of the colour.

(C) Copyright Nokia 2014. All rights reserved.



Custom Color Picker

Also, we use `IsolatedStorageSettings` to store last used `ColorItem`. When app is start for the first time, we choose some colour to be default (Cyan) and when we are navigating to the main page and if we have value for the global variable `CurrentColorItem` then means that we select some color from `ColorPickerPage`.

MainPage.xaml.cs

```
public partial class MainPage : PhoneApplicationPage
{
    private ColorItem _colorItem;
    IsolatedStorageSettings settings = IsolatedStorageSettings.ApplicationSettings;

    // Constructor
    public MainPage()
    {
        InitializeComponent();
        InitializeSettings();
    }

    private void InitializeSettings()
    {
        if (!settings.Contains("LastColorItem"))
        {
            ColorItem item = new ColorItem();
            item.Color = Colors.Cyan;
            item.Text = "Cyan";
            settings.Add("LastColorItem", item);
        }
    }

    protected override void
OnNavigatedFrom(System.Windows.Navigation.NavigationEventArgs e)
    {
        base.OnNavigatedFrom(e);
        settings["LastColorItem"] = _colorItem;
    }

    protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs
e)
    {
        base.OnNavigatedTo(e);
```


(C) Copyright Nokia 2014. All rights reserved.

```
        //get settings value
        if (settings.Contains("LastColorItem"))
        {
            _colorItem = (ColorItem)settings["LastColorItem"];
        }

        ColorItem myColorItem = (Application.Current as App).CurrentColorItem;
        if (myColorItem != null)
        {
            _colorItem = (ColorItem)myColorItem;
        }

        this.ColorName.Text = _colorItem.Text;
        this.FillRect.Fill = new SolidColorBrush(_colorItem.Color);
    }

    private void PickColorButton_Click(object sender, RoutedEventArgs e)
    {
        NavigationService.Navigate(new Uri("/ColorPickerPage.xaml", UriKind.Relative));
    }
}
```

In `OnNavigatedTo` method we use local variable `_colorItem` which is `ColorItem` type to fill `Rectangle` with colour and `TextBox` with text whether from last used color (`LastColorItem` stored in `IsolatedStorageSettings`) or from ColorPickerPage (using `CurrentColorItem`). When we are navigating from `MainPage` we store last used colour in `IsolatedStorageSettings`.

XAML

In `MainPage` we use three controls in `StackPanel`, `Rectangle` for colour, `TextBox` for name of the colour, `Button` for calling `ColorPickerPage`:

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
    <StackPanel Orientation="Vertical">
        <Rectangle x:Name="FillRect" Width="400" Height="200" Stroke="White"
  Margin="0,50,0,20"/>
        <TextBox x:Name="ColorName" HorizontalContentAlignment="Center" Width="400"/>
        <Button Height="80" Width="200" Content="Pick Color"
Click="PickColorButton_Click"/>
    </StackPanel>
</Grid>
```

That is it!

References

- Custom Color Picker for Windows Phone (Spaso Lazarevic Blog - Original source)

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.