Combining imaging filters to create new real-time camera effects

This article shows you how to combine existing filters provided by Nokia Imaging SDK to create totally new real-time camera filters and effects.

Introduction

Applying real-time camera filters and effects is an interesting and cool topic. However implementing new filters requires quite a lot of coding and tweaking on performance issues.

Nokia has made this task much easier with Nokia Imaging SDK for Nokia Lumia and Windows Phone 8 developers. There also exists good code samples like Real-time Filter Demo that can be used as a starting point.

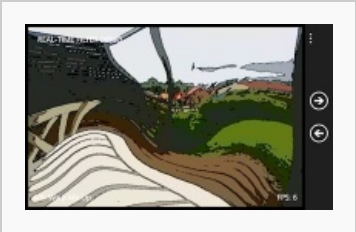
The SDK contains more than 50 different filters and effects. This is a quite huge collection but sometimes you need to apply also other imaging processing filters. The article shows you how to combine existing filters and effects to create new cool and/or useful effects.

Filter gallery

Magic Edge Pen

Night Vision

X-Ray



Warp cartoon

Build environment

All you need to try example code snippets presented on this page:

1. Make sure you have Windows 8 on your development PC
2. Download Windows Phone 8 SDK from Microsoft, this contains also a stand-alone Visual Studio Express 2012
3. Get Real-time Filter Demo sample project from Nokia
4. Open the Real-time Filter Demo project on Visual Studio
5. Use (that is *copy-paste*..) code presented on this article to test new combined real-time filters

Combining filters to create new effects

Rather than reinvent the wheel and create a custom app for viewing the filters, this article simply adds the filters to the existing framework in hte Real-time Filter Demo app.

All you need to do is extend the `switch` in the project's `NokiaImagingSDKEffects` class, and add each of the filters listed below to the frameworks "list of filters" (`filters`). The framework does everything else (setting the source, creating the effect and rendering) so this should be quite straightforward.

(C) Copyright Nokia 2014. All rights reserved.

Magic Edge Pen

Nokia Imaging SDK ships with plenty of filters and effects. However it misses actual edge detection algoritms like Sobel operator. There exists filters like Stamp, Sketch, Magic Pen that makes edges more distinquishable but still not quite.

To implement Sobel operator filter efficiently we could use native C++ code with DirectX APIs and custom code. Same way as described on a wiki article. But on this demo we want to stick on managed C# code and let Nokia Imaging SDK to handle heavy-weight imaging processing ops.

So we have to combine existing filters. This is not Sobel operator but it mimics edge detection to some degree:

```
// "Magic Edge Pen"
filters.Add(new GrayscaleFilter());
filters.Add(new MagicPenFilter());
filters.Add(new ContrastFilter(0.6));
```

So first real-time camera frames are transformed to grayscale, then Magic Pen is applied and finally some contrast is added to make edges more crisp and to remove some ambient noise.

Magic Edge Pen

Night Vision

Second example shows you how to create real Night Vision binoculars!

Really? Well this is not really a real effect as at least until now Nokia Lumia devices do not have infrared or thermal cameras (maybe someday?). So we have to implement a fake one as this needs daylight to function. Anyway this "simulates" night vision binoculurs by adjusting camera output color to a green image.

```
// "Night Vision"
filters.Add(new ColorAdjustFilter(-0.9f, 1.0f, -0.9f));
filters.Add(new SharpnessFilter(5));
filters.Add(new BrightnessFilter(-0.5));
filters.Add(new ContrastFilter(0.4));
```

The first filter adjusts colors by decreasing red and blue components and increasing green component on a RGB image frame. Then some sharpness and contrast is added but brightness level is lowered.

(C) Copyright Nokia 2014. All rights reserved.

