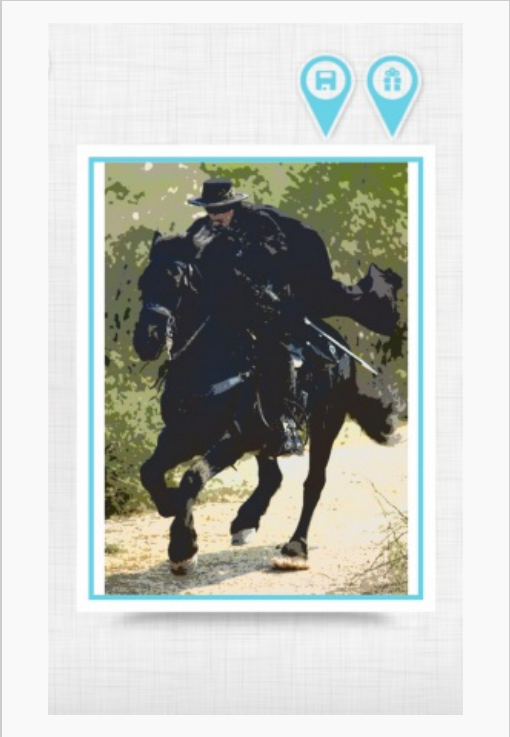
Using Nokia Imaging SDK Create Cartoon Photo

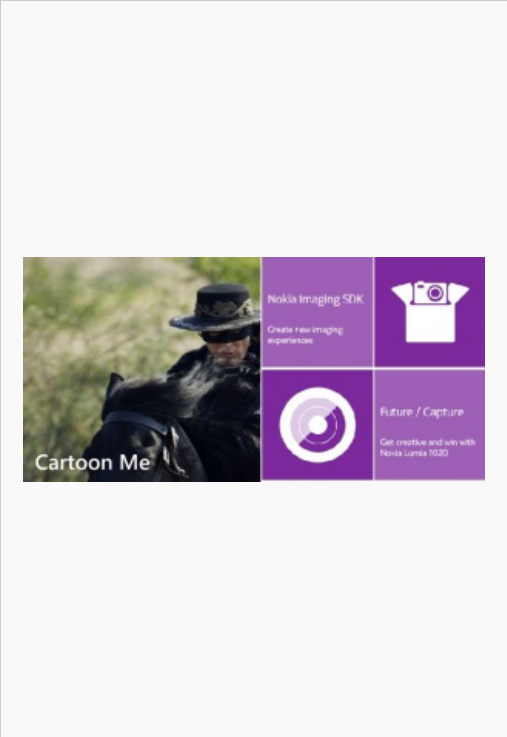
This article explains how to use the Nokia Imaging SDK to create a cartoon image, crop images and splice photos for wide tile image.

Introduction

Applying image effects with the Nokia Imaging SDK is both amazing and efficient. Changing portrait photo into cartoon picture is very funny. The article provides how to use `FilterFactory.CreateCartoonFilter` method to transforms image to typical cartoon graphic style. And the article also describes the process of using `FilterFactory.CreateCropFilter` Method to crop the picture, splicing five photos for wide tile image.



