FXAA

FXAA.vert

```
// File:            es3-kepler\FXAA\assets\shaders/FXAA.vert
// SDK Version: v2.0
// Email:           gameworks@nvidia.com
// Site:            http://developer.nvidia.com/
// attribute vec2 aPosition;
// attribute vec2 aTexCoord;

varying vec2 vTexCoord;

void main(void)
{
    vTexCoord = gl_Vertex;
    gl_Position = gl_Vertex * 2.0 - 1.0;
}
```

FXAA_Default.frag

```
#define FXAA_PC 1
#define FXAA_GLSL_130 1
#define FXAA_QUALITY__PRESET 12

#define FXAA_GREEN_AS_LUMA 1


/*--------------------------------------------------------------------------*/
#ifndef FXAA_PC_CONSOLE
    //
    // The console algorithm for PC is included
    // for developers targeting really low spec machines.
    // Likely better to just run FXAA_PC, and use a really low preset.
    //
    #define FXAA_PC_CONSOLE 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_GLSL_120
    #define FXAA_GLSL_120 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_GLSL_130
    #define FXAA_GLSL_130 0
#endif
```

```
/*--------------------------------------------------------------------------*/
#ifndef FXAA_HLSL_3
    #define FXAA_HLSL_3 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_HLSL_4
    #define FXAA_HLSL_4 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_HLSL_5
    #define FXAA_HLSL_5 0
#endif
/*==========================================================================
================*/
#ifndef FXAA_GREEN_AS_LUMA
    //
    // For those using non-linear color,
    // and either not able to get luma in alpha, or not wanting to,
    // this enables FXAA to run using green as a proxy for luma.
    // So with this enabled, no need to pack luma in alpha.
    //
    // This will turn off AA on anything which lacks some amount of green.
    // Pure red and blue or combination of only R and B, will get no AA.
    //
    // Might want to lower the settings for both,
    //     fxaaConsoleEdgeThresholdMin
    //     fxaaQualityEdgeThresholdMin
    // In order to insure AA does not get turned off on colors
    // which contain a minor amount of green.
    //
    // 1 = On.
    // 0 = Off.
    //
    #define FXAA_GREEN_AS_LUMA 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_EARLY_EXIT
    //
    // Controls algorithm's early exit path.
    // On PS3 turning this ON adds 2 cycles to the shader.
    // On 360 turning this OFF adds 10ths of a millisecond to the shader.
    // Turning this off on console will result in a more blurry image.
    // So this defaults to on.
    //
```

```
    // 1 = On.
    // 0 = Off.
    //
    #define FXAA_EARLY_EXIT 1
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_DISCARD
    //
    // Only valid for PC OpenGL currently.
    // Probably will not work when FXAA_GREEN_AS_LUMA = 1.
    //
    // 1 = Use discard on pixels which don't need AA.
    //     For APIs which enable concurrent TEX+ROP from same surface.
    // 0 = Return unchanged color on pixels which don't need AA.
    //
    #define FXAA_DISCARD 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_FAST_PIXEL_OFFSET
    //
    // Used for GLSL 120 only.
    //
    // 1 = GL API supports fast pixel offsets
    // 0 = do not use fast pixel offsets
    //
    #ifdef GL_EXT_gpu_shader4
        #define FXAA_FAST_PIXEL_OFFSET 1
    #endif
    #ifdef GL_NV_gpu_shader5
        #define FXAA_FAST_PIXEL_OFFSET 1
    #endif
    #ifdef GL_ARB_gpu_shader5
        #define FXAA_FAST_PIXEL_OFFSET 1
    #endif
    #ifndef FXAA_FAST_PIXEL_OFFSET
        #define FXAA_FAST_PIXEL_OFFSET 0
    #endif
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_GATHER4_ALPHA
    //
    // 1 = API supports gather4 on alpha channel.
    // 0 = API does not support gather4 on alpha channel.
    //
```

```
    #if (FXAA_HLSL_5 == 1)
        #define FXAA_GATHER4_ALPHA 1
    #endif
    #ifdef GL_ARB_gpu_shader5
        #define FXAA_GATHER4_ALPHA 1
    #endif
    #ifdef GL_NV_gpu_shader5
        #define FXAA_GATHER4_ALPHA 1
    #endif
    #ifndef FXAA_GATHER4_ALPHA
        #define FXAA_GATHER4_ALPHA 0
    #endif
#endif


/*============================================================================
============================
                        FXAA QUALITY - TUNING KNOBS
------------------------------------------------------------------------------
NOTE the other tuning knobs are now in the shader function inputs!
============================================================================
============================*/
#ifndef FXAA_QUALITY__PRESET
    //
    // Choose the quality preset.
    // This needs to be compiled into the shader as it effects code.
    // Best option to include multiple presets is to
    // in each shader define the preset, then include this file.
    //
    // OPTIONS
    // -----------------------------------------------------------------------
    // 10 to 15 - default medium dither (10=fastest, 15=highest quality)
    // 20 to 29 - less dither, more expensive (20=fastest, 29=highest quality)
    // 39       - no dither, very expensive
    //
    // NOTES
    // -----------------------------------------------------------------------
    // 12 = slightly faster then FXAA 3.9 and higher edge quality (default)
    // 13 = about same speed as FXAA 3.9 and better than 12
    // 23 = closest to FXAA 3.9 visually and performance wise
    //  _ = the lowest digit is directly related to performance
    // _  = the highest digit is directly related to style
    //
    #define FXAA_QUALITY__PRESET 12
```

```
#endif


/*============================================================
==================

                        FXAA QUALITY - PRESETS

============================================================
==================*/

/*============================================================
==================
                    FXAA QUALITY - MEDIUM DITHER PRESETS
============================================================
==================*/
#if (FXAA_QUALITY__PRESET == 10)
    #define FXAA_QUALITY__PS 3
    #define FXAA_QUALITY__P0 1.5
    #define FXAA_QUALITY__P1 3.0
    #define FXAA_QUALITY__P2 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 11)
    #define FXAA_QUALITY__PS 4
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 3.0
    #define FXAA_QUALITY__P3 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 12)
    #define FXAA_QUALITY__PS 5
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 4.0
    #define FXAA_QUALITY__P4 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 13)
    #define FXAA_QUALITY__PS 6
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
```

```
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 4.0
    #define FXAA_QUALITY__P5 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 14)
    #define FXAA_QUALITY__PS 7
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 4.0
    #define FXAA_QUALITY__P6 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 15)
    #define FXAA_QUALITY__PS 8
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 2.0
    #define FXAA_QUALITY__P6 4.0
    #define FXAA_QUALITY__P7 12.0
#endif

/*============================================================================
==================
                        FXAA QUALITY - LOW DITHER PRESETS
============================================================================
==================*/
#if (FXAA_QUALITY__PRESET == 20)
    #define FXAA_QUALITY__PS 3
    #define FXAA_QUALITY__P0 1.5
    #define FXAA_QUALITY__P1 2.0
    #define FXAA_QUALITY__P2 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 21)
    #define FXAA_QUALITY__PS 4
    #define FXAA_QUALITY__P0 1.0
```

```
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 22)
    #define FXAA_QUALITY__PS 5
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 23)
    #define FXAA_QUALITY__PS 6
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 24)
    #define FXAA_QUALITY__PS 7
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 3.0
    #define FXAA_QUALITY__P6 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 25)
    #define FXAA_QUALITY__PS 8
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 2.0
    #define FXAA_QUALITY__P6 4.0
```

```
    #define FXAA_QUALITY__P7 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 26)
    #define FXAA_QUALITY__PS 9
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 2.0
    #define FXAA_QUALITY__P6 2.0
    #define FXAA_QUALITY__P7 4.0
    #define FXAA_QUALITY__P8 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 27)
    #define FXAA_QUALITY__PS 10
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 2.0
    #define FXAA_QUALITY__P6 2.0
    #define FXAA_QUALITY__P7 2.0
    #define FXAA_QUALITY__P8 4.0
    #define FXAA_QUALITY__P9 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 28)
    #define FXAA_QUALITY__PS 11
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 2.0
    #define FXAA_QUALITY__P6 2.0
    #define FXAA_QUALITY__P7 2.0
    #define FXAA_QUALITY__P8 2.0
    #define FXAA_QUALITY__P9 4.0
    #define FXAA_QUALITY__P10 8.0
#endif
```

```
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 29)
    #define FXAA_QUALITY__PS 12
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 2.0
    #define FXAA_QUALITY__P6 2.0
    #define FXAA_QUALITY__P7 2.0
    #define FXAA_QUALITY__P8 2.0
    #define FXAA_QUALITY__P9 2.0
    #define FXAA_QUALITY__P10 4.0
    #define FXAA_QUALITY__P11 8.0
#endif

/*============================================================================
=================
                        FXAA QUALITY - EXTREME QUALITY
============================================================================
=================*/
#if (FXAA_QUALITY__PRESET == 39)
    #define FXAA_QUALITY__PS 12
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.0
    #define FXAA_QUALITY__P2 1.0
    #define FXAA_QUALITY__P3 1.0
    #define FXAA_QUALITY__P4 1.0
    #define FXAA_QUALITY__P5 1.5
    #define FXAA_QUALITY__P6 2.0
    #define FXAA_QUALITY__P7 2.0
    #define FXAA_QUALITY__P8 2.0
    #define FXAA_QUALITY__P9 2.0
    #define FXAA_QUALITY__P10 4.0
    #define FXAA_QUALITY__P11 8.0
#endif



/*============================================================================
=================

                        API PORTING
```

```
============================================================
=================*/
#if (FXAA_GLSL_120 == 1) || (FXAA_GLSL_130 == 1)
    #define FxaaBool bool
    #define FxaaDiscard discard
    #define FxaaFloat float
    #define FxaaFloat2 vec2
    #define FxaaFloat3 vec3
    #define FxaaFloat4 vec4
    #define FxaaHalf float
    #define FxaaHalf2 vec2
    #define FxaaHalf3 vec3
    #define FxaaHalf4 vec4
    #define FxaaInt2 ivec2
    #define FxaaSat(x) clamp(x, 0.0, 1.0)
    #define FxaaTex sampler2D
#else
    #define FxaaBool bool
    #define FxaaDiscard clip(-1)
    #define FxaaFloat float
    #define FxaaFloat2 float2
    #define FxaaFloat3 float3
    #define FxaaFloat4 float4
    #define FxaaHalf half
    #define FxaaHalf2 half2
    #define FxaaHalf3 half3
    #define FxaaHalf4 half4
    #define FxaaSat(x) saturate(x)
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_GLSL_120 == 1)
    // Requires,
    //  #version 120
    // And at least,
    //  #extension GL_EXT_gpu_shader4 : enable
    //  (or set FXAA_FAST_PIXEL_OFFSET 1 to work like DX9)
    #define FxaaTexTop(t, p) texture2DLod(t, p, 0.0)
    #if (FXAA_FAST_PIXEL_OFFSET == 1)
        #define FxaaTexOff(t, p, o, r) texture2DLodOffset(t, p, 0.0, o)
    #else
        #define FxaaTexOff(t, p, o, r) texture2DLod(t, p + (o * r), 0.0)
    #endif
    #if (FXAA_GATHER4_ALPHA == 1)
```

```
            // use #extension GL_ARB_gpu_shader5 : enable
            #define FxaaTexAlpha4(t, p) textureGather(t, p, 3)
            #define FxaaTexOffAlpha4(t, p, o) textureGatherOffset(t, p, o, 3)
            #define FxaaTexGreen4(t, p) textureGather(t, p, 1)
            #define FxaaTexOffGreen4(t, p, o) textureGatherOffset(t, p, o, 1)
        #endif
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_GLSL_130 == 1)
        // Requires "#version 130" or better
        #define FxaaTexTop(t, p) textureLod(t, p, 0.0)
        #define FxaaTexOff(t, p, o, r) textureLodOffset(t, p, 0.0, o)
        #if (FXAA_GATHER4_ALPHA == 1)
            // use #extension GL_ARB_gpu_shader5 : enable
            #define FxaaTexAlpha4(t, p) textureGather(t, p, 3)
            #define FxaaTexOffAlpha4(t, p, o) textureGatherOffset(t, p, o, 3)
            #define FxaaTexGreen4(t, p) textureGather(t, p, 1)
            #define FxaaTexOffGreen4(t, p, o) textureGatherOffset(t, p, o, 1)
        #endif
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_HLSL_3 == 1) || (FXAA_360 == 1) || (FXAA_PS3 == 1)
        #define FxaaInt2 float2
        #define FxaaTex sampler2D
        #define FxaaTexTop(t, p) tex2Dlod(t, float4(p, 0.0, 0.0))
        #define FxaaTexOff(t, p, o, r) tex2Dlod(t, float4(p + (o * r), 0, 0))
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_HLSL_4 == 1)
        #define FxaaInt2 int2
        struct FxaaTex { SamplerState smpl; Texture2D tex; };
        #define FxaaTexTop(t, p) t.tex.SampleLevel(t.smpl, p, 0.0)
        #define FxaaTexOff(t, p, o, r) t.tex.SampleLevel(t.smpl, p, 0.0, o)
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_HLSL_5 == 1)
        #define FxaaInt2 int2
        struct FxaaTex { SamplerState smpl; Texture2D tex; };
        #define FxaaTexTop(t, p) t.tex.SampleLevel(t.smpl, p, 0.0)
        #define FxaaTexOff(t, p, o, r) t.tex.SampleLevel(t.smpl, p, 0.0, o)
        #define FxaaTexAlpha4(t, p) t.tex.GatherAlpha(t.smpl, p)
        #define FxaaTexOffAlpha4(t, p, o) t.tex.GatherAlpha(t.smpl, p, o)
        #define FxaaTexGreen4(t, p) t.tex.GatherGreen(t.smpl, p)
        #define FxaaTexOffGreen4(t, p, o) t.tex.GatherGreen(t.smpl, p, o)
```

```
#endif


/*============================================================
==================
                        GREEN AS LUMA OPTION SUPPORT FUNCTION
============================================================
==================*/
#if (FXAA_GREEN_AS_LUMA == 0)
    FxaaFloat FxaaLuma(FxaaFloat4 rgba) { return rgba.w; }
#else
    FxaaFloat FxaaLuma(FxaaFloat4 rgba) { return rgba.y; }
#endif




/*============================================================
==================

                                FXAA3 QUALITY - PC

============================================================
==================*/
#if (FXAA_PC == 1)
/*--------------------------------------------------------------------------*/
FxaaFloat4 FxaaPixelShader(
    //
    // Use noperspective interpolation here (turn off perspective interpolation).