Night Vision

X-Ray

X-Ray effect is quite similar to the Night Vision effect but here negative grayscale filter is used as a main tool. Code follows:

```
// "X-Ray"
filters.Add(new GrayscaleNegativeFilter());
filters.Add(new SharpnessFilter(6));
filters.Add(new ContrastFilter(-0.4));
```

X-Ray

Warp Cartoon

As a final case we present a Twister effect combined with Cartoon filtering. Probably this is totally useless combination but anyway it can be used to get some cool effects.

```
// "Warp Cartoon"
filters.Add(new WarpFilter(WarpEffect.Twister, 0.75));
filters.Add(new CartoonFilter(true));
```

As shown on the code this filter group contains Twister warp effect with strength of 0.75f and Cartoon filter with edges distinctable (*true* parameter).

Warp cartoon

(C) Copyright Nokia 2014. All rights reserved.

Analyzing filters and real-world testing

Okay now we have introduced some combined filters with code snippets above. You may have wondered how combinations evolved when coding them. Let's break down the filter definition for the Magic Edge Pen effect and see how it has been built.

As its name suggest the main component is the Magic Pen effect. See below the screenshot 1A that shows a photo of a building with only Magic Pen applied. Even if edges for the building are quite distinguishable the result is too colorful.

So for the next step we add the Grayscale filter and you can see the combination on the screenshot 1B. It's now better as we wanted to detect edges but still not good enough.

1A. Magic Pen

1B. Grayscale + Magic Pen

1C. Grayscale + Magic Pen + Contrast(-0.6)

1D. Grayscale + Magic Pen + Contrast(0.6)

Next let's try to play with contrast adjustments. On the screenshot 1C you can see that lowering contrast (in this case by factor -0.6) is not making the result any better but adding some contrast (factor 0.6) makes edges on the image more detectable. And that was what we wanted. Of course on the real app the contrast factor should be configurable or auto-adjustable but you can see the main principle from these examples.

However this combination is not true edge detection for all real-world scenarios. If there are lot of trees, water, big grass field etc. then this filter effect combination is not perfect. For example below is a shot taken from a boat moving on a lake. You can see that the city skyline is quite detectable but the waves on the water surface are quite noisy.

City skyline over a lake (Magic Edge Pen applied)

Measuring FPS

It's important to check performance when utilizing real-time filters. Test results below are measured using Nokia Lumia 620 that is a mid-range Windows Phone 8 device with 480x800 pixel screen, Qualcomm Snapdragon Dual-core CPU (1 MHz) and Adreno 305 GPU inside. Results may vary on different devices but test cases presented here should be comparable.

Note also that to display a frame rate on the screen code on `OnFPSChanged` method of **MainPage.xaml.cs** file should be uncommented.

(C) Copyright Nokia 2014. All rights reserved.

Real-Time Filter Demo

First results from the base demo app so that we have some baseline test results to compare with.

Real-time Filter Demo examples (single filters at once)

Filter	Average FPS	FilterFactory method used
Lomo	12	`CreateLomoFilter(0.5,0.5,LomoVignetting.High,LomoStyle.Yellow)`
Magic Pen	5	`CreateMagicPenFilter()`
Grayscale	13	`CreateGrayscaleFilter()`
Antique	10	`CreateAntiqueFilter()`
StampFilter	6	`CreateStampFilter(5,100)`
Cartoon	8	`CreateCartoonFilter(false)`

From these results we see that frame rate performance depends heavily on filters used. Magic Pen displays coolest effect but it's also the slowest one.

But what happens when we apply new combined filter groups used on this demo. Let's see...

Combined filters

Frame rate results for our new combined filters as tested on Nokia Lumia 620:

Combined filter cases

Filter Group	Average FPS	Filters combined
Magic Edge Pen	5	Grayscale + Magic Pen + Contrast
Night Vision	9	Color Adjust + Sharpness + Brightness + Contrast
X-Ray	9	Grayscale Negative + Sharpness + Contrast
Warp Me Up	6	Warp (Twister) + Cartoon

We can see that frame rates are not really fast but anyway acceptable if an user understands that heavy imaging ops are involved.