Cartoon Filter Photo



Wide Live Tile

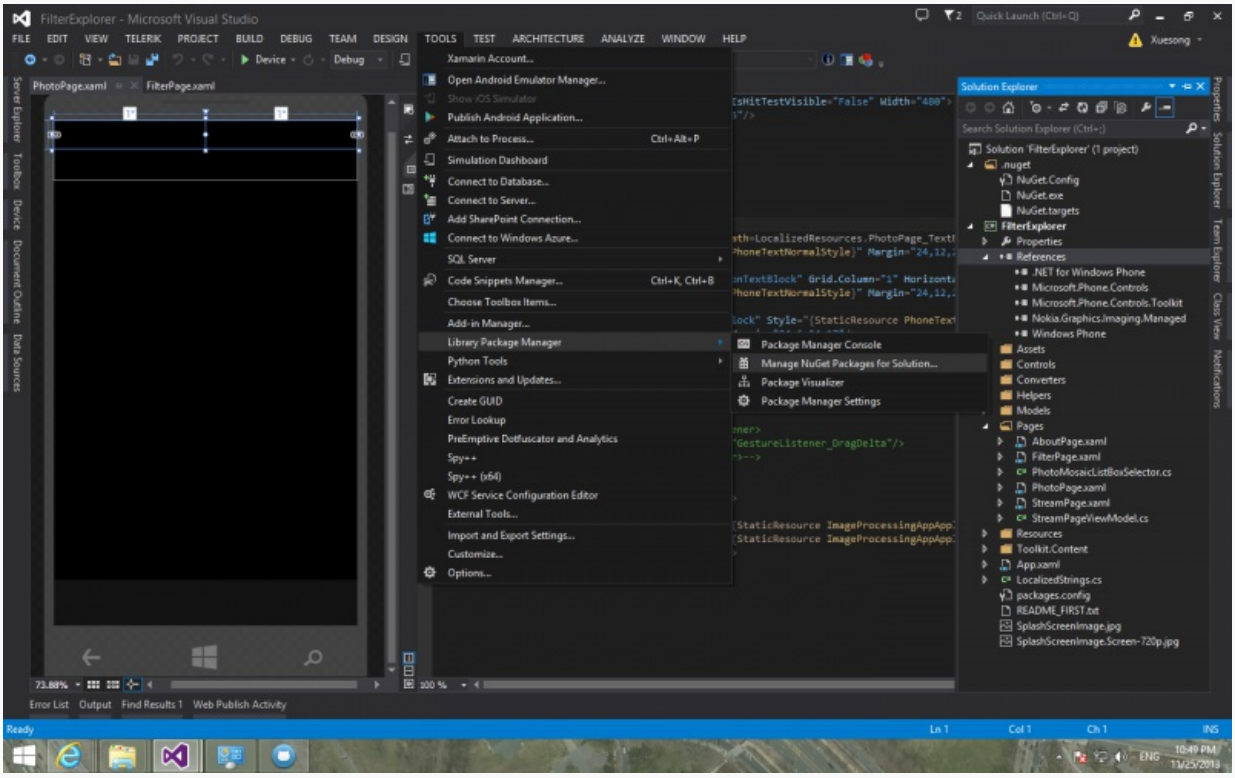
Installing Nokia Imaging SDK

The links below provide a good starting point:

- Core concepts (Nokia Imaging SDK documentation)
- Download and add the libraries to the project (Nokia Imaging SDK documentation)

In Visual Studio, start NuGet package manager: Tools menu > Library Package Manager > Manage NuGet Packages for Solution...

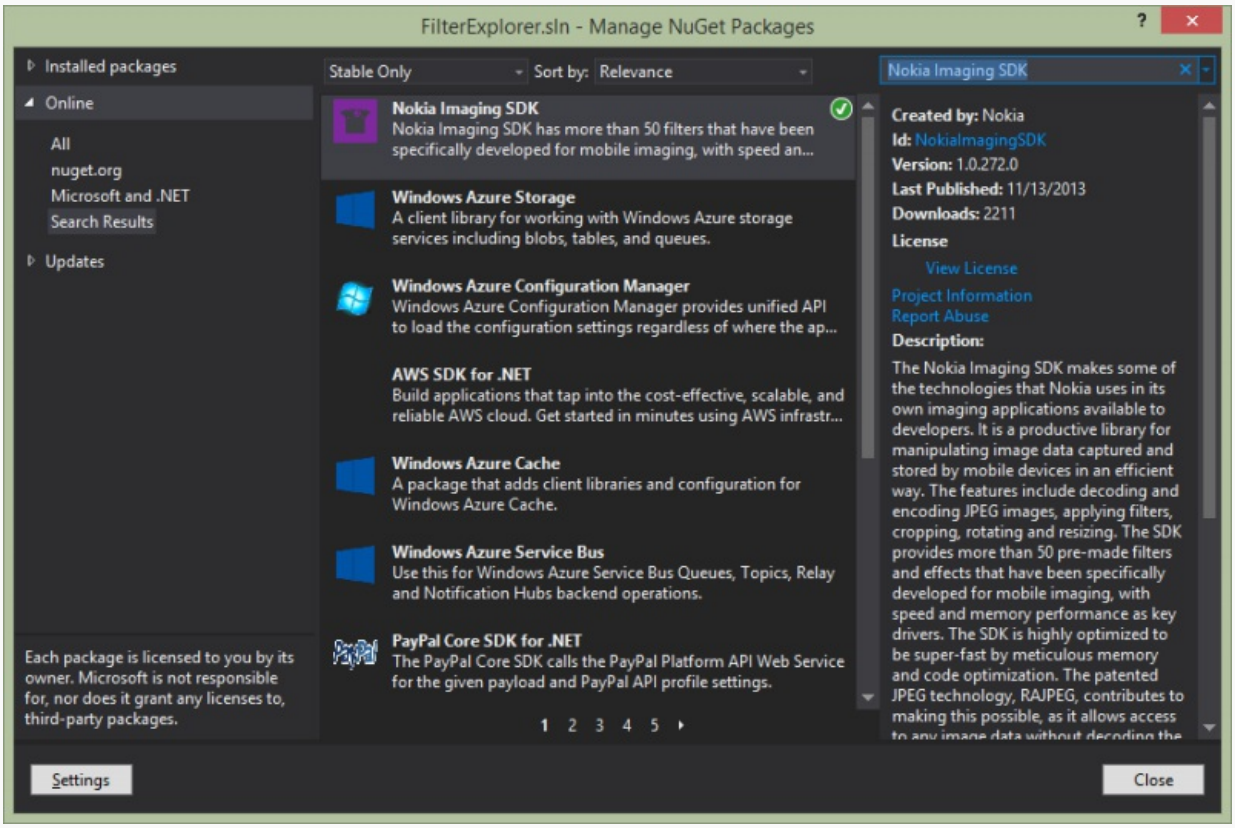
(C) Copyright Nokia 2014. All rights reserved.