    // {xy} = center of pixel
    FxaaFloat2 pos,
    //
    // Used only for FXAA Console, and not used on the 360 version.
    // Use noperspective interpolation here (turn off perspective interpolation).
    // {xy__} = upper left of pixel
    // {__zw} = lower right of pixel
    FxaaFloat4 fxaaConsolePosPos,
    //
    // Input color texture.
    // {rgb_} = color in linear or perceptual color space
    // if (FXAA_GREEN_AS_LUMA == 0)
    //     {___a} = luma in perceptual color space (not linear)
    FxaaTex tex,
    //
```

```
// Only used on the optimized 360 version of FXAA Console.
// For everything but 360, just use the same input here as for "tex".
// For 360, same texture, just alias with a 2nd sampler.
// This sampler needs to have an exponent bias of -1.
FxaaTex fxaaConsole360TexExpBiasNegOne,
//
// Only used on the optimized 360 version of FXAA Console.
// For everything but 360, just use the same input here as for "tex".
// For 360, same texture, just alias with a 3nd sampler.
// This sampler needs to have an exponent bias of -2.
FxaaTex fxaaConsole360TexExpBiasNegTwo,
//
// Only used on FXAA Quality.
// This must be from a constant/uniform.
// {x_} = 1.0/screenWidthInPixels
// {_y} = 1.0/screenHeightInPixels
FxaaFloat2 fxaaQualityRcpFrame,
//
// Only used on FXAA Console.
// This must be from a constant/uniform.
// This effects sub-pixel AA quality and inversely sharpness.
//     Where N ranges between,
//         N = 0.50 (default)
//         N = 0.33 (sharper)
// {x___} = -N/screenWidthInPixels
// {_y__} = -N/screenHeightInPixels
// {__z_} =  N/screenWidthInPixels
// {___w} =  N/screenHeightInPixels
FxaaFloat4 fxaaConsoleRcpFrameOpt,
//
// Only used on FXAA Console.
// Not used on 360, but used on PS3 and PC.
// This must be from a constant/uniform.
// {x___} = -2.0/screenWidthInPixels
// {_y__} = -2.0/screenHeightInPixels
// {__z_} =  2.0/screenWidthInPixels
// {___w} =  2.0/screenHeightInPixels
FxaaFloat4 fxaaConsoleRcpFrameOpt2,
//
// Only used on FXAA Console.
// Only used on 360 in place of fxaaConsoleRcpFrameOpt2.
// This must be from a constant/uniform.
// {x___} =  8.0/screenWidthInPixels
// {_y__} =  8.0/screenHeightInPixels
```

```
// {__z_} = -4.0/screenWidthInPixels
// {___w} = -4.0/screenHeightInPixels
FxaaFloat4 fxaaConsole360RcpFrameOpt2,
//
// Only used on FXAA Quality.
// This used to be the FXAA_QUALITY__SUBPIX define.
// It is here now to allow easier tuning.
// Choose the amount of sub-pixel aliasing removal.
// This can effect sharpness.
//   1.00 - upper limit (softer)
//   0.75 - default amount of filtering
//   0.50 - lower limit (sharper, less sub-pixel aliasing removal)
//   0.25 - almost off
//   0.00 - completely off
FxaaFloat fxaaQualitySubpix,
//
// Only used on FXAA Quality.
// This used to be the FXAA_QUALITY__EDGE_THRESHOLD define.
// It is here now to allow easier tuning.
// The minimum amount of local contrast required to apply algorithm.
//   0.333 - too little (faster)
//   0.250 - low quality
//   0.166 - default
//   0.125 - high quality
//   0.063 - overkill (slower)
FxaaFloat fxaaQualityEdgeThreshold,
//
// Only used on FXAA Quality.
// This used to be the FXAA_QUALITY__EDGE_THRESHOLD_MIN define.
// It is here now to allow easier tuning.
// Trims the algorithm from processing darks.
//   0.0833 - upper limit (default, the start of visible unfiltered edges)
//   0.0625 - high quality (faster)
//   0.0312 - visible limit (slower)
// Special notes when using FXAA_GREEN_AS_LUMA,
//   Likely want to set this to zero.
//   As colors that are mostly not-green
//   will appear very dark in the green channel!
//   Tune by looking at mostly non-green content,
//   then start at zero and increase until aliasing is a problem.
FxaaFloat fxaaQualityEdgeThresholdMin,
//
// Only used on FXAA Console.
// This used to be the FXAA_CONSOLE__EDGE_SHARPNESS define.
```

// It is here now to allow easier tuning.
// This does not effect PS3, as this needs to be compiled in.
//   Use FXAA_CONSOLE__PS3_EDGE_SHARPNESS for PS3.
//   Due to the PS3 being ALU bound,
//   there are only three safe values here: 2 and 4 and 8.
//   These options use the shaders ability to a free *|/ by 2|4|8.
// For all other platforms can be a non-power of two.
//   8.0 is sharper (default!!!)
//   4.0 is softer
//   2.0 is really soft (good only for vector graphics inputs)
FxaaFloat fxaaConsoleEdgeSharpness,
//
// Only used on FXAA Console.
// This used to be the FXAA_CONSOLE__EDGE_THRESHOLD define.
// It is here now to allow easier tuning.
// This does not effect PS3, as this needs to be compiled in.
//   Use FXAA_CONSOLE__PS3_EDGE_THRESHOLD for PS3.
//   Due to the PS3 being ALU bound,
//   there are only two safe values here: 1/4 and 1/8.
//   These options use the shaders ability to a free *|/ by 2|4|8.
// The console setting has a different mapping than the quality setting.
// Other platforms can use other values.
//   0.125 leaves less aliasing, but is softer (default!!!)
//   0.25 leaves more aliasing, and is sharper
FxaaFloat fxaaConsoleEdgeThreshold,
//
// Only used on FXAA Console.
// This used to be the FXAA_CONSOLE__EDGE_THRESHOLD_MIN define.
// It is here now to allow easier tuning.
// Trims the algorithm from processing darks.
// The console setting has a different mapping than the quality setting.
// This only applies when FXAA_EARLY_EXIT is 1.
// This does not apply to PS3,
// PS3 was simplified to avoid more shader instructions.
//   0.06 - faster but more aliasing in darks
//   0.05 - default
//   0.04 - slower and less aliasing in darks
// Special notes when using FXAA_GREEN_AS_LUMA,
//   Likely want to set this to zero.
//   As colors that are mostly not-green
//   will appear very dark in the green channel!
//   Tune by looking at mostly non-green content,
//   then start at zero and increase until aliasing is a problem.
FxaaFloat fxaaConsoleEdgeThresholdMin,

```
    //
    // Extra constants for 360 FXAA Console only.
    // Use zeros or anything else for other platforms.
    // These must be in physical constant registers and NOT immedates.
    // Immedates will result in compiler un-optimizing.
