Filter and Custom Filter Management Framework for the Nokia Imaging SDK

This article explains Imaging Framework that implement for to use custom filters and built-in filters.

Nokia Original Imaging Effect Wiki Challenge (21 Jul 2014): Winners have been announced for the Nokia Original Imaging Effect Wiki Challenge. The Tweet-Off to decide which authors win a Nokia Lumia 630 starts today. Follow the Nokia Developer Twitter feed and retweet your favorite entries.

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

Introduction

The Nokia Imaging SDK provides more than 50 filters and effects for developers to use and combine them to create new filters. Moreover with SDK 1.0, It allows us to implement custom filters and image providers.

After Nokia Imaging SDK graduates from beta, there were lots of changes. I think it become more difficult to use, but It become more flexible and allows almost anything we want. With the new SDK, custom image source and custom filter introduced which I excited most.

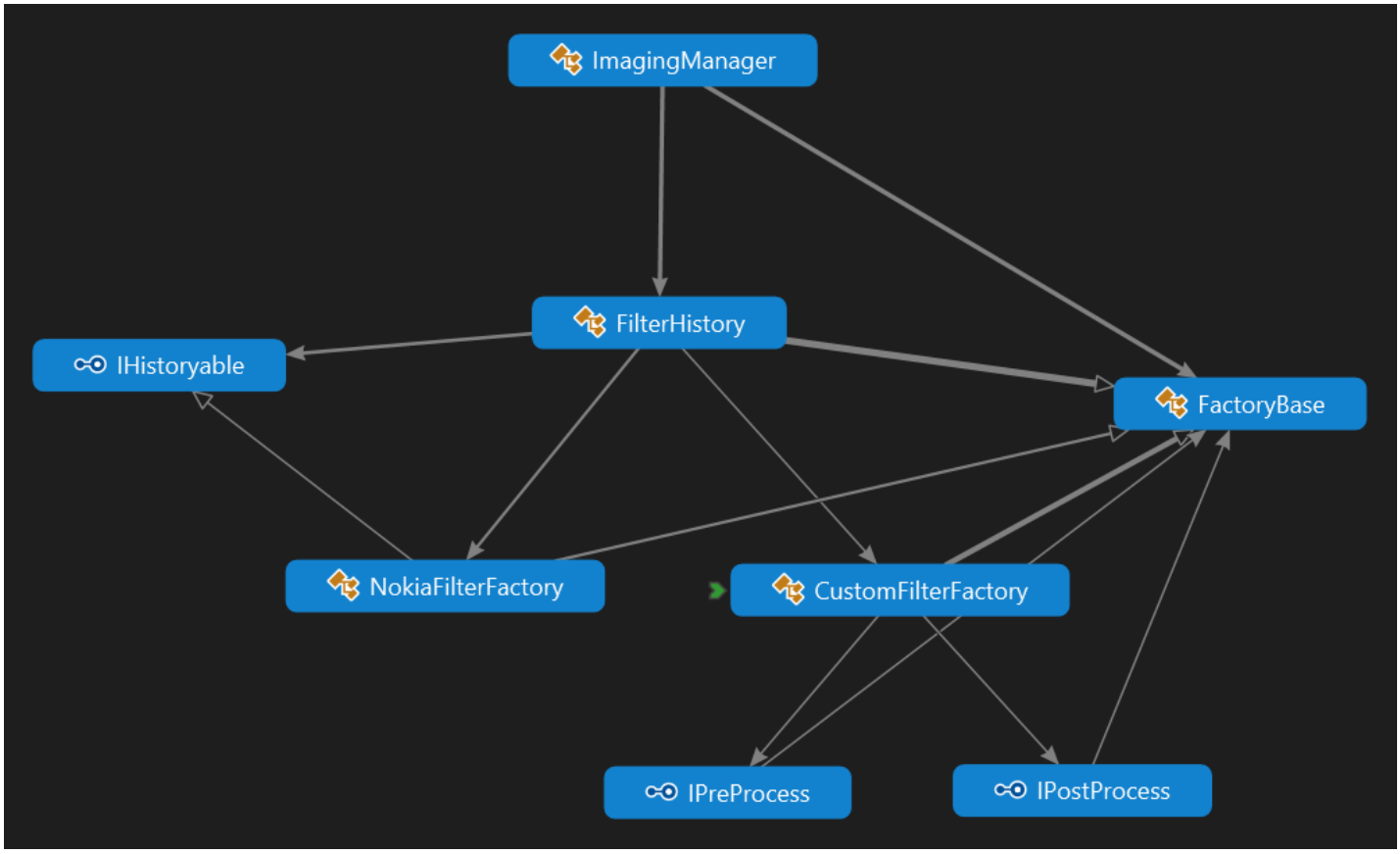
It become more difficult to use because of to add/undo filter we have to build pipeline like Image Source > Effect > ... > Renderer. In beta version of the SDK, we could simply add/undo filters only calling one method. About custom filters, It's confusing that we can't use them as we use built-in filters because CustomEffectBase is not an instance of IFilter interface. Also custom filter doesn't allow to change output size, sourcePixelRegion.ImagePixels and targetPixelRegion.ImagePixels implemented to be equal size iin the SDK.