Library Package Manager

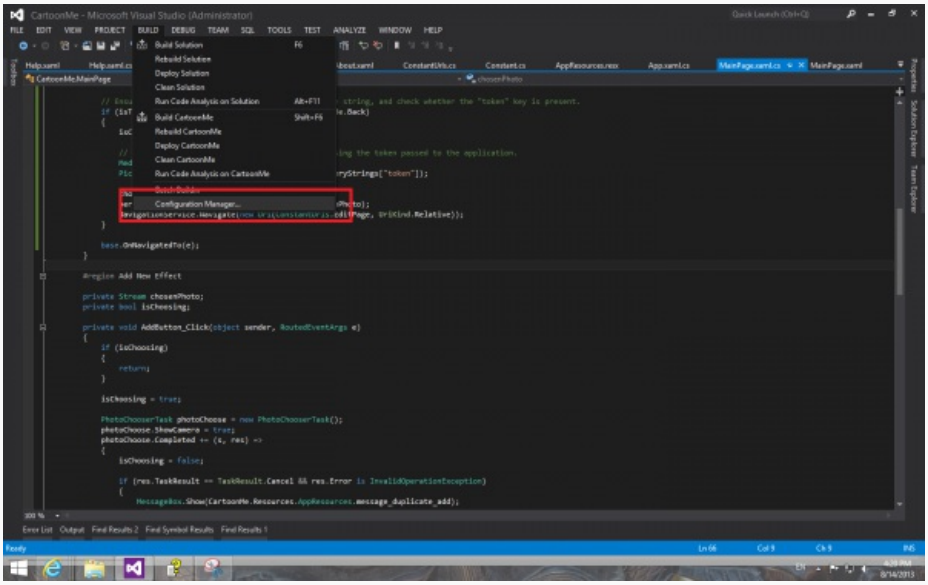
In the search box, type "Nokia Imaging SDK". The SDK entry will be displayed. Click the install button associated with the entry.

(C) Copyright Nokia 2014. All rights reserved.



Library Package Manager

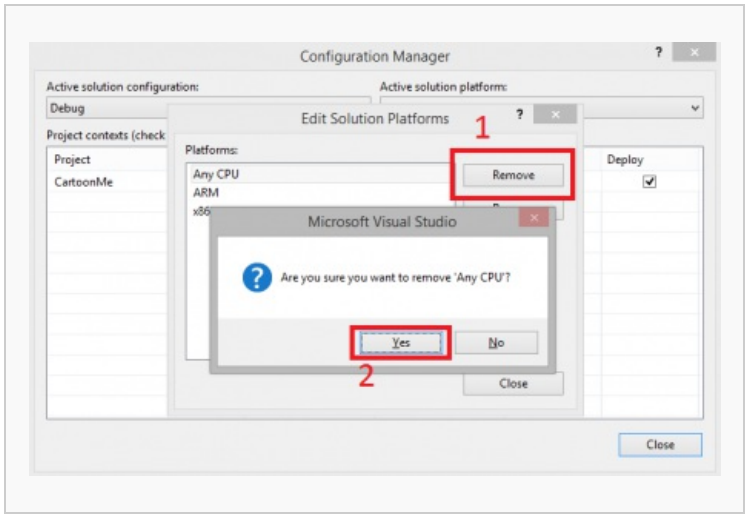
Special attention must be granted to the target architecture: the ARM library must be used on the package that will be installed on the phone, while the X86 library must be used for the application that will run on the emulator. So the "All CPU" configuration must be removed from the project, to leave only "ARM" and "X86".