    // {xyzw} = float4(1.0, -1.0, 0.25, -0.25)
    FxaaFloat4 fxaaConsole360ConstDir
) {
/*--------------------------------------------------------------------------*/
    FxaaFloat2 posM;
    posM.x = pos.x;
    posM.y = pos.y;
    #if (FXAA_GATHER4_ALPHA == 1)
        #if (FXAA_DISCARD == 0)
            FxaaFloat4 rgbyM = FxaaTexTop(tex, posM);
            #if (FXAA_GREEN_AS_LUMA == 0)
                #define lumaM rgbyM.w
            #else
                #define lumaM rgbyM.y
            #endif
        #endif
        #if (FXAA_GREEN_AS_LUMA == 0)
            FxaaFloat4 luma4A = FxaaTexAlpha4(tex, posM);
            FxaaFloat4 luma4B = FxaaTexOffAlpha4(tex, posM, FxaaInt2(-1, -1));
        #else
            FxaaFloat4 luma4A = FxaaTexGreen4(tex, posM);
            FxaaFloat4 luma4B = FxaaTexOffGreen4(tex, posM, FxaaInt2(-1, -1));
        #endif
        #if (FXAA_DISCARD == 1)
            #define lumaM luma4A.w
        #endif
        #define lumaE luma4A.z
        #define lumaS luma4A.x
        #define lumaSE luma4A.y
        #define lumaNW luma4B.w
        #define lumaN luma4B.z
        #define lumaW luma4B.x
    #else
        FxaaFloat4 rgbyM = FxaaTexTop(tex, posM);
        #if (FXAA_GREEN_AS_LUMA == 0)
            #define lumaM rgbyM.w
        #else
            #define lumaM rgbyM.y
        #endif
```

```
        FxaaFloat lumaS = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 0, 1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaE = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 1, 0),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaN = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 0,-1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaW = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(-1, 0),
fxaaQualityRcpFrame.xy));
    #endif
/*--------------------------------------------------------------------------*/
    FxaaFloat maxSM = max(lumaS, lumaM);
    FxaaFloat minSM = min(lumaS, lumaM);
    FxaaFloat maxESM = max(lumaE, maxSM);
    FxaaFloat minESM = min(lumaE, minSM);
    FxaaFloat maxWN = max(lumaN, lumaW);
    FxaaFloat minWN = min(lumaN, lumaW);
    FxaaFloat rangeMax = max(maxWN, maxESM);
    FxaaFloat rangeMin = min(minWN, minESM);
    FxaaFloat rangeMaxScaled = rangeMax * fxaaQualityEdgeThreshold;
    FxaaFloat range = rangeMax - rangeMin;
    FxaaFloat rangeMaxClamped = max(fxaaQualityEdgeThresholdMin, rangeMaxScaled);
    FxaaBool earlyExit = range < rangeMaxClamped;
/*--------------------------------------------------------------------------*/
    if(earlyExit)
        #if (FXAA_DISCARD == 1)
            FxaaDiscard;
        #else
            return rgbyM;
        #endif
/*--------------------------------------------------------------------------*/
    #if (FXAA_GATHER4_ALPHA == 0)
        FxaaFloat lumaNW = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(-1,-1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaSE = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 1, 1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaNE = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 1,-1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaSW = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(-1, 1),
fxaaQualityRcpFrame.xy));
    #else
        FxaaFloat lumaNE = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(1, -1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaSW = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(-1, 1),
fxaaQualityRcpFrame.xy));
```

```
    #endif
/*--------------------------------------------------------------------------*/
    FxaaFloat lumaNS = lumaN + lumaS;
    FxaaFloat lumaWE = lumaW + lumaE;
    FxaaFloat subpixRcpRange = 1.0/range;
    FxaaFloat subpixNSWE = lumaNS + lumaWE;
    FxaaFloat edgeHorz1 = (-2.0 * lumaM) + lumaNS;
    FxaaFloat edgeVert1 = (-2.0 * lumaM) + lumaWE;
/*--------------------------------------------------------------------------*/
    FxaaFloat lumaNESE = lumaNE + lumaSE;
    FxaaFloat lumaNWNE = lumaNW + lumaNE;
    FxaaFloat edgeHorz2 = (-2.0 * lumaE) + lumaNESE;
    FxaaFloat edgeVert2 = (-2.0 * lumaN) + lumaNWNE;
/*--------------------------------------------------------------------------*/
    FxaaFloat lumaNWSW = lumaNW + lumaSW;
    FxaaFloat lumaSWSE = lumaSW + lumaSE;
    FxaaFloat edgeHorz4 = (abs(edgeHorz1) * 2.0) + abs(edgeHorz2);
    FxaaFloat edgeVert4 = (abs(edgeVert1) * 2.0) + abs(edgeVert2);
    FxaaFloat edgeHorz3 = (-2.0 * lumaW) + lumaNWSW;
    FxaaFloat edgeVert3 = (-2.0 * lumaS) + lumaSWSE;
    FxaaFloat edgeHorz = abs(edgeHorz3) + edgeHorz4;
    FxaaFloat edgeVert = abs(edgeVert3) + edgeVert4;
/*--------------------------------------------------------------------------*/
    FxaaFloat subpixNWSWNESE = lumaNWSW + lumaNESE;
    FxaaFloat lengthSign = fxaaQualityRcpFrame.x;
    FxaaBool horzSpan = edgeHorz >= edgeVert;
    FxaaFloat subpixA = subpixNSWE * 2.0 + subpixNWSWNESE;
/*--------------------------------------------------------------------------*/
    if(!horzSpan) lumaN = lumaW;
    if(!horzSpan) lumaS = lumaE;
    if(horzSpan) lengthSign = fxaaQualityRcpFrame.y;
    FxaaFloat subpixB = (subpixA * (1.0/12.0)) - lumaM;
/*--------------------------------------------------------------------------*/
    FxaaFloat gradientN = lumaN - lumaM;
    FxaaFloat gradientS = lumaS - lumaM;
    FxaaFloat lumaNN = lumaN + lumaM;
    FxaaFloat lumaSS = lumaS + lumaM;
    FxaaBool pairN = abs(gradientN) >= abs(gradientS);
    FxaaFloat gradient = max(abs(gradientN), abs(gradientS));
    if(pairN) lengthSign = -lengthSign;
    FxaaFloat subpixC = FxaaSat(abs(subpixB) * subpixRcpRange);
/*--------------------------------------------------------------------------*/
    FxaaFloat2 posB;
    posB.x = posM.x;
```

```
    posB.y = posM.y;
    FxaaFloat2 offNP;
    offNP.x = (!horzSpan) ? 0.0 : fxaaQualityRcpFrame.x;
    offNP.y = ( horzSpan) ? 0.0 : fxaaQualityRcpFrame.y;
    if(!horzSpan) posB.x += lengthSign * 0.5;
    if( horzSpan) posB.y += lengthSign * 0.5;
/*--------------------------------------------------------------------------*/
    FxaaFloat2 posN;
    posN.x = posB.x - offNP.x * FXAA_QUALITY__P0;
    posN.y = posB.y - offNP.y * FXAA_QUALITY__P0;
    FxaaFloat2 posP;
    posP.x = posB.x + offNP.x * FXAA_QUALITY__P0;
    posP.y = posB.y + offNP.y * FXAA_QUALITY__P0;
    FxaaFloat subpixD = ((-2.0)*subpixC) + 3.0;
    FxaaFloat lumaEndN = FxaaLuma(FxaaTexTop(tex, posN));
    FxaaFloat subpixE = subpixC * subpixC;
    FxaaFloat lumaEndP = FxaaLuma(FxaaTexTop(tex, posP));
/*--------------------------------------------------------------------------*/
    if(!pairN) lumaNN = lumaSS;
    FxaaFloat gradientScaled = gradient * 1.0/4.0;
    FxaaFloat lumaMM = lumaM - lumaNN * 0.5;
    FxaaFloat subpixF = subpixD * subpixE;
    FxaaBool lumaMLTZero = lumaMM < 0.0;
/*--------------------------------------------------------------------------*/
    lumaEndN -= lumaNN * 0.5;
    lumaEndP -= lumaNN * 0.5;
    FxaaBool doneN = abs(lumaEndN) >= gradientScaled;
    FxaaBool doneP = abs(lumaEndP) >= gradientScaled;
    if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P1;
    if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P1;
    FxaaBool doneNP = (!doneN) || (!doneP);
    if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P1;
    if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P1;
/*--------------------------------------------------------------------------*/
    if(doneNP) {
        if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
        if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
        if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
        if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
        doneN = abs(lumaEndN) >= gradientScaled;
        doneP = abs(lumaEndP) >= gradientScaled;
        if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P2;
        if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P2;
        doneNP = (!doneN) || (!doneP);
```

```
        if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P2;
        if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P2;
/*--------------------------------------------------------------------------*/
        #if (FXAA_QUALITY__PS > 3)
        if(doneNP) {
            if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
            if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
            if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
            if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
            doneN = abs(lumaEndN) >= gradientScaled;
            doneP = abs(lumaEndP) >= gradientScaled;
            if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P3;
            if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P3;
            doneNP = (!doneN) || (!doneP);
            if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P3;
            if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P3;
/*--------------------------------------------------------------------------*/
            #if (FXAA_QUALITY__PS > 4)
            if(doneNP) {
                if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                doneN = abs(lumaEndN) >= gradientScaled;
                doneP = abs(lumaEndP) >= gradientScaled;
                if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P4;
                if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P4;
                doneNP = (!doneN) || (!doneP);
                if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P4;
                if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P4;
/*--------------------------------------------------------------------------*/
                #if (FXAA_QUALITY__PS > 5)
                if(doneNP) {
                    if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                    if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                    if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                    if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                    doneN = abs(lumaEndN) >= gradientScaled;
                    doneP = abs(lumaEndP) >= gradientScaled;
                    if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P5;
                    if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P5;
                    doneNP = (!doneN) || (!doneP);
                    if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P5;
                    if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P5;
```

```
/*--------------------------------------------------------------------------*/
                        #if (FXAA_QUALITY__PS > 6)
                        if(doneNP) {
                            if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                            if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                            if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                            if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                            doneN = abs(lumaEndN) >= gradientScaled;
                            doneP = abs(lumaEndP) >= gradientScaled;
                            if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P6;
                            if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P6;
                            doneNP = (!doneN) || (!doneP);
                            if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P6;
                            if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P6;
/*--------------------------------------------------------------------------*/
                            #if (FXAA_QUALITY__PS > 7)
                            if(doneNP) {
                                if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex,
posN.xy));
                                if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex,
posP.xy));
                                if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                                if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                                doneN = abs(lumaEndN) >= gradientScaled;
                                doneP = abs(lumaEndP) >= gradientScaled;
                                if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P7;
                                if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P7;
                                doneNP = (!doneN) || (!doneP);
                                if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P7;
                                if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P7;
/*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PS > 8)
    if(doneNP) {
        if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
        if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
        if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
        if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
        doneN = abs(lumaEndN) >= gradientScaled;
        doneP = abs(lumaEndP) >= gradientScaled;
        if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P8;
        if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P8;
        doneNP = (!doneN) || (!doneP);
        if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P8;
        if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P8;
```

```
/*--------------------------------------------------------------------------*/
        #if (FXAA_QUALITY__PS > 9)
        if(doneNP) {
            if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
            if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
            if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
            if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
            doneN = abs(lumaEndN) >= gradientScaled;
            doneP = abs(lumaEndP) >= gradientScaled;
            if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P9;
            if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P9;
            doneNP = (!doneN) || (!doneP);
            if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P9;
            if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P9;
/*--------------------------------------------------------------------------*/
            #if (FXAA_QUALITY__PS > 10)
            if(doneNP) {
                if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                doneN = abs(lumaEndN) >= gradientScaled;
                doneP = abs(lumaEndP) >= gradientScaled;
                if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P10;
                if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P10;
                doneNP = (!doneN) || (!doneP);
                if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P10;
                if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P10;
/*--------------------------------------------------------------------------*/
                #if (FXAA_QUALITY__PS > 11)
                if(doneNP) {
                    if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                    if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                    if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                    if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                    doneN = abs(lumaEndN) >= gradientScaled;
                    doneP = abs(lumaEndP) >= gradientScaled;
                    if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P11;
                    if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P11;
                    doneNP = (!doneN) || (!doneP);
                    if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P11;
                    if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P11;
/*--------------------------------------------------------------------------*/
                    #if (FXAA_QUALITY__PS > 12)
```

```
                        if(doneNP) {
                            if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                            if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                            if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                            if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                            doneN = abs(lumaEndN) >= gradientScaled;
                            doneP = abs(lumaEndP) >= gradientScaled;
                            if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P12;
                            if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P12;
                            doneNP = (!doneN) || (!doneP);
                            if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P12;
                            if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P12;
/*--------------------------------------------------------------------------*/
                        }
                        #endif
/*--------------------------------------------------------------------------*/
                    }
                    #endif
/*--------------------------------------------------------------------------*/
                }
                #endif
/*--------------------------------------------------------------------------*/
            }
            #endif
/*--------------------------------------------------------------------------*/
        }
        #endif
/*--------------------------------------------------------------------------*/
    }
    #endif
/*--------------------------------------------------------------------------*/
                            }
                            #endif
/*--------------------------------------------------------------------------*/
                        }
                        #endif
/*--------------------------------------------------------------------------*/
                    }
                    #endif
/*--------------------------------------------------------------------------*/
                }
                #endif
/*--------------------------------------------------------------------------*/
            }
            #endif
/*--------------------------------------------------------------------------*/
        }
        #endif
/*--------------------------------------------------------------------------*/
    }
```

```
/*--------------------------------------------------------------------------*/
    FxaaFloat dstN = posM.x - posN.x;
    FxaaFloat dstP = posP.x - posM.x;
    if(!horzSpan) dstN = posM.y - posN.y;
    if(!horzSpan) dstP = posP.y - posM.y;
/*--------------------------------------------------------------------------*/
    FxaaBool goodSpanN = (lumaEndN < 0.0) != lumaMLTZero;
    FxaaFloat spanLength = (dstP + dstN);
    FxaaBool goodSpanP = (lumaEndP < 0.0) != lumaMLTZero;
    FxaaFloat spanLengthRcp = 1.0/spanLength;
/*--------------------------------------------------------------------------*/
    FxaaBool directionN = dstN < dstP;
    FxaaFloat dst = min(dstN, dstP);
    FxaaBool goodSpan = directionN ? goodSpanN : goodSpanP;
    FxaaFloat subpixG = subpixF * subpixF;
    FxaaFloat pixelOffset = (dst * (-spanLengthRcp)) + 0.5;
    FxaaFloat subpixH = subpixG * fxaaQualitySubpix;
/*--------------------------------------------------------------------------*/
    FxaaFloat pixelOffsetGood = goodSpan ? pixelOffset : 0.0;
    FxaaFloat pixelOffsetSubpix = max(pixelOffsetGood, subpixH);
    if(!horzSpan) posM.x += pixelOffsetSubpix * lengthSign;
    if( horzSpan) posM.y += pixelOffsetSubpix * lengthSign;
    #if (FXAA_DISCARD == 1)
        return FxaaTexTop(tex, posM);
    #else
        return FxaaFloat4(FxaaTexTop(tex, posM).xyz, lumaM);
    #endif
}
/*==========================================================================
================*/
#endif




//--------------------------------------------------------------------------
-----
// File:              es3-kepler\FXAA\assets\shaders/FXAA_Default.frag
// SDK Version: v2.0
// Email:             gameworks@nvidia.com
// Site:              http://developer.nvidia.com/
//
// Copyright (c) 2014, NVIDIA CORPORATION. All rights reserved.