Because of those problem I implement a framework which extends Nokia Imaging SDK, solves those problem and gives more extensibility for further needs. In this article I'll explain problems and how I solve those, custom filter implemantation and technical information about framework in detail. Also you can find sample filters which are used for image processing. Stay in touch, more custom filters will be added to framework in the future so watch here and github page.

Note: Please read all documents about Nokia Imaging SDK to understand this article.

Imaging Framework

Architecture



 (C) Copyright Nokia 2014. All rights reserved.

Interfaces

- `IHistoryable`

It used for factory classes, gives factory to historize filters. Example is `NokiaFilterFactory` which stores every built-in filters. This interface provides following methods;

```
bool CanUndo();
void Undo();
void UndoAll();
```

- `IPostProcess`

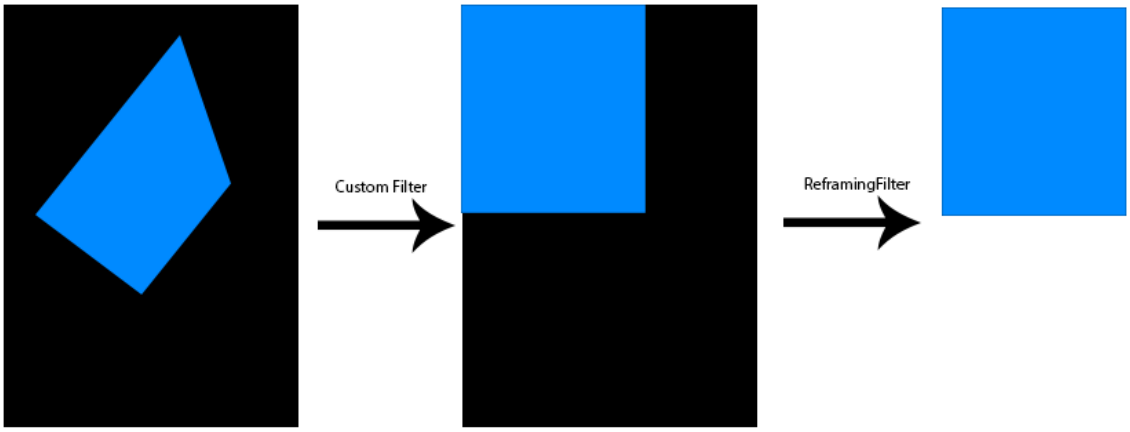
Use to extend custom filters. It provides custom filter to post process factory. So after custom filter processed, post process factory's filters processed. For example, `QuadTransform` filter implementation.

```
public class QuadTransformation : CustomEffectBase, IPostProcess
{
.
.
        protected override void OnProcess(PixelRegion sourcePixelRegion, PixelRegion
targetPixelRegion)
        {
        .
        .
        }

       public FactoryBase PostProcessFactory()
        {
            var factory = new NokiaFilterFactory(null);
            factory.Filters.Add(new ReframingFilter(new Rect(0, 0, Size.Width,
Size.Height), 0));

            return factory;
        }
}
```

- `IPreProcess`

Use to extend custom filters. It provides custom filter to pre process factory. So before custom filter processed, pre process factory's filters processed.

Note: `IPostProcess` solves the problem which is custom filter can't change output size of an image. Normally after we use a customfilter, if we want to change output size, we have to call `ReframingFilter`. For example, I need to get Quadrilateral object from image using QuadTranform filter in sample filters. Lets says image is 800x600 and I want object to be 300x300. My filter get object pixels to "targetPixelRegion". But "targetPixelRegion" dimension is fixed to 800x600. So after custom filter, I have to use `ReframingFilter`. As shown in the picture.



With `IPostProcess`, if `QuadTransform` filter added, `ReframingFilter` filter also added after it. So when filters applied to image, result will same as above. Also that if `Undo()` method called, both of `QuadTransform` and `ReframingFilter` filters will be removed from

 (C) Copyright Nokia 2014. All rights reserved.

history. Same as `IPreProcess` will be needed for different situation.

Classes

- `ImagingManager`

This class is the main core, everything happens here.

- `FactoryBase`

Factories derives from this class. It has `Input` and `Output` as `ImageProvider` so that it builds pipeline with another factory. There is already two factories; `CustomFilterFactory` for customfilter and `NokiaFilterFactory` for built-in filters. In future, there'll be more for same type of custom filters like convulation filters used for smoothing, blurring, etc.