Configuration Manager

In the configuration manager, pull down the "Active solution platforms" list, and select "<Edit...>". In the Edit Solution Platforms window, select "Any CPU" and "Remove". Only "ARM" and "x86" should remain in the list of solution platforms.

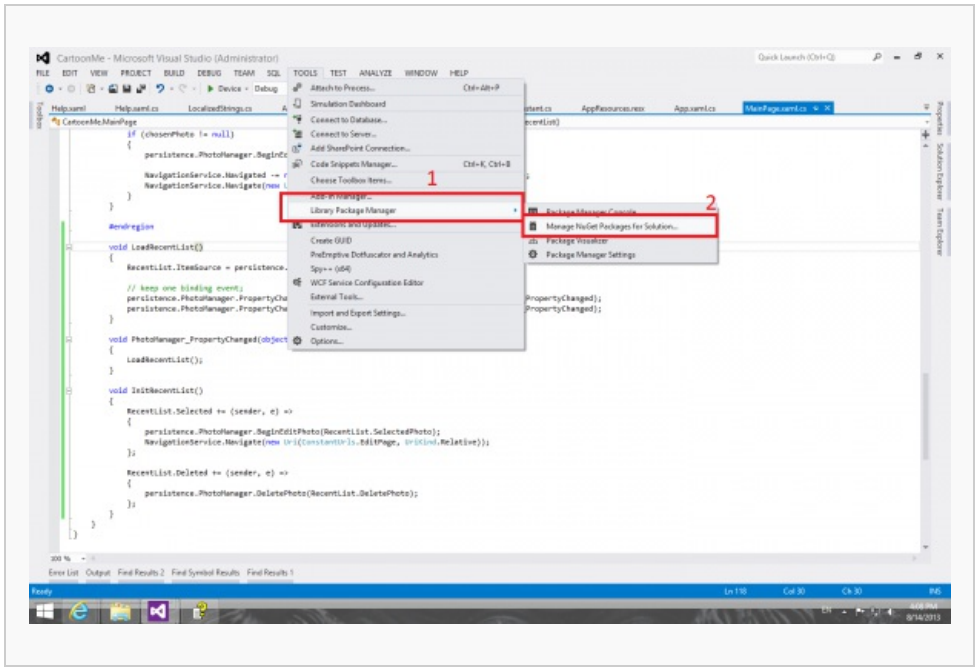
(C) Copyright Nokia 2014. All rights reserved.



select "Any CPU" and "Remove"

Installing the WriteableBitmapEx library

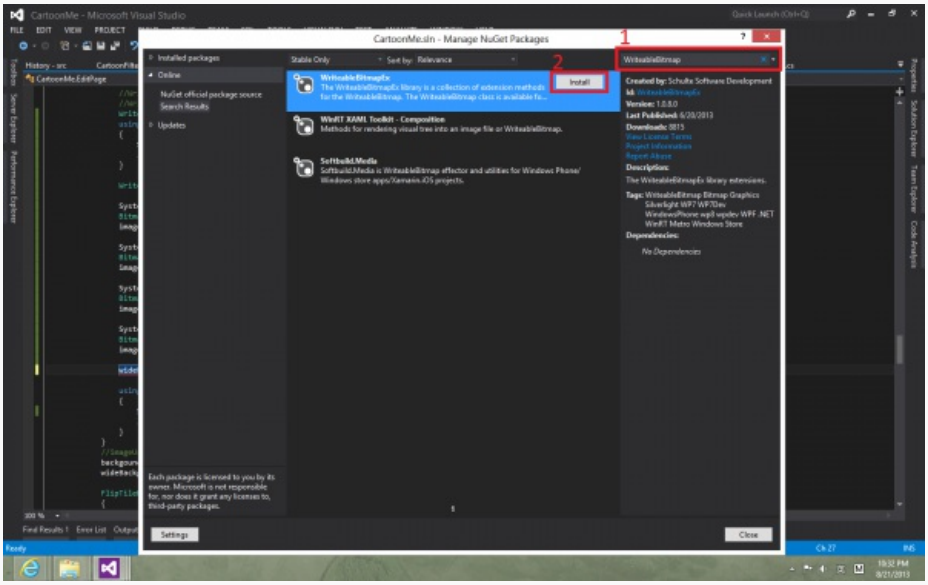
Start NuGet package manager: Tools menu > Library Package Manager > Manage NuGet Packages for Solution...