//
```

```
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions
// are met:
//  * Redistributions of source code must retain the above copyright
//    notice, this list of conditions and the following disclaimer.
//  * Redistributions in binary form must reproduce the above copyright
//    notice, this list of conditions and the following disclaimer in the
//    documentation and/or other materials provided with the distribution.
//  * Neither the name of NVIDIA CORPORATION nor the names of its
//    contributors may be used to endorse or promote products derived
//    from this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS ``AS IS'' AND ANY
// EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
// PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
// CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
// EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
// PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
// PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
// OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
//------------------------------------------------------------------------
-----
//#version 100

precision highp float;

uniform sampler2D uSourceTex;
uniform vec2 RCPFrame;
varying vec2 vTexCoord;

void main(void)
{
    gl_FragColor = FxaaPixelShader(vTexCoord,
                    FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f),     // FxaaFloat4 fxaaConsolePosPos,
        uSourceTex,                                 // FxaaTex tex,
        uSourceTex,                                 // FxaaTex
fxaaConsole360TexExpBiasNegOne,
        uSourceTex,                                 // FxaaTex
fxaaConsole360TexExpBiasNegTwo,
        RCPFrame,                                   // FxaaFloat2 fxaaQualityRcpFrame,
```

```
            FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f),         // FxaaFloat4 fxaaConsoleRcpFrameOpt,
            FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f),         // FxaaFloat4 fxaaConsoleRcpFrameOpt2,
            FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f),         // FxaaFloat4 fxaaConsole360RcpFrameOpt2,
            0.75f,                                              // FxaaFloat fxaaQualitySubpix,
            0.166f,                                             // FxaaFloat
fxaaQualityEdgeThreshold,
            0.0833f,                                    // FxaaFloat
fxaaQualityEdgeThresholdMin,
            0.0f,                                               // FxaaFloat fxaaConsoleEdgeSharpness,
            0.0f,                                               // FxaaFloat fxaaConsoleEdgeThreshold,
            0.0f,                                               // FxaaFloat
fxaaConsoleEdgeThresholdMin,
            FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f)          // FxaaFloat fxaaConsole360ConstDir,
      );
}
//////////////////////////////////////FXAA_Extreme_Quality.frag//////////////////////////
#define FXAA_PC 1
#define FXAA_GLSL_130 1
#define FXAA_QUALITY__PRESET 39

#define FXAA_GREEN_AS_LUMA 1


/*--------------------------------------------------------------------------*/
#ifndef FXAA_PC_CONSOLE
      //
      // The console algorithm for PC is included
      // for developers targeting really low spec machines.
      // Likely better to just run FXAA_PC, and use a really low preset.
      //
      #define FXAA_PC_CONSOLE 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_GLSL_120
      #define FXAA_GLSL_120 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_GLSL_130
      #define FXAA_GLSL_130 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_HLSL_3
      #define FXAA_HLSL_3 0
#endif
```

```
/*--------------------------------------------------------------------------*/
#ifndef FXAA_HLSL_4
    #define FXAA_HLSL_4 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_HLSL_5
    #define FXAA_HLSL_5 0
#endif
/*==========================================================================
================*/
#ifndef FXAA_GREEN_AS_LUMA
    //
    // For those using non-linear color,
    // and either not able to get luma in alpha, or not wanting to,
    // this enables FXAA to run using green as a proxy for luma.
    // So with this enabled, no need to pack luma in alpha.
    //
    // This will turn off AA on anything which lacks some amount of green.
    // Pure red and blue or combination of only R and B, will get no AA.
    //
    // Might want to lower the settings for both,
    //      fxaaConsoleEdgeThresholdMin
    //      fxaaQualityEdgeThresholdMin
    // In order to insure AA does not get turned off on colors
    // which contain a minor amount of green.
    //
    // 1 = On.
    // 0 = Off.
    //
    #define FXAA_GREEN_AS_LUMA 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_EARLY_EXIT
    //
    // Controls algorithm's early exit path.
    // On PS3 turning this ON adds 2 cycles to the shader.
    // On 360 turning this OFF adds 10ths of a millisecond to the shader.
    // Turning this off on console will result in a more blurry image.
    // So this defaults to on.
    //
    // 1 = On.
    // 0 = Off.
    //
    #define FXAA_EARLY_EXIT 1
```

```
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_DISCARD
    //
    // Only valid for PC OpenGL currently.
    // Probably will not work when FXAA_GREEN_AS_LUMA = 1.
    //
    // 1 = Use discard on pixels which don't need AA.
    //     For APIs which enable concurrent TEX+ROP from same surface.
    // 0 = Return unchanged color on pixels which don't need AA.
    //
    #define FXAA_DISCARD 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_FAST_PIXEL_OFFSET
    //
    // Used for GLSL 120 only.
    //
    // 1 = GL API supports fast pixel offsets
    // 0 = do not use fast pixel offsets
    //
    #ifdef GL_EXT_gpu_shader4
        #define FXAA_FAST_PIXEL_OFFSET 1
    #endif
    #ifdef GL_NV_gpu_shader5
        #define FXAA_FAST_PIXEL_OFFSET 1
    #endif
    #ifdef GL_ARB_gpu_shader5
        #define FXAA_FAST_PIXEL_OFFSET 1
    #endif
    #ifndef FXAA_FAST_PIXEL_OFFSET
        #define FXAA_FAST_PIXEL_OFFSET 0
    #endif
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_GATHER4_ALPHA
    //
    // 1 = API supports gather4 on alpha channel.
    // 0 = API does not support gather4 on alpha channel.
    //
    #if (FXAA_HLSL_5 == 1)
        #define FXAA_GATHER4_ALPHA 1
    #endif
    #ifdef GL_ARB_gpu_shader5
```

```
        #define FXAA_GATHER4_ALPHA 1
    #endif
    #ifdef GL_NV_gpu_shader5
        #define FXAA_GATHER4_ALPHA 1
    #endif
    #ifndef FXAA_GATHER4_ALPHA
        #define FXAA_GATHER4_ALPHA 0
    #endif
#endif


/*============================================================
=================
                        FXAA QUALITY - TUNING KNOBS
------------------------------------------------------------------------
NOTE the other tuning knobs are now in the shader function inputs!
============================================================
=================*/
#ifndef FXAA_QUALITY__PRESET
    //
    // Choose the quality preset.
    // This needs to be compiled into the shader as it effects code.
    // Best option to include multiple presets is to
    // in each shader define the preset, then include this file.
    //
    // OPTIONS
    // -------------------------------------------------------------------
    // 10 to 15 - default medium dither (10=fastest, 15=highest quality)
    // 20 to 29 - less dither, more expensive (20=fastest, 29=highest quality)
    // 39       - no dither, very expensive
    //
    // NOTES
    // -------------------------------------------------------------------
    // 12 = slightly faster then FXAA 3.9 and higher edge quality (default)
    // 13 = about same speed as FXAA 3.9 and better than 12
    // 23 = closest to FXAA 3.9 visually and performance wise
    //  _ = the lowest digit is directly related to performance
    // _  = the highest digit is directly related to style
    //
    #define FXAA_QUALITY__PRESET 12
#endif


/*============================================================
```

```
==================

                                FXAA QUALITY - PRESETS

============================================================
==================*/

/*============================================================
==================
                        FXAA QUALITY - MEDIUM DITHER PRESETS
============================================================
==================*/
#if (FXAA_QUALITY__PRESET == 10)
    #define FXAA_QUALITY__PS 3
    #define FXAA_QUALITY__P0 1.5
    #define FXAA_QUALITY__P1 3.0
    #define FXAA_QUALITY__P2 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 11)
    #define FXAA_QUALITY__PS 4
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 3.0
    #define FXAA_QUALITY__P3 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 12)
    #define FXAA_QUALITY__PS 5
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 4.0
    #define FXAA_QUALITY__P4 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 13)
    #define FXAA_QUALITY__PS 6
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 4.0
    #define FXAA_QUALITY__P5 12.0
```

```
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 14)
    #define FXAA_QUALITY__PS 7
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 4.0
    #define FXAA_QUALITY__P6 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 15)
    #define FXAA_QUALITY__PS 8
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 2.0
    #define FXAA_QUALITY__P6 4.0
    #define FXAA_QUALITY__P7 12.0
#endif

/*============================================================================
==================
                        FXAA QUALITY - LOW DITHER PRESETS
============================================================================
==================*/
#if (FXAA_QUALITY__PRESET == 20)
    #define FXAA_QUALITY__PS 3
    #define FXAA_QUALITY__P0 1.5
    #define FXAA_QUALITY__P1 2.0
    #define FXAA_QUALITY__P2 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 21)
    #define FXAA_QUALITY__PS 4
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 8.0
#endif
```

```
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 22)
    #define FXAA_QUALITY__PS 5
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 23)
    #define FXAA_QUALITY__PS 6
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 24)
    #define FXAA_QUALITY__PS 7
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 3.0
    #define FXAA_QUALITY__P6 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 25)
    #define FXAA_QUALITY__PS 8
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 2.0
    #define FXAA_QUALITY__P6 4.0
    #define FXAA_QUALITY__P7 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 26)
```

```
        #define FXAA_QUALITY__PS 9
        #define FXAA_QUALITY__P0 1.0
        #define FXAA_QUALITY__P1 1.5
        #define FXAA_QUALITY__P2 2.0
        #define FXAA_QUALITY__P3 2.0
        #define FXAA_QUALITY__P4 2.0
        #define FXAA_QUALITY__P5 2.0
        #define FXAA_QUALITY__P6 2.0
        #define FXAA_QUALITY__P7 4.0
        #define FXAA_QUALITY__P8 8.0
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PRESET == 27)
        #define FXAA_QUALITY__PS 10
        #define FXAA_QUALITY__P0 1.0
        #define FXAA_QUALITY__P1 1.5
        #define FXAA_QUALITY__P2 2.0
        #define FXAA_QUALITY__P3 2.0
        #define FXAA_QUALITY__P4 2.0