And we can see that the performance for the new Magic Edge Pen (with three combined filters) is same as the performance for the Magic Pen filter from Real-time Filter Demo.

Warp Me Up combined filter was first containing also Blur effect. If Blur with level 7 is added to this filter group then frame rate drops below FPS 1. Not good... And even with Blur level 1 added to the group FPS is about 3-4. So Blur was removed from the Warp Me Up effect.

Animated filter effect

Warning: This section has not yet been updated to Nokia Imaging SDK 1.0. It is still accurate for Nokia Imaging SDK Beta

This wiki article was about combining existing filters to create new effects. Of course that's not enough when developing real apps. We don't drill down on possibilities on enhancing the demo app but as a demonstration we try a small trick to add some animation to filters explained above.

First we need to get some animation ticks. Because this demo is a real-time filter app no timer is needed as it draws new frames at a speed it can. So simply add a frame counter as a private member to the `NokiaImagingSDKEffects` class:

```
private int frameCounter = -1;
```

And then at the beginning of `GetNewFrameAndApplyEffect` method counter is increased and animation phase factor is calculated using simple math:

```
frameCounter++; // increase counter
int animPhaseAngle = (frameCounter % 30) * 6; // value range [0, 180[
```


(C) Copyright Nokia 2014. All rights reserved.

```
float animPhaseFactor = (float)Math.Sin(Math.PI * animPhaseAngle / 180.0d); // value range [0, 1.0]
```

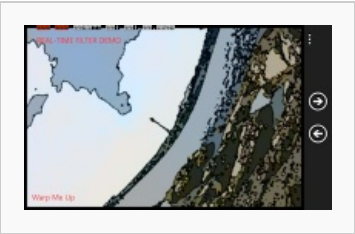
And finally filter case 4 (Warp Me Up) is changed with animated version:

```
// "Warp Me Up (animated)"
session.AddFilter(
      new FilterGroup(new IFilter[] {
            FilterFactory.CreateWarpFilter(WarpEffect.Twister, animPhaseFactor),
            FilterFactory.CreateCartoonFilter(true),
            FilterFactory.CreateFreeRotationFilter(Math.Max(360.0f - animPhaseAngle * 2.0f, 0.0f), RotationResizeMode.FitInside)
      }));
```

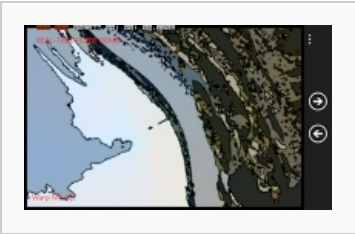
As you can see from the code now Twister effect strength on the Warp filter is oscillating between 0.0f (no effect at all) to 1.0f (full effect) as a frame counter increases by time while user is watching camera viewfinder on screen. And the filtered image is rotated around it's center point.

And here are some example screenshots how the animated filter effect works. The photo has been taken from a small islet on a lake near a city (on the skyline).

No rotation and no twist



Rotated 60 degrees and twisted



Rotated 120 degrees and twisted

Rotated 180 degrees and twisted

Summary

On this article we demonstrated that it's possible to create totally new camera effects just by combining existing Nokia Imaging SDK filters in a meaningful way.

This required only few lines of extra code so it was quite simple to enhance the Real-time Filter Demo app with these combined filters. More time was spent on finding best parameters for filters used on filter groups and testing filter combinations on real-world environment.

While waiting new filters and effects (like Sobel operator would be nice!) arriving to future Nokia Imaging SDK versions there is also much possibilities by combining existing filters.

References

Nokia Developer resources

- Nokia Imaging SDK
- Nokia Imaging SDK : API reference
- Real-time Filter Demo

(C) Copyright Nokia 2014. All rights reserved.

- Discussion thread about Nokia Imaging SDK feedback

Microsoft resources

- Windows Phone : develop

(C) Copyright Nokia 2014. All rights reserved.