Library Package Manager

In the search box, type `WriteableBitmap`. The SDK entry will be displayed. Click the install button associated with the entry.

(C) Copyright Nokia 2014. All rights reserved.

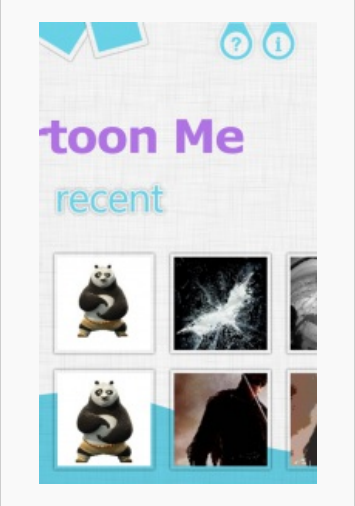


WriteableBitmapEx library

"Cartoon Me" app

"Cartoon Me" app is an example demonstrating the use of the cartoon filter of Nokia Imaging SDK with camera photos. This example app uses the camera and displays the viewfinder for taking a picture. The taken photo is then processed with the cartoon filters. The processed image can be saved in JPEG format into the camera roll album. You can also select an existing photo and apply an effect to it.

Add Photo

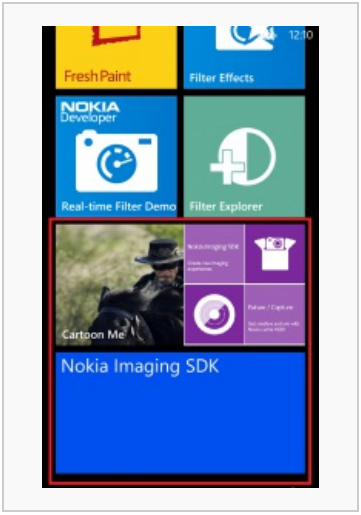


Recent Photos

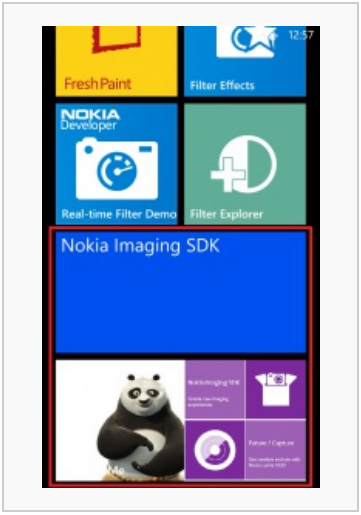
Cartoon Filter Photo

"Cartoon Me" app also has the feature of using FilterFactory.CreateCropFilter Method to crop the picture, splicing five photos for wide tile image, and creating live tile. The wide tile image is composed by five images.

(C) Copyright Nokia 2014. All rights reserved.



Live Tiles



Live Tiles

"Cartoon Me" app architecture

The example's main functionality is implemented in two classes: MainPage and EditPage. However, a simple, abstract base class for each filter is implemented: AbstractFilter.

Capture photo

The data of the captured photo is owned by MainPage class (see **MainPage.xaml.cs**). In MainPage, adding photo UI is designed by `UIFramework` reference. When click the add button, the following code is running. If the chose photo task returning is OK, app load image stream into a bitmap image and then write as jpeg, then app navigate to EditPage.

```
private void AddButton_Click(object sender, RoutedEventArgs e)
{
    if (isChoosing)
    {
        return;
    }

    isChoosing = true;

    PhotoChooserTask photoChoose = new PhotoChooserTask();
    photoChoose.ShowCamera = true;
    photoChoose.Completed += (s, res) =>
    {
        isChoosing = false;

        if (res.TaskResult == TaskResult.Cancel && res.Error is
InvalidOperationException)
        {
            MessageBox.Show(CartoonMe.Resources.AppResources.message_duplicate_add);
            return;
        }

        if (res.TaskResult == TaskResult.OK)
        {
            chosenPhoto = res.ChosenPhoto;

            if (null != res.ChosenPhoto)
            {
                DataContext dataContext = CartoonMe.DataContext.Singleton;
```