        #define FXAA_QUALITY__P5 2.0
        #define FXAA_QUALITY__P6 2.0
        #define FXAA_QUALITY__P7 2.0
        #define FXAA_QUALITY__P8 4.0
        #define FXAA_QUALITY__P9 8.0
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PRESET == 28)
        #define FXAA_QUALITY__PS 11
        #define FXAA_QUALITY__P0 1.0
        #define FXAA_QUALITY__P1 1.5
        #define FXAA_QUALITY__P2 2.0
        #define FXAA_QUALITY__P3 2.0
        #define FXAA_QUALITY__P4 2.0
        #define FXAA_QUALITY__P5 2.0
        #define FXAA_QUALITY__P6 2.0
        #define FXAA_QUALITY__P7 2.0
        #define FXAA_QUALITY__P8 2.0
        #define FXAA_QUALITY__P9 4.0
        #define FXAA_QUALITY__P10 8.0
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PRESET == 29)
        #define FXAA_QUALITY__PS 12
        #define FXAA_QUALITY__P0 1.0
```

```
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 2.0
    #define FXAA_QUALITY__P6 2.0
    #define FXAA_QUALITY__P7 2.0
    #define FXAA_QUALITY__P8 2.0
    #define FXAA_QUALITY__P9 2.0
    #define FXAA_QUALITY__P10 4.0
    #define FXAA_QUALITY__P11 8.0
#endif

/*============================================================
=================
                    FXAA QUALITY - EXTREME QUALITY
============================================================
=================*/
#if (FXAA_QUALITY__PRESET == 39)
    #define FXAA_QUALITY__PS 12
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.0
    #define FXAA_QUALITY__P2 1.0
    #define FXAA_QUALITY__P3 1.0
    #define FXAA_QUALITY__P4 1.0
    #define FXAA_QUALITY__P5 1.5
    #define FXAA_QUALITY__P6 2.0
    #define FXAA_QUALITY__P7 2.0
    #define FXAA_QUALITY__P8 2.0
    #define FXAA_QUALITY__P9 2.0
    #define FXAA_QUALITY__P10 4.0
    #define FXAA_QUALITY__P11 8.0
#endif



/*============================================================
=================

                    API PORTING

============================================================
=================*/
#if (FXAA_GLSL_120 == 1) || (FXAA_GLSL_130 == 1)
```

```
    #define FxaaBool bool
    #define FxaaDiscard discard
    #define FxaaFloat float
    #define FxaaFloat2 vec2
    #define FxaaFloat3 vec3
    #define FxaaFloat4 vec4
    #define FxaaHalf float
    #define FxaaHalf2 vec2
    #define FxaaHalf3 vec3
    #define FxaaHalf4 vec4
    #define FxaaInt2 ivec2
    #define FxaaSat(x) clamp(x, 0.0, 1.0)
    #define FxaaTex sampler2D
#else
    #define FxaaBool bool
    #define FxaaDiscard clip(-1)
    #define FxaaFloat float
    #define FxaaFloat2 float2
    #define FxaaFloat3 float3
    #define FxaaFloat4 float4
    #define FxaaHalf half
    #define FxaaHalf2 half2
    #define FxaaHalf3 half3
    #define FxaaHalf4 half4
    #define FxaaSat(x) saturate(x)
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_GLSL_120 == 1)
    // Requires,
    //  #version 120
    // And at least,
    //  #extension GL_EXT_gpu_shader4 : enable
    //  (or set FXAA_FAST_PIXEL_OFFSET 1 to work like DX9)
    #define FxaaTexTop(t, p) texture2DLod(t, p, 0.0)
    #if (FXAA_FAST_PIXEL_OFFSET == 1)
        #define FxaaTexOff(t, p, o, r) texture2DLodOffset(t, p, 0.0, o)
    #else
        #define FxaaTexOff(t, p, o, r) texture2DLod(t, p + (o * r), 0.0)
    #endif
    #if (FXAA_GATHER4_ALPHA == 1)
        // use #extension GL_ARB_gpu_shader5 : enable
        #define FxaaTexAlpha4(t, p) textureGather(t, p, 3)
        #define FxaaTexOffAlpha4(t, p, o) textureGatherOffset(t, p, o, 3)
        #define FxaaTexGreen4(t, p) textureGather(t, p, 1)
```

```
            #define FxaaTexOffGreen4(t, p, o) textureGatherOffset(t, p, o, 1)
        #endif
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_GLSL_130 == 1)
        // Requires "#version 130" or better
        #define FxaaTexTop(t, p) textureLod(t, p, 0.0)
        #define FxaaTexOff(t, p, o, r) textureLodOffset(t, p, 0.0, o)
        #if (FXAA_GATHER4_ALPHA == 1)
            // use #extension GL_ARB_gpu_shader5 : enable
            #define FxaaTexAlpha4(t, p) textureGather(t, p, 3)
            #define FxaaTexOffAlpha4(t, p, o) textureGatherOffset(t, p, o, 3)
            #define FxaaTexGreen4(t, p) textureGather(t, p, 1)
            #define FxaaTexOffGreen4(t, p, o) textureGatherOffset(t, p, o, 1)
        #endif
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_HLSL_3 == 1) || (FXAA_360 == 1) || (FXAA_PS3 == 1)
        #define FxaaInt2 float2
        #define FxaaTex sampler2D
        #define FxaaTexTop(t, p) tex2Dlod(t, float4(p, 0.0, 0.0))
        #define FxaaTexOff(t, p, o, r) tex2Dlod(t, float4(p + (o * r), 0, 0))
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_HLSL_4 == 1)
        #define FxaaInt2 int2
        struct FxaaTex { SamplerState smpl; Texture2D tex; };
        #define FxaaTexTop(t, p) t.tex.SampleLevel(t.smpl, p, 0.0)
        #define FxaaTexOff(t, p, o, r) t.tex.SampleLevel(t.smpl, p, 0.0, o)
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_HLSL_5 == 1)
        #define FxaaInt2 int2
        struct FxaaTex { SamplerState smpl; Texture2D tex; };
        #define FxaaTexTop(t, p) t.tex.SampleLevel(t.smpl, p, 0.0)
        #define FxaaTexOff(t, p, o, r) t.tex.SampleLevel(t.smpl, p, 0.0, o)
        #define FxaaTexAlpha4(t, p) t.tex.GatherAlpha(t.smpl, p)
        #define FxaaTexOffAlpha4(t, p, o) t.tex.GatherAlpha(t.smpl, p, o)
        #define FxaaTexGreen4(t, p) t.tex.GatherGreen(t.smpl, p)
        #define FxaaTexOffGreen4(t, p, o) t.tex.GatherGreen(t.smpl, p, o)
    #endif


    /*============================================================
```

```
==================
                        GREEN AS LUMA OPTION SUPPORT FUNCTION
============================================================
==================*/
#if (FXAA_GREEN_AS_LUMA == 0)
    FxaaFloat FxaaLuma(FxaaFloat4 rgba) { return rgba.w; }
#else
    FxaaFloat FxaaLuma(FxaaFloat4 rgba) { return rgba.y; }
#endif



/*============================================================
==================

                        FXAA3 QUALITY - PC

============================================================
==================*/
#if (FXAA_PC == 1)
/*--------------------------------------------------------------------------*/
FxaaFloat4 FxaaPixelShader(
    //
    // Use noperspective interpolation here (turn off perspective interpolation).
    // {xy} = center of pixel
    FxaaFloat2 pos,
    //
    // Used only for FXAA Console, and not used on the 360 version.
    // Use noperspective interpolation here (turn off perspective interpolation).
    // {xy__} = upper left of pixel
    // {__zw} = lower right of pixel
    FxaaFloat4 fxaaConsolePosPos,
    //
    // Input color texture.
    // {rgb_} = color in linear or perceptual color space
    // if (FXAA_GREEN_AS_LUMA == 0)
    //     {___a} = luma in perceptual color space (not linear)
    FxaaTex tex,
    //
    // Only used on the optimized 360 version of FXAA Console.
    // For everything but 360, just use the same input here as for "tex".
    // For 360, same texture, just alias with a 2nd sampler.
    // This sampler needs to have an exponent bias of -1.
```

```
FxaaTex fxaaConsole360TexExpBiasNegOne,
//
// Only used on the optimized 360 version of FXAA Console.
// For everything but 360, just use the same input here as for "tex".
// For 360, same texture, just alias with a 3nd sampler.
// This sampler needs to have an exponent bias of -2.
FxaaTex fxaaConsole360TexExpBiasNegTwo,
//
// Only used on FXAA Quality.
// This must be from a constant/uniform.
// {x_} = 1.0/screenWidthInPixels
// {_y} = 1.0/screenHeightInPixels
FxaaFloat2 fxaaQualityRcpFrame,
//
// Only used on FXAA Console.
// This must be from a constant/uniform.
// This effects sub-pixel AA quality and inversely sharpness.
//     Where N ranges between,
//         N = 0.50 (default)
//         N = 0.33 (sharper)
// {x___} = -N/screenWidthInPixels
// {_y__} = -N/screenHeightInPixels
// {__z_} =  N/screenWidthInPixels
// {___w} =  N/screenHeightInPixels
FxaaFloat4 fxaaConsoleRcpFrameOpt,
//
// Only used on FXAA Console.
// Not used on 360, but used on PS3 and PC.
// This must be from a constant/uniform.
// {x___} = -2.0/screenWidthInPixels
// {_y__} = -2.0/screenHeightInPixels
// {__z_} =  2.0/screenWidthInPixels
// {___w} =  2.0/screenHeightInPixels
FxaaFloat4 fxaaConsoleRcpFrameOpt2,
//
// Only used on FXAA Console.
// Only used on 360 in place of fxaaConsoleRcpFrameOpt2.
// This must be from a constant/uniform.
// {x___} =  8.0/screenWidthInPixels
// {_y__} =  8.0/screenHeightInPixels
// {__z_} = -4.0/screenWidthInPixels
// {___w} = -4.0/screenHeightInPixels
FxaaFloat4 fxaaConsole360RcpFrameOpt2,
//
```

```
// Only used on FXAA Quality.
// This used to be the FXAA_QUALITY__SUBPIX define.
// It is here now to allow easier tuning.
// Choose the amount of sub-pixel aliasing removal.
// This can effect sharpness.
//     1.00 - upper limit (softer)
//     0.75 - default amount of filtering
//     0.50 - lower limit (sharper, less sub-pixel aliasing removal)
//     0.25 - almost off
//     0.00 - completely off
FxaaFloat fxaaQualitySubpix,
//
// Only used on FXAA Quality.
// This used to be the FXAA_QUALITY__EDGE_THRESHOLD define.
// It is here now to allow easier tuning.
// The minimum amount of local contrast required to apply algorithm.
//     0.333 - too little (faster)
//     0.250 - low quality
//     0.166 - default
//     0.125 - high quality
//     0.063 - overkill (slower)
FxaaFloat fxaaQualityEdgeThreshold,
//
// Only used on FXAA Quality.
// This used to be the FXAA_QUALITY__EDGE_THRESHOLD_MIN define.
// It is here now to allow easier tuning.
// Trims the algorithm from processing darks.
//     0.0833 - upper limit (default, the start of visible unfiltered edges)
//     0.0625 - high quality (faster)
//     0.0312 - visible limit (slower)
// Special notes when using FXAA_GREEN_AS_LUMA,
//     Likely want to set this to zero.
//     As colors that are mostly not-green
//     will appear very dark in the green channel!
//     Tune by looking at mostly non-green content,
//     then start at zero and increase until aliasing is a problem.
FxaaFloat fxaaQualityEdgeThresholdMin,
//
// Only used on FXAA Console.
// This used to be the FXAA_CONSOLE__EDGE_SHARPNESS define.
// It is here now to allow easier tuning.
// This does not effect PS3, as this needs to be compiled in.
//     Use FXAA_CONSOLE__PS3_EDGE_SHARPNESS for PS3.
//     Due to the PS3 being ALU bound,
```

```
//     there are only three safe values here: 2 and 4 and 8.
//     These options use the shaders ability to a free *|/ by 2|4|8.
// For all other platforms can be a non-power of two.
//     8.0 is sharper (default!!!)
//     4.0 is softer
//     2.0 is really soft (good only for vector graphics inputs)
FxaaFloat fxaaConsoleEdgeSharpness,
//
// Only used on FXAA Console.
// This used to be the FXAA_CONSOLE__EDGE_THRESHOLD define.
// It is here now to allow easier tuning.
// This does not effect PS3, as this needs to be compiled in.
//     Use FXAA_CONSOLE__PS3_EDGE_THRESHOLD for PS3.
//     Due to the PS3 being ALU bound,
//     there are only two safe values here: 1/4 and 1/8.
//     These options use the shaders ability to a free *|/ by 2|4|8.
// The console setting has a different mapping than the quality setting.
// Other platforms can use other values.
//     0.125 leaves less aliasing, but is softer (default!!!)
//     0.25 leaves more aliasing, and is sharper
FxaaFloat fxaaConsoleEdgeThreshold,
//
// Only used on FXAA Console.