- `CustomFilterFactory`

This factory stores only one custom filter. If custom filter has `IPreProcess` or/and `IPostProcess` interface, with `ApplyFilter` method, it builds pipeline inside it.

- `NokiaFilterFactory`

This factory stores built-in filters which are added consecutively.

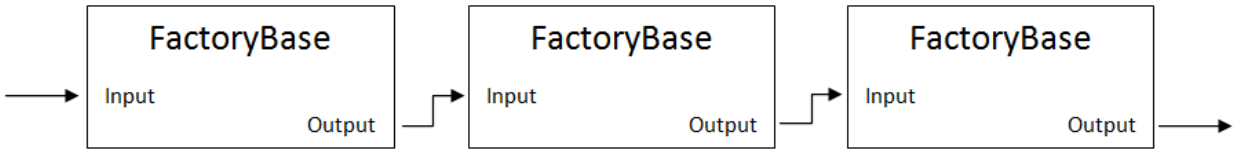
Note: Factories implemented because of custom filter use different than built-in filters. It inherited from `CustomEffectBase` and isn't an instance of `IFilter`. Despite built-in filters are an instance of `IFilter`. Built-in filters used like that;

```
var Effects = new FilterEffect(source);
Effects.Filters = new IFilter[]
{
      new AntiqueFilter(),
      new RotationFilter(35.0)
};
var renderer = new WriteableBitmapRenderer(Effects  outputBitmap);
```

But Custom filters as following;

```
var customFilter = new ThresholdFilter(source, 200, 200, 0);
renderer = new WriteableBitmapRenderer(customFilter, outputBitmap);
```

Because of difference use with the help of factories, pipeline can be built easily like that;



- `FilterHistory`

This class used in `ImagingManager`. It historizes factories and It has the key role building pipeline between factories. It connects one factory output to other as input as shown in diagram above. It provides `Add`, `Undo` and `UndoAll` method for filters.

How to implement custom filters?

I'll be explaining custom filter implementations in different article. Because of all of custom filters implemented for this framework can be used directly outside this framework. Only thing you have to consider, filters are instance of `IPreprocess` and `IPostProcess` interfaces, you need to apply pre/post filters manually when you use that filters.

You will find more detailed information by following this guide, "Implementations of Custom Filter for the Nokia Imaging SDK"

How to use?

As far as now, I described confusing things. The most easy part of this Imaging Framework is how to use. It's really simple.

First of all, add ImagingManager to page;

```
ImagingManager ImagingManager = new ImagingManager();
```

 (C) Copyright Nokia 2014. All rights reserved.

Then, start a new session;

```
await ImagingManager.StartSession(imageStream, imageControl);
```

- `ImageStream` is whatever you want to use from them.
- `imageControl` for viewing purpose. Use image control which is in page.

After that, add filters to be applied. The only restriction here, when initializing custom filters, you have to pass `ImageProvider`. But you don't have to worry about provider, use `DummySource`. `ImagingManager` handles and changes it.

```
var effects2 = new FilterEffect();
effects2.Filters = new IFilter[]
{
    new BlurFilter(32)
};

ImagingManager.AddFilter(effects2);

ImagingManager.AddFilter(new ThresholdFilter(ImagingManager.DummySource, 150, 100, 0));
ImagingManager.AddFilter(new ColorPaletteFilter(ImagingManager.DummySource,
AccentColorsPalette()));
ImagingManager.AddFilter(new PixelInterpolation(ImagingManager.DummySource, 64));
```

To render filters,

```
await ImagingManager.Render();
```

Finally releasing resources,

```
ImagingManager.DisposeSession();
```

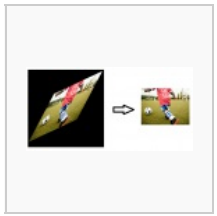
Note: In sample project, you can see more detailed use of framework.

Sample Filters

ThresholdFilter

ColorPaletteFilter

Pixel Interpolation



QuadTransformation (gets quadratic shape using corner points, then transforms shape to rectangular form)

CornerDetection (detects corners of quadratical shape in black&white image)

- `HistogramFilter` (calculates Red, Green, Blue and Luminance data for histogram of image)

Note: Histogram filter doesn't render image, it's for calculation purpose.

Contribution

If you want to contribute the project, add your custom filters or you want to improve framework, please contact me over github page.

Conclusion

It's my approach for custom filter and built-in filters for Nokia Imaging SDK. I'm not an expert in imaging process, I tried my best and continue to learn new things. I believe Imaging Framework will speed up development of filter based application. I'm also

 (C) Copyright Nokia 2014. All rights reserved.

using it with my applications and constantly updated and added new things to it. I'm planning to implement more filters that used for image processing.

 (C) Copyright Nokia 2014. All rights reserved.