(C) Copyright Nokia 2014. All rights reserved.

```
                // Reset the streams
                dataContext.CreateStreams();

                // Use the largest possible dimensions
                WriteableBitmap bitmap = new WriteableBitmap(3552, 2448);
                try
                {
                    // Jpeg images can be used as such.
                    bitmap.LoadJpeg(chosenPhoto);
                    chosenPhoto.Position = 0;
                    chosenPhoto.CopyTo(dataContext.ImageStream);
                }
                catch (Exception /*ex*/)
                {
                    // Image format is not jpeg. Can be anything, so first
                    // load it into a bitmap image and then write as jpeg.
                    BitmapImage image = new BitmapImage();
                    image.SetSource(chosenPhoto);

                    bitmap = new WriteableBitmap(image);
                    bitmap.SaveJpeg(dataContext.ImageStream, image.PixelWidth,
image.PixelHeight, 0, 100);
                }
            }

            // Since PhotoChooserTask is not completed yet to front,
NavigationService.Navigate is not allowed here.
            // This is a workaround by one time navigation event subscribed, and
deplayed the navigation to another page.
            NavigationService.Navigated += new
NavigatedEventHandler(NavigationService_Navigated);
        }
    };
    photoChoose.Show();
}
```

Note that you need `ID_CAP_ISV_CAMERA` capability to access camera device.

Cartooning the image

The images are processed by the `EditPage` class:

```
/// <summary>
/// Takes the captured image and applies filters to it.
/// </summary>
private async void CreateCartoonImage()
{
    DataContext dataContext = CartoonMe.DataContext.Singleton;

    if (dataContext.ThumbStream == null)
    {
        // No captured image available!
        NavigationService.GoBack();
        return;
    }
```


(C) Copyright Nokia 2014. All rights reserved.

```
    // Create a session that we will use with all the filters
    IBuffer buffer = dataContext.ImageStream.GetWindowsRuntimeBuffer();
    EditingSession session = new EditingSession(buffer);

    // Perform the filtering
    using (session)
    {
        foreach (AbstractFilter filter in _filters)
        {
            filter.SetBuffer(buffer);
            filter.DefineFilter(session);

            await filter.RenderToBitmapAsync(session);
            //session.UndoAll();
        }
    }
}
```

Cartoon filter implements the abstract method `DefineFilter()`. Here is an example of how it is implemented in **CartoonFilter.cs**:

```
public override void DefineFilter(EditingSession session)
{
    session.AddFilter(FilterFactory.CreateCartoonFilter(_distinctEdges));
}
```

`RenderToBitmapAsync()` is implemented by the base class:

```
public async Task RenderToBitmapAsync(EditingSession session)
{
    await session.RenderToBitmapAsync(_filteredBitmap,
OutputOption.PreserveAspectRatio);
    _previewBitmap.Invalidate(); // Force a redraw
}
```

Save image into media library:

```
private async void OnSaveButtonClicked(object sender, RoutedEventArgs e)
{
    _progressIndicator.Text = AppResources.SavingText;
    _progressIndicator.IsVisible = true;
    SystemTray.SetProgressIndicator(this, _progressIndicator);

    MediaLibrary library = new MediaLibrary();

    DataContext dataContext = CartoonMe.DataContext.Singleton;

    try
    {
        AbstractFilter filter = _filters[0]; // cartoon filter
        EditingSession session = new
EditingSession(dataContext.ImageStream.GetWindowsRuntimeBuffer());

        using (session)
```


(C) Copyright Nokia 2014. All rights reserved.

```
        {
            filter.DefineFilter(session);
            IBuffer data = await session.RenderToJpegAsync();

            // Save the rendered image into a file
            library.SavePictureToCameraRoll(FileNamePrefix
                + DateTime.UtcNow.ToString(Constant.DateTimeFormat) +
Constant.ImageFileSuffix, data.AsStream());

        }

    }
    catch (Exception ex)
    {
        Debug.WriteLine("EditPage::SaveButton_Click(): Failed to save the image to
camera roll: "
            + ex.ToString());
    }

    _progressIndicator.IsVisible = false;
    SystemTray.SetProgressIndicator(this, _progressIndicator);

    NavigationService.GoBack();
}
```

Note that you need `ID_CAP_MEDIALIB_PHOTO` capability to store images into the media library.