// This used to be the FXAA_CONSOLE__EDGE_THRESHOLD_MIN define.
// It is here now to allow easier tuning.
// Trims the algorithm from processing darks.
// The console setting has a different mapping than the quality setting.
// This only applies when FXAA_EARLY_EXIT is 1.
// This does not apply to PS3,
// PS3 was simplified to avoid more shader instructions.
//     0.06 - faster but more aliasing in darks
//     0.05 - default
//     0.04 - slower and less aliasing in darks
// Special notes when using FXAA_GREEN_AS_LUMA,
//     Likely want to set this to zero.
//     As colors that are mostly not-green
//     will appear very dark in the green channel!
//     Tune by looking at mostly non-green content,
//     then start at zero and increase until aliasing is a problem.
FxaaFloat fxaaConsoleEdgeThresholdMin,
//
// Extra constants for 360 FXAA Console only.
// Use zeros or anything else for other platforms.
// These must be in physical constant registers and NOT immedates.
```

```
    // Immedates will result in compiler un-optimizing.
    // {xyzw} = float4(1.0, -1.0, 0.25, -0.25)
    FxaaFloat4 fxaaConsole360ConstDir
) {
/*--------------------------------------------------------------------------*/
    FxaaFloat2 posM;
    posM.x = pos.x;
    posM.y = pos.y;
    #if (FXAA_GATHER4_ALPHA == 1)
        #if (FXAA_DISCARD == 0)
            FxaaFloat4 rgbyM = FxaaTexTop(tex, posM);
            #if (FXAA_GREEN_AS_LUMA == 0)
                #define lumaM rgbyM.w
            #else
                #define lumaM rgbyM.y
            #endif
        #endif
        #if (FXAA_GREEN_AS_LUMA == 0)
            FxaaFloat4 luma4A = FxaaTexAlpha4(tex, posM);
            FxaaFloat4 luma4B = FxaaTexOffAlpha4(tex, posM, FxaaInt2(-1, -1));
        #else
            FxaaFloat4 luma4A = FxaaTexGreen4(tex, posM);
            FxaaFloat4 luma4B = FxaaTexOffGreen4(tex, posM, FxaaInt2(-1, -1));
        #endif
        #if (FXAA_DISCARD == 1)
            #define lumaM luma4A.w
        #endif
        #define lumaE luma4A.z
        #define lumaS luma4A.x
        #define lumaSE luma4A.y
        #define lumaNW luma4B.w
        #define lumaN luma4B.z
        #define lumaW luma4B.x
    #else
        FxaaFloat4 rgbyM = FxaaTexTop(tex, posM);
        #if (FXAA_GREEN_AS_LUMA == 0)
            #define lumaM rgbyM.w
        #else
            #define lumaM rgbyM.y
        #endif
        FxaaFloat lumaS = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 0, 1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaE = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 1, 0),
fxaaQualityRcpFrame.xy));
```

```
        FxaaFloat lumaN = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 0,-1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaW = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(-1, 0),
fxaaQualityRcpFrame.xy));
    #endif
/*--------------------------------------------------------------------------*/
    FxaaFloat maxSM = max(lumaS, lumaM);
    FxaaFloat minSM = min(lumaS, lumaM);
    FxaaFloat maxESM = max(lumaE, maxSM);
    FxaaFloat minESM = min(lumaE, minSM);
    FxaaFloat maxWN = max(lumaN, lumaW);
    FxaaFloat minWN = min(lumaN, lumaW);
    FxaaFloat rangeMax = max(maxWN, maxESM);
    FxaaFloat rangeMin = min(minWN, minESM);
    FxaaFloat rangeMaxScaled = rangeMax * fxaaQualityEdgeThreshold;
    FxaaFloat range = rangeMax - rangeMin;
    FxaaFloat rangeMaxClamped = max(fxaaQualityEdgeThresholdMin, rangeMaxScaled);
    FxaaBool earlyExit = range < rangeMaxClamped;
/*--------------------------------------------------------------------------*/
    if(earlyExit)
        #if (FXAA_DISCARD == 1)
            FxaaDiscard;
        #else
            return rgbyM;
        #endif
/*--------------------------------------------------------------------------*/
    #if (FXAA_GATHER4_ALPHA == 0)
        FxaaFloat lumaNW = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(-1,-1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaSE = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 1, 1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaNE = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 1,-1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaSW = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(-1, 1),
fxaaQualityRcpFrame.xy));
    #else
        FxaaFloat lumaNE = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(1, -1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaSW = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(-1, 1),
fxaaQualityRcpFrame.xy));
    #endif
/*--------------------------------------------------------------------------*/
    FxaaFloat lumaNS = lumaN + lumaS;
    FxaaFloat lumaWE = lumaW + lumaE;
```

```
    FxaaFloat subpixRcpRange = 1.0/range;
    FxaaFloat subpixNSWE = lumaNS + lumaWE;
    FxaaFloat edgeHorz1 = (-2.0 * lumaM) + lumaNS;
    FxaaFloat edgeVert1 = (-2.0 * lumaM) + lumaWE;
/*--------------------------------------------------------------------------*/
    FxaaFloat lumaNESE = lumaNE + lumaSE;
    FxaaFloat lumaNWNE = lumaNW + lumaNE;
    FxaaFloat edgeHorz2 = (-2.0 * lumaE) + lumaNESE;
    FxaaFloat edgeVert2 = (-2.0 * lumaN) + lumaNWNE;
/*--------------------------------------------------------------------------*/
    FxaaFloat lumaNWSW = lumaNW + lumaSW;
    FxaaFloat lumaSWSE = lumaSW + lumaSE;
    FxaaFloat edgeHorz4 = (abs(edgeHorz1) * 2.0) + abs(edgeHorz2);
    FxaaFloat edgeVert4 = (abs(edgeVert1) * 2.0) + abs(edgeVert2);
    FxaaFloat edgeHorz3 = (-2.0 * lumaW) + lumaNWSW;
    FxaaFloat edgeVert3 = (-2.0 * lumaS) + lumaSWSE;
    FxaaFloat edgeHorz = abs(edgeHorz3) + edgeHorz4;
    FxaaFloat edgeVert = abs(edgeVert3) + edgeVert4;
/*--------------------------------------------------------------------------*/
    FxaaFloat subpixNWSWNESE = lumaNWSW + lumaNESE;
    FxaaFloat lengthSign = fxaaQualityRcpFrame.x;
    FxaaBool horzSpan = edgeHorz >= edgeVert;
    FxaaFloat subpixA = subpixNSWE * 2.0 + subpixNWSWNESE;
/*--------------------------------------------------------------------------*/
    if(!horzSpan) lumaN = lumaW;
    if(!horzSpan) lumaS = lumaE;
    if(horzSpan) lengthSign = fxaaQualityRcpFrame.y;
    FxaaFloat subpixB = (subpixA * (1.0/12.0)) - lumaM;
/*--------------------------------------------------------------------------*/
    FxaaFloat gradientN = lumaN - lumaM;
    FxaaFloat gradientS = lumaS - lumaM;
    FxaaFloat lumaNN = lumaN + lumaM;
    FxaaFloat lumaSS = lumaS + lumaM;
    FxaaBool pairN = abs(gradientN) >= abs(gradientS);
    FxaaFloat gradient = max(abs(gradientN), abs(gradientS));
    if(pairN) lengthSign = -lengthSign;
    FxaaFloat subpixC = FxaaSat(abs(subpixB) * subpixRcpRange);
/*--------------------------------------------------------------------------*/
    FxaaFloat2 posB;
    posB.x = posM.x;
    posB.y = posM.y;
    FxaaFloat2 offNP;
    offNP.x = (!horzSpan) ? 0.0 : fxaaQualityRcpFrame.x;
    offNP.y = ( horzSpan) ? 0.0 : fxaaQualityRcpFrame.y;
```

```
    if(!horzSpan) posB.x += lengthSign * 0.5;
    if( horzSpan) posB.y += lengthSign * 0.5;
/*--------------------------------------------------------------------------*/
    FxaaFloat2 posN;
    posN.x = posB.x - offNP.x * FXAA_QUALITY__P0;
    posN.y = posB.y - offNP.y * FXAA_QUALITY__P0;
    FxaaFloat2 posP;
    posP.x = posB.x + offNP.x * FXAA_QUALITY__P0;
    posP.y = posB.y + offNP.y * FXAA_QUALITY__P0;
    FxaaFloat subpixD = ((-2.0)*subpixC) + 3.0;
    FxaaFloat lumaEndN = FxaaLuma(FxaaTexTop(tex, posN));
    FxaaFloat subpixE = subpixC * subpixC;
    FxaaFloat lumaEndP = FxaaLuma(FxaaTexTop(tex, posP));
/*--------------------------------------------------------------------------*/
    if(!pairN) lumaNN = lumaSS;
    FxaaFloat gradientScaled = gradient * 1.0/4.0;
    FxaaFloat lumaMM = lumaM - lumaNN * 0.5;
    FxaaFloat subpixF = subpixD * subpixE;
    FxaaBool lumaMLTZero = lumaMM < 0.0;
/*--------------------------------------------------------------------------*/
    lumaEndN -= lumaNN * 0.5;
    lumaEndP -= lumaNN * 0.5;
    FxaaBool doneN = abs(lumaEndN) >= gradientScaled;
    FxaaBool doneP = abs(lumaEndP) >= gradientScaled;
    if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P1;
    if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P1;
    FxaaBool doneNP = (!doneN) || (!doneP);
    if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P1;
    if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P1;
/*--------------------------------------------------------------------------*/
    if(doneNP) {
        if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
        if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
        if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
        if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
        doneN = abs(lumaEndN) >= gradientScaled;
        doneP = abs(lumaEndP) >= gradientScaled;
        if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P2;
        if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P2;
        doneNP = (!doneN) || (!doneP);
        if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P2;
        if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P2;
/*--------------------------------------------------------------------------*/
        #if (FXAA_QUALITY__PS > 3)
```

```
        if(doneNP) {
            if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
            if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
            if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
            if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
            doneN = abs(lumaEndN) >= gradientScaled;
            doneP = abs(lumaEndP) >= gradientScaled;
            if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P3;
            if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P3;
            doneNP = (!doneN) || (!doneP);
            if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P3;
            if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P3;
/*--------------------------------------------------------------------------*/
            #if (FXAA_QUALITY__PS > 4)
            if(doneNP) {
                if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                doneN = abs(lumaEndN) >= gradientScaled;
                doneP = abs(lumaEndP) >= gradientScaled;
                if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P4;
                if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P4;
                doneNP = (!doneN) || (!doneP);
                if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P4;
                if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P4;
/*--------------------------------------------------------------------------*/
                #if (FXAA_QUALITY__PS > 5)
                if(doneNP) {
                    if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                    if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                    if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                    if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                    doneN = abs(lumaEndN) >= gradientScaled;
                    doneP = abs(lumaEndP) >= gradientScaled;
                    if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P5;
                    if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P5;
                    doneNP = (!doneN) || (!doneP);
                    if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P5;
                    if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P5;
/*--------------------------------------------------------------------------*/
                    #if (FXAA_QUALITY__PS > 6)
                    if(doneNP) {
                        if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
```

```
                            if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                            if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                            if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                            doneN = abs(lumaEndN) >= gradientScaled;
                            doneP = abs(lumaEndP) >= gradientScaled;
                            if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P6;
                            if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P6;
                            doneNP = (!doneN) || (!doneP);
                            if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P6;
                            if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P6;
/*--------------------------------------------------------------------------*/
                            #if (FXAA_QUALITY__PS > 7)
                            if(doneNP) {
                                if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex,
posN.xy));
                                if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex,
posP.xy));
                                if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                                if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                                doneN = abs(lumaEndN) >= gradientScaled;
                                doneP = abs(lumaEndP) >= gradientScaled;
                                if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P7;
                                if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P7;
                                doneNP = (!doneN) || (!doneP);
                                if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P7;
                                if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P7;
