

EdgePostFilterGen User's Guide

Table of Contents

About This Document	3
Purpose	3
Audience and Prerequisites	3
Related Documentation.....	3
SPA Tool	3
Edge Post	3
Typographic Conventions.....	3
1 Introducing EdgePostFilterGen	4
Application Overview.....	4
Limitations	4
2 Using EdgePostFilterGen.....	5
Invoking EdgePostFilterGen	5
Filter Description	5
Pixel Format	6
Number of Pixels per Iteration.....	6
Clamping	7

About This Document

Purpose

EdgePostFilterGen is a command-line utility for generating SPA-optimized loops for simple image kernel operations (such as Gaussian image filters) that can be used in Edge Post. The output is in the form of an SPA file ready to be fed to the SPU Pipelining Assembler (SPA) tool, `spa.exe`.

Audience and Prerequisites

EdgePostFilterGen is primarily intended for PlayStation®3 developers who want to use Edge Post without directly writing SPA code for simple image kernel operations.

Related Documentation

SPA Tool

The *SPU Pipelining Assembler (SPA) User's Guide*, included with the Edge package, describes the SPA, an assembly optimization tool.

Edge Post

The following documents provide complete usage and reference information about the Edge Post library:

- *PlayStation®Edge Library Overview*
- *PlayStation®Edge Post Library Reference*

Typographic Conventions

This document uses the following typographic conventions:

Convention	Meaning
fixed-width font	Indicates programming code and literals, such as processing instructions, register names, data types, events, and file names. Also indicates function, structure, and macro names.
<u>blue + underlined text</u>	Indicates a hyperlink (blue displays in color printers or online only).

1 Introducing EdgePostFilterGen

Application Overview

EdgePostFilterGen is a command-line utility for generating SPA-optimized loops for simple image kernel operations that can be used in Edge Post. Examples of such kernel operations include Gaussian image filters and similar filters. Any filter that can be expressed through a fixed number of offsets and coefficients around the kernel center can be generated by this tool.

The tool accepts as input a brief text file with a description of the filter and generates one function per filter. The output is in the form of an SPA file ready to be fed to the SPU Pipelining Assembler (SPA) tool, `spa.exe`.

Limitations

- Only four channel images are supported.
- Only three pixel types are supported: `fx16`, `u8n`, and `float`.

2 Using EdgePostFilterGen

Invoking EdgePostFilterGen

You can invoke EdgePostFilterGen from the command line:

```
EdgePostFilterGen.exe -o <outfile> <infile>
```

Where:

- <infile> is the text file with the filter descriptions.
- <outfile> is the name of the generated file. The output is in a form ready to be fed to `spa.exe`.

For example usage of this tool, see the following sample program (included with the Edge samples):
post-sample-hdr.

Filter Description

Create a text file with the following content:

```
# vertical 7 wide image filter
Filter Filter1x7_fx16_fx16
inputFormat fx16
outputFormat fx16
offset 0 -3
offset 0 -2
offset 0 -1
offset 0 0
offset 0 1
offset 0 2
offset 0 3
```

If you feed this file to EdgePostFilterGen.exe, it will generate an SPA function with the following C prototype:

```
extern "C"
void Filter1x7_fx16_fx16( void* output, const void* input, uint32_t input_stride,
const vec_float4* weights, uint32_t count );
```

This function will process *count* pixels and will perform the following operation on each pixel:

```
result = input[0,-3] * weight[0] +
         input[0,-2] * weight[1] +
         input[0,-1] * weight[2] +
         input[0,0]  * weight[3] +
         input[0,1]  * weight[4] +
         input[0,2]  * weight[5] +
         input[0,3]  * weight[6];
```

Input pixels are read from *input*. The parameter *input_stride* is the size in byte of one scanline for the input image.

Results are written sequentially to *output*.

Pixel Format

The filter definition also specifies the type of input/output pixels.

In the previous example, both input and output are declared as `fx16` format, which is one of the supported types. Supported types are limited to:

Type	Description
<code>float</code>	The <code>float</code> type is a full-precision, 32-bit per-channel floating-point pixel.
<code>u8n</code>	The <code>u8n</code> is 8-bit per channel, mapped to 0..1.
<code>fx16</code>	The <code>fx16</code> is a 16-bit per-channel, 0:5:11 fixed-point format.

No other formats are currently supported.

It is possible to have different input and output formats. For example, the following filter definition would generate the exact same filter operation (which is a vertical 1x7 blur) but would process an `fx16` input, and would output `u8n` pixels:

```
Filter Filter1x7_fx16_u8n
inputFormat fx16
outputFormat u8n
offset 0 -3
offset 0 -2
offset 0 -1
offset 0 0
offset 0 1
offset 0 2
offset 0 3
```

Number of Pixels per Iteration

The number of pixels processed per loop is variable and is dependent on the output pixel format:

- If the output format is `float`, one pixel is processed per loop.
- If the output format is `fx16`, two pixels are processed per loop.
- If the output format is `u8n`, four pixels are processed per loop.

It is possible to alter the number of pixels processed per loop by using `unrollCount`. For example:

```
Filter Filter1x7_fx16_u8n
inputFormat fx16
outputFormat u8n
unrollCount 8
offset 0 -3
offset 0 -2
offset 0 -1
offset 0 0
offset 0 1
offset 0 2
offset 0 3
```

This filter would generate an SPA loop that processes 8 pixels at a time. This can sometimes improve performance of the generated function.

Clamping

It is possible to clamp the final result to some specific value. For example:

```
Filter Filter1x7_fx16_fx16
numChannels 4
inputFormat fx16
outputFormat u8n
unrollCount 8
clamp true
clampValue 1.0
offset 0 -3
offset 0 -2
offset 0 -1
offset 0 0
offset 0 1
offset 0 2
offset 0 3
```

This filter will clamp the result of the kernel operation to 1.0. This is desirable if, for example, you are outputting to the u8n pixel format.

Clamping is not implied if outputting to the u8n format.