Crop filter

We will use `FilterFactory.CreateCropFilter` Method to crop the picture for tile background image and wide background image. To crop the image, just add the new `CreateCropImage` method in **AbstractFilter.cs**, and use it in `EditPage` class:

```
DataContext dataContext = CartoonMe.DataContext.Singleton;
// Create a session that we will use with all the filters
IBuffer buffer = dataContext.ImageStream.GetWindowsRuntimeBuffer();
EditingSession session = new EditingSession(buffer);

// Perform the filtering
using (session)
{
    foreach (AbstractFilter filter in _filters)
    {
        filter.SetBuffer(buffer);
        filter.CropFilter(session);

        await filter.CropRenderToBitmapAsync(session);
        session.UndoAll();
    }
}
```

Implement the `CreateCropImage` method in `AbstractFilter` class:

```
public abstract void CropFilter(EditingSession session);
/// <summary>
/// Creates a filtered crip image from the given buffer.
```


(C) Copyright Nokia 2014. All rights reserved.

```
/// </summary>
public async void CreateCropImage()
{
    if (_buffer == null)
    {
        return;
    }

    EditingSession session = new EditingSession(_buffer);

    // Perform the filtering
    using (session)
    {
        CropFilter(session);
        await CropRenderToBitmapAsync(session);
    }
}

/// <summary>
/// Creates the preview buffers or resize them if needed.
/// </summary>
/// <param name="session">The session initialized with the desired jpeg.</param>
public async Task CropRenderToBitmapAsync(EditingSession session)
{
    await session.RenderToBitmapAsync(_cropFilteredBitmap,
OutputOption.PreserveAspectRatio);
    _cropBitmap.Invalidate(); // Force a redraw
}
```

Set Output Resolution in `AbstractFilter` class:

```
private WriteableBitmap _cropBitmap;
private Bitmap _cropFilteredBitmap;

/// <summary>
/// Creates the PreviewBuffers or resize them if needed.
/// </summary>
/// <param name="session">The session initialized with the desired jpeg.</param>
public void SetOutputResolution(int width, int height)
{
    ......

    _cropBitmap = new WriteableBitmap(355, 355);
    _cropFilteredBitmap = _cropBitmap.AsBitmap();
    CropImage.Source = _cropBitmap; // Force a redraw
}
```

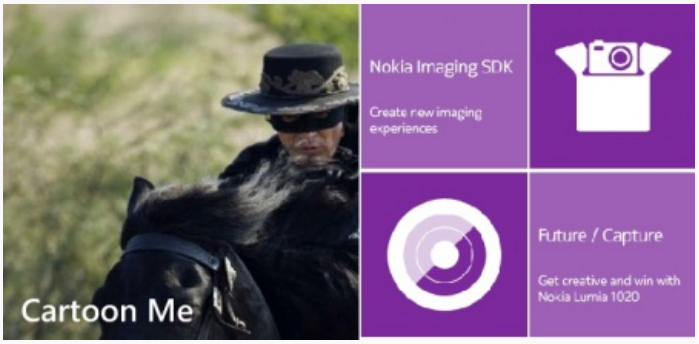
Implement the abstract method `CropFilter(EditingSession session)`, and set `Windows.Foundation.Rect` parameter in `CartoonFilter` class:

```
public override void CropFilter(EditingSession session)
{
    Windows.Foundation.Rect cropArea = new Windows.Foundation.Rect(0, 0, 355, 355);
    session.AddFilter(FilterFactory.CreateCropFilter(cropArea));
}
```


(C) Copyright Nokia 2014. All rights reserved.