/*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PS > 8)
    if(doneNP) {
        if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
        if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
        if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
        if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
        doneN = abs(lumaEndN) >= gradientScaled;
        doneP = abs(lumaEndP) >= gradientScaled;
        if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P8;
        if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P8;
        doneNP = (!doneN) || (!doneP);
        if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P8;
        if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P8;
/*--------------------------------------------------------------------------*/
        #if (FXAA_QUALITY__PS > 9)
        if(doneNP) {
            if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
```

```
    if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
    if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
    if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
    doneN = abs(lumaEndN) >= gradientScaled;
    doneP = abs(lumaEndP) >= gradientScaled;
    if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P9;
    if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P9;
    doneNP = (!doneN) || (!doneP);
    if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P9;
    if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P9;
/*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PS > 10)
    if(doneNP) {
        if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
        if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
        if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
        if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
        doneN = abs(lumaEndN) >= gradientScaled;
        doneP = abs(lumaEndP) >= gradientScaled;
        if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P10;
        if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P10;
        doneNP = (!doneN) || (!doneP);
        if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P10;
        if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P10;
/*--------------------------------------------------------------------------*/
        #if (FXAA_QUALITY__PS > 11)
        if(doneNP) {
            if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
            if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
            if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
            if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
            doneN = abs(lumaEndN) >= gradientScaled;
            doneP = abs(lumaEndP) >= gradientScaled;
            if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P11;
            if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P11;
            doneNP = (!doneN) || (!doneP);
            if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P11;
            if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P11;
/*--------------------------------------------------------------------------*/
            #if (FXAA_QUALITY__PS > 12)
            if(doneNP) {
                if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
```

```
                                if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                                doneN = abs(lumaEndN) >= gradientScaled;
                                doneP = abs(lumaEndP) >= gradientScaled;
                                if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P12;
                                if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P12;
                                doneNP = (!doneN) || (!doneP);
                                if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P12;
                                if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P12;
/*--------------------------------------------------------------------------*/
                            }
                            #endif
/*--------------------------------------------------------------------------*/
                        }
                        #endif
/*--------------------------------------------------------------------------*/
                    }
                    #endif
/*--------------------------------------------------------------------------*/
                }
                #endif
/*--------------------------------------------------------------------------*/
            }
            #endif
/*--------------------------------------------------------------------------*/
        }
        #endif
/*--------------------------------------------------------------------------*/
    FxaaFloat dstN = posM.x - posN.x;
    FxaaFloat dstP = posP.x - posM.x;
    if(!horzSpan) dstN = posM.y - posN.y;
```

```
    if(!horzSpan) dstP = posP.y - posM.y;
/*--------------------------------------------------------------------------*/
    FxaaBool goodSpanN = (lumaEndN < 0.0) != lumaMLTZero;
    FxaaFloat spanLength = (dstP + dstN);
    FxaaBool goodSpanP = (lumaEndP < 0.0) != lumaMLTZero;
    FxaaFloat spanLengthRcp = 1.0/spanLength;
/*--------------------------------------------------------------------------*/
    FxaaBool directionN = dstN < dstP;
    FxaaFloat dst = min(dstN, dstP);
    FxaaBool goodSpan = directionN ? goodSpanN : goodSpanP;
    FxaaFloat subpixG = subpixF * subpixF;
    FxaaFloat pixelOffset = (dst * (-spanLengthRcp)) + 0.5;
    FxaaFloat subpixH = subpixG * fxaaQualitySubpix;
/*--------------------------------------------------------------------------*/
    FxaaFloat pixelOffsetGood = goodSpan ? pixelOffset : 0.0;
    FxaaFloat pixelOffsetSubpix = max(pixelOffsetGood, subpixH);
    if(!horzSpan) posM.x += pixelOffsetSubpix * lengthSign;
    if( horzSpan) posM.y += pixelOffsetSubpix * lengthSign;
    #if (FXAA_DISCARD == 1)
        return FxaaTexTop(tex, posM);
    #else
        return FxaaFloat4(FxaaTexTop(tex, posM).xyz, lumaM);
    #endif
}
/*==========================================================================
================*/
#endif




//--------------------------------------------------------------------------
-----
// File:            es3-kepler\FXAA\assets\shaders/FXAA_Extreme_Quality.frag
// SDK Version: v2.0
// Email:           gameworks@nvidia.com
// Site:            http://developer.nvidia.com/
//
// Copyright (c) 2014, NVIDIA CORPORATION. All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions
// are met:
//  * Redistributions of source code must retain the above copyright
```

```
//       notice, this list of conditions and the following disclaimer.
//   * Redistributions in binary form must reproduce the above copyright
//       notice, this list of conditions and the following disclaimer in the
//       documentation and/or other materials provided with the distribution.
//   * Neither the name of NVIDIA CORPORATION nor the names of its
//       contributors may be used to endorse or promote products derived
//       from this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS ``AS IS'' AND ANY
// EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
// PURPOSE ARE DISCLAIMED.    IN NO EVENT SHALL THE COPYRIGHT OWNER OR
// CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
// EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
// PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
// PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
// OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
//------------------------------------------------------------------------------
-----
//#version 100

precision highp float;

uniform sampler2D uSourceTex;
uniform vec2 RCPFrame;
varying vec2 vTexCoord;

void main(void)
{
    gl_FragColor = FxaaPixelShader(vTexCoord,
                    FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f),     // FxaaFloat4 fxaaConsolePosPos,
        uSourceTex,                                 // FxaaTex tex,
        uSourceTex,                                 // FxaaTex
fxaaConsole360TexExpBiasNegOne,
        uSourceTex,                                 // FxaaTex
fxaaConsole360TexExpBiasNegTwo,
        RCPFrame,                                   // FxaaFloat2 fxaaQualityRcpFrame,
        FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f),     // FxaaFloat4 fxaaConsoleRcpFrameOpt,
        FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f),     // FxaaFloat4 fxaaConsoleRcpFrameOpt2,
        FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f),     // FxaaFloat4 fxaaConsole360RcpFrameOpt2,
        0.75f,                                          // FxaaFloat fxaaQualitySubpix,
```

```
        0.166f,                                         // FxaaFloat
fxaaQualityEdgeThreshold,
        0.0833f,                                    // FxaaFloat
fxaaQualityEdgeThresholdMin,
        0.0f,                                       // FxaaFloat fxaaConsoleEdgeSharpness,
        0.0f,                                       // FxaaFloat fxaaConsoleEdgeThreshold,
        0.0f,                                       // FxaaFloat
fxaaConsoleEdgeThresholdMin,
        FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f)      // FxaaFloat fxaaConsole360ConstDir,
    );
}
//////////////////////////FXAA_Fastest.frag///////////////////////////////////////////
#define FXAA_PC 1
#define FXAA_GLSL_130 1
#define FXAA_QUALITY__PRESET 10

#define FXAA_GREEN_AS_LUMA 1


/*--------------------------------------------------------------------------*/
#ifndef FXAA_PC_CONSOLE
    //
    // The console algorithm for PC is included
    // for developers targeting really low spec machines.
    // Likely better to just run FXAA_PC, and use a really low preset.
    //
    #define FXAA_PC_CONSOLE 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_GLSL_120
    #define FXAA_GLSL_120 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_GLSL_130
    #define FXAA_GLSL_130 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_HLSL_3
    #define FXAA_HLSL_3 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_HLSL_4
    #define FXAA_HLSL_4 0
#endif
```

```
/*--------------------------------------------------------------------------*/
#ifndef FXAA_HLSL_5
    #define FXAA_HLSL_5 0
#endif
/*==========================================================================
================*/
#ifndef FXAA_GREEN_AS_LUMA
    //
    // For those using non-linear color,
    // and either not able to get luma in alpha, or not wanting to,
    // this enables FXAA to run using green as a proxy for luma.
    // So with this enabled, no need to pack luma in alpha.
    //
    // This will turn off AA on anything which lacks some amount of green.
    // Pure red and blue or combination of only R and B, will get no AA.
    //
    // Might want to lower the settings for both,
    //      fxaaConsoleEdgeThresholdMin
    //      fxaaQualityEdgeThresholdMin
    // In order to insure AA does not get turned off on colors
    // which contain a minor amount of green.
    //
    // 1 = On.
    // 0 = Off.
    //
    #define FXAA_GREEN_AS_LUMA 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_EARLY_EXIT
    //
    // Controls algorithm's early exit path.
    // On PS3 turning this ON adds 2 cycles to the shader.
    // On 360 turning this OFF adds 10ths of a millisecond to the shader.
    // Turning this off on console will result in a more blurry image.
    // So this defaults to on.
    //
    // 1 = On.
    // 0 = Off.
    //
    #define FXAA_EARLY_EXIT 1
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_DISCARD
    //
```

```
    // Only valid for PC OpenGL currently.
    // Probably will not work when FXAA_GREEN_AS_LUMA = 1.
    //
    // 1 = Use discard on pixels which don't need AA.
    //         For APIs which enable concurrent TEX+ROP from same surface.
    // 0 = Return unchanged color on pixels which don't need AA.
    //
    #define FXAA_DISCARD 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_FAST_PIXEL_OFFSET
    //
    // Used for GLSL 120 only.
    //
    // 1 = GL API supports fast pixel offsets
    // 0 = do not use fast pixel offsets
    //
    #ifdef GL_EXT_gpu_shader4
        #define FXAA_FAST_PIXEL_OFFSET 1
    #endif
    #ifdef GL_NV_gpu_shader5
        #define FXAA_FAST_PIXEL_OFFSET 1
    #endif
    #ifdef GL_ARB_gpu_shader5
        #define FXAA_FAST_PIXEL_OFFSET 1
    #endif
    #ifndef FXAA_FAST_PIXEL_OFFSET
        #define FXAA_FAST_PIXEL_OFFSET 0
    #endif
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_GATHER4_ALPHA
    //
    // 1 = API supports gather4 on alpha channel.
    // 0 = API does not support gather4 on alpha channel.
    //
    #if (FXAA_HLSL_5 == 1)
        #define FXAA_GATHER4_ALPHA 1
    #endif
    #ifdef GL_ARB_gpu_shader5
        #define FXAA_GATHER4_ALPHA 1
    #endif
    #ifdef GL_NV_gpu_shader5
        #define FXAA_GATHER4_ALPHA 1
```

```
    #endif
    #ifndef FXAA_GATHER4_ALPHA
        #define FXAA_GATHER4_ALPHA 0
    #endif
#endif


/*============================================================
==================
                        FXAA QUALITY - TUNING KNOBS
------------------------------------------------------------------------
NOTE the other tuning knobs are now in the shader function inputs!
============================================================
==================*/
#ifndef FXAA_QUALITY__PRESET
    //
    // Choose the quality preset.
    // This needs to be compiled into the shader as it effects code.
    // Best option to include multiple presets is to
    // in each shader define the preset, then include this file.