Splicing images

Once all five images have been stitched together, we can create the wide tile image. The image like this:



Wide Live Tile

The `WriteableBitmapEx` library is a collection of extension methods for the `WriteableBitmap`. The library extends the `WriteableBitmap` class with elementary and fast (2D drawing) functionality, conversion methods and functions to combine (`blit`) `WriteableBitmaps`. Wide tile image size and coordinates:

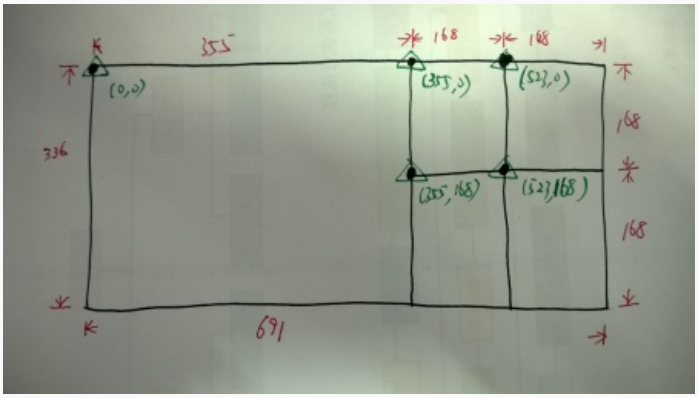


image size and coordinates

We use `Blit` method to combine five `WriteableBitmaps` in `EiitPage` class:

```
WriteableBitmap wbmp = new WriteableBitmap(_filters[0].CropImage.Source as
BitmapSource);

using (var stream = store.OpenFile(backgoundImagePath, System.IO.FileMode.OpenOrCreate))
{
    System.Windows.Media.Imaging.Extensions.SaveJpeg(wbmp, stream, MediumTilePixelWidth,
MediumTilePixelHeight, 0, 100);
    //wbmp.SaveJpeg(stream, bitmap.PixelWidth, bitmap.PixelHeight, 0, 100);
}

WriteableBitmap wideBitmap = new WriteableBitmap(WideTilePixelWidth,
WideTilePixelHeight);
```


(C) Copyright Nokia 2014. All rights reserved.

```
System.IO.Stream stream1 = Application.GetResourceStream(new
Uri(@"Assets/Images/01_imaging_sdk_text.png", UriKind.Relative)).Stream;
BitmapImage image1 = new BitmapImage();
image1.SetSource(stream1);

System.IO.Stream stream2 = Application.GetResourceStream(new
Uri(@"Assets/Images/02_imaging_sdk_icon.png", UriKind.Relative)).Stream;
BitmapImage image2 = new BitmapImage();
image2.SetSource(stream2);

System.IO.Stream stream3 = Application.GetResourceStream(new
Uri(@"Assets/Images/03_nokia_future_icon.png", UriKind.Relative)).Stream;
BitmapImage image3 = new BitmapImage();
image3.SetSource(stream3);

System.IO.Stream stream4 = Application.GetResourceStream(new
Uri(@"Assets/Images/04_nokia_future_text.png", UriKind.Relative)).Stream;
BitmapImage image4 = new BitmapImage();
image4.SetSource(stream4);

int leftImageWidth = (wbmp.PixelWidth > 355) ? 355 : wbmp.PixelWidth;
int leftImageHeight = (wbmp.PixelHeight > 336) ? 336 : wbmp.PixelHeight;

wideBitmap.Blit(new Rect(0, 0, 355, 336), new WriteableBitmap(wbmp), new Rect(0, 0,
wbmp.PixelWidth, wbmp.PixelHeight), WriteableBitmapExtensions.BlendMode.Additive);
wideBitmap.Blit(new Rect(355, 0, 168, 168), new WriteableBitmap(image1), new Rect(0, 0,
image1.PixelWidth, image1.PixelHeight), WriteableBitmapExtensions.BlendMode.Additive);
wideBitmap.Blit(new Rect(523, 0, 168, 168), new WriteableBitmap(image2), new Rect(0, 0,
image4.PixelWidth, image4.PixelHeight), WriteableBitmapExtensions.BlendMode.Additive);
wideBitmap.Blit(new Rect(355, 168, 168, 168), new WriteableBitmap(image3), new Rect(0,
0, image3.PixelWidth, image3.PixelHeight),
WriteableBitmapExtensions.BlendMode.Additive);
wideBitmap.Blit(new Rect(523, 168, 168, 168), new WriteableBitmap(image4), new Rect(0,
0, image4.PixelWidth, image4.PixelHeight),
WriteableBitmapExtensions.BlendMode.Additive);
```

Reference

- Core concepts (Nokia Imaging SDK documentation)
- Download and add the libraries to the project (Nokia Imaging SDK documentation)
- Quick Start (Nokia Imaging SDK documentation)
- Nokia Imaging SDK API Reference (Nokia Imaging SDK documentation)
- Microsoft Research Face SDK Beta (Microsoft Research)
- WriteableBitmapEx library (CodePlex)

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.