    //
    // OPTIONS
    // -----------------------------------------------------------------
    // 10 to 15 - default medium dither (10=fastest, 15=highest quality)
    // 20 to 29 - less dither, more expensive (20=fastest, 29=highest quality)
    // 39       - no dither, very expensive
    //
    // NOTES
    // -----------------------------------------------------------------
    // 12 = slightly faster then FXAA 3.9 and higher edge quality (default)
    // 13 = about same speed as FXAA 3.9 and better than 12
    // 23 = closest to FXAA 3.9 visually and performance wise
    //  _ = the lowest digit is directly related to performance
    // _  = the highest digit is directly related to style
    //
    #define FXAA_QUALITY__PRESET 12
#endif


/*============================================================
==================

                        FXAA QUALITY - PRESETS
```

```
============================================================
=================*/

/*============================================================
==================
                        FXAA QUALITY - MEDIUM DITHER PRESETS
============================================================
=================*/
#if (FXAA_QUALITY__PRESET == 10)
    #define FXAA_QUALITY__PS 3
    #define FXAA_QUALITY__P0 1.5
    #define FXAA_QUALITY__P1 3.0
    #define FXAA_QUALITY__P2 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 11)
    #define FXAA_QUALITY__PS 4
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 3.0
    #define FXAA_QUALITY__P3 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 12)
    #define FXAA_QUALITY__PS 5
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 4.0
    #define FXAA_QUALITY__P4 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 13)
    #define FXAA_QUALITY__PS 6
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 4.0
    #define FXAA_QUALITY__P5 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 14)
    #define FXAA_QUALITY__PS 7
```

```
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 4.0
    #define FXAA_QUALITY__P6 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 15)
    #define FXAA_QUALITY__PS 8
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 2.0
    #define FXAA_QUALITY__P6 4.0
    #define FXAA_QUALITY__P7 12.0
#endif

/*============================================================================
==================
                        FXAA QUALITY - LOW DITHER PRESETS
============================================================================
==================*/
#if (FXAA_QUALITY__PRESET == 20)
    #define FXAA_QUALITY__PS 3
    #define FXAA_QUALITY__P0 1.5
    #define FXAA_QUALITY__P1 2.0
    #define FXAA_QUALITY__P2 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 21)
    #define FXAA_QUALITY__PS 4
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 22)
    #define FXAA_QUALITY__PS 5
    #define FXAA_QUALITY__P0 1.0
```

```
        #define FXAA_QUALITY__P1 1.5
        #define FXAA_QUALITY__P2 2.0
        #define FXAA_QUALITY__P3 2.0
        #define FXAA_QUALITY__P4 8.0
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PRESET == 23)
        #define FXAA_QUALITY__PS 6
        #define FXAA_QUALITY__P0 1.0
        #define FXAA_QUALITY__P1 1.5
        #define FXAA_QUALITY__P2 2.0
        #define FXAA_QUALITY__P3 2.0
        #define FXAA_QUALITY__P4 2.0
        #define FXAA_QUALITY__P5 8.0
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PRESET == 24)
        #define FXAA_QUALITY__PS 7
        #define FXAA_QUALITY__P0 1.0
        #define FXAA_QUALITY__P1 1.5
        #define FXAA_QUALITY__P2 2.0
        #define FXAA_QUALITY__P3 2.0
        #define FXAA_QUALITY__P4 2.0
        #define FXAA_QUALITY__P5 3.0
        #define FXAA_QUALITY__P6 8.0
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PRESET == 25)
        #define FXAA_QUALITY__PS 8
        #define FXAA_QUALITY__P0 1.0
        #define FXAA_QUALITY__P1 1.5
        #define FXAA_QUALITY__P2 2.0
        #define FXAA_QUALITY__P3 2.0
        #define FXAA_QUALITY__P4 2.0
        #define FXAA_QUALITY__P5 2.0
        #define FXAA_QUALITY__P6 4.0
        #define FXAA_QUALITY__P7 8.0
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PRESET == 26)
        #define FXAA_QUALITY__PS 9
        #define FXAA_QUALITY__P0 1.0
        #define FXAA_QUALITY__P1 1.5
        #define FXAA_QUALITY__P2 2.0
```

```
        #define FXAA_QUALITY__P3 2.0
        #define FXAA_QUALITY__P4 2.0
        #define FXAA_QUALITY__P5 2.0
        #define FXAA_QUALITY__P6 2.0
        #define FXAA_QUALITY__P7 4.0
        #define FXAA_QUALITY__P8 8.0
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PRESET == 27)
        #define FXAA_QUALITY__PS 10
        #define FXAA_QUALITY__P0 1.0
        #define FXAA_QUALITY__P1 1.5
        #define FXAA_QUALITY__P2 2.0
        #define FXAA_QUALITY__P3 2.0
        #define FXAA_QUALITY__P4 2.0
        #define FXAA_QUALITY__P5 2.0
        #define FXAA_QUALITY__P6 2.0
        #define FXAA_QUALITY__P7 2.0
        #define FXAA_QUALITY__P8 4.0
        #define FXAA_QUALITY__P9 8.0
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PRESET == 28)
        #define FXAA_QUALITY__PS 11
        #define FXAA_QUALITY__P0 1.0
        #define FXAA_QUALITY__P1 1.5
        #define FXAA_QUALITY__P2 2.0
        #define FXAA_QUALITY__P3 2.0
        #define FXAA_QUALITY__P4 2.0
        #define FXAA_QUALITY__P5 2.0
        #define FXAA_QUALITY__P6 2.0
        #define FXAA_QUALITY__P7 2.0
        #define FXAA_QUALITY__P8 2.0
        #define FXAA_QUALITY__P9 4.0
        #define FXAA_QUALITY__P10 8.0
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PRESET == 29)
        #define FXAA_QUALITY__PS 12
        #define FXAA_QUALITY__P0 1.0
        #define FXAA_QUALITY__P1 1.5
        #define FXAA_QUALITY__P2 2.0
        #define FXAA_QUALITY__P3 2.0
        #define FXAA_QUALITY__P4 2.0
```

```
    #define FXAA_QUALITY__P5 2.0
    #define FXAA_QUALITY__P6 2.0
    #define FXAA_QUALITY__P7 2.0
    #define FXAA_QUALITY__P8 2.0
    #define FXAA_QUALITY__P9 2.0
    #define FXAA_QUALITY__P10 4.0
    #define FXAA_QUALITY__P11 8.0
#endif

/*============================================================================
                        FXAA QUALITY - EXTREME QUALITY
============================================================================*/
#if (FXAA_QUALITY__PRESET == 39)
    #define FXAA_QUALITY__PS 12
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.0
    #define FXAA_QUALITY__P2 1.0
    #define FXAA_QUALITY__P3 1.0
    #define FXAA_QUALITY__P4 1.0
    #define FXAA_QUALITY__P5 1.5
    #define FXAA_QUALITY__P6 2.0
    #define FXAA_QUALITY__P7 2.0
    #define FXAA_QUALITY__P8 2.0
    #define FXAA_QUALITY__P9 2.0
    #define FXAA_QUALITY__P10 4.0
    #define FXAA_QUALITY__P11 8.0
#endif



/*============================================================================

                            API PORTING

============================================================================*/
#if (FXAA_GLSL_120 == 1) || (FXAA_GLSL_130 == 1)
    #define FxaaBool bool
    #define FxaaDiscard discard
    #define FxaaFloat float
    #define FxaaFloat2 vec2
```

```
    #define FxaaFloat3 vec3
    #define FxaaFloat4 vec4
    #define FxaaHalf float
    #define FxaaHalf2 vec2
    #define FxaaHalf3 vec3
    #define FxaaHalf4 vec4
    #define FxaaInt2 ivec2
    #define FxaaSat(x) clamp(x, 0.0, 1.0)
    #define FxaaTex sampler2D
#else
    #define FxaaBool bool
    #define FxaaDiscard clip(-1)
    #define FxaaFloat float
    #define FxaaFloat2 float2
    #define FxaaFloat3 float3
    #define FxaaFloat4 float4
    #define FxaaHalf half
    #define FxaaHalf2 half2
    #define FxaaHalf3 half3
    #define FxaaHalf4 half4
    #define FxaaSat(x) saturate(x)
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_GLSL_120 == 1)
    // Requires,
    //  #version 120
    // And at least,
    //  #extension GL_EXT_gpu_shader4 : enable
    //  (or set FXAA_FAST_PIXEL_OFFSET 1 to work like DX9)
    #define FxaaTexTop(t, p) texture2DLod(t, p, 0.0)
    #if (FXAA_FAST_PIXEL_OFFSET == 1)
        #define FxaaTexOff(t, p, o, r) texture2DLodOffset(t, p, 0.0, o)
    #else
        #define FxaaTexOff(t, p, o, r) texture2DLod(t, p + (o * r), 0.0)
    #endif
    #if (FXAA_GATHER4_ALPHA == 1)
        // use #extension GL_ARB_gpu_shader5 : enable
        #define FxaaTexAlpha4(t, p) textureGather(t, p, 3)
        #define FxaaTexOffAlpha4(t, p, o) textureGatherOffset(t, p, o, 3)
        #define FxaaTexGreen4(t, p) textureGather(t, p, 1)
        #define FxaaTexOffGreen4(t, p, o) textureGatherOffset(t, p, o, 1)
    #endif
#endif
/*--------------------------------------------------------------------------*/
```

```
#if (FXAA_GLSL_130 == 1)
    // Requires "#version 130" or better
    #define FxaaTexTop(t, p) textureLod(t, p, 0.0)
    #define FxaaTexOff(t, p, o, r) textureLodOffset(t, p, 0.0, o)
    #if (FXAA_GATHER4_ALPHA == 1)
        // use #extension GL_ARB_gpu_shader5 : enable
        #define FxaaTexAlpha4(t, p) textureGather(t, p, 3)
        #define FxaaTexOffAlpha4(t, p, o) textureGatherOffset(t, p, o, 3)
        #define FxaaTexGreen4(t, p) textureGather(t, p, 1)
        #define FxaaTexOffGreen4(t, p, o) textureGatherOffset(t, p, o, 1)
    #endif
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_HLSL_3 == 1) || (FXAA_360 == 1) || (FXAA_PS3 == 1)
    #define FxaaInt2 float2
    #define FxaaTex sampler2D
    #define FxaaTexTop(t, p) tex2Dlod(t, float4(p, 0.0, 0.0))
    #define FxaaTexOff(t, p, o, r) tex2Dlod(t, float4(p + (o * r), 0, 0))
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_HLSL_4 == 1)
    #define FxaaInt2 int2
    struct FxaaTex { SamplerState smpl; Texture2D tex; };
    #define FxaaTexTop(t, p) t.tex.SampleLevel(t.smpl, p, 0.0)
    #define FxaaTexOff(t, p, o, r) t.tex.SampleLevel(t.smpl, p, 0.0, o)
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_HLSL_5 == 1)
    #define FxaaInt2 int2
    struct FxaaTex { SamplerState smpl; Texture2D tex; };
    #define FxaaTexTop(t, p) t.tex.SampleLevel(t.smpl, p, 0.0)
    #define FxaaTexOff(t, p, o, r) t.tex.SampleLevel(t.smpl, p, 0.0, o)
    #define FxaaTexAlpha4(t, p) t.tex.GatherAlpha(t.smpl, p)
    #define FxaaTexOffAlpha4(t, p, o) t.tex.GatherAlpha(t.smpl, p, o)
    #define FxaaTexGreen4(t, p) t.tex.GatherGreen(t.smpl, p)
    #define FxaaTexOffGreen4(t, p, o) t.tex.GatherGreen(t.smpl, p, o)
#endif


/*============================================================================
                    GREEN AS LUMA OPTION SUPPORT FUNCTION
============================================================================*/
```

```
#if (FXAA_GREEN_AS_LUMA == 0)
    FxaaFloat FxaaLuma(FxaaFloat4 rgba) { return rgba.w; }
#else
    FxaaFloat FxaaLuma(FxaaFloat4 rgba) { return rgba.y; }
#endif



/*==========================================================================
==================

                                FXAA3 QUALITY - PC

============================================================================
==================*/
#if (FXAA_PC == 1)
/*--------------------------------------------------------------------------*/
FxaaFloat4 FxaaPixelShader(
    //
    // Use noperspective interpolation here (turn off perspective interpolation).
    // {xy} = center of pixel
    FxaaFloat2 pos,
    //
    // Used only for FXAA Console, and not used on the 360 version.
    // Use noperspective interpolation here (turn off perspective interpolation).
    // {xy__} = upper left of pixel
    // {__zw} = lower right of pixel
    FxaaFloat4 fxaaConsolePosPos,
    //
    // Input color texture.
    // {rgb_} = color in linear or perceptual color space
    // if (FXAA_GREEN_AS_LUMA == 0)
    //     {___a} = luma in perceptual color space (not linear)
    FxaaTex tex,
    //
    // Only used on the optimized 360 version of FXAA Console.
    // For everything but 360, just use the same input here as for "tex".
    // For 360, same texture, just alias with a 2nd sampler.
    // This sampler needs to have an exponent bias of -1.
    FxaaTex fxaaConsole360TexExpBiasNegOne,
    //
    // Only used on the optimized 360 version of FXAA Console.
    // For everything but 360, just use the same input here as for "tex".
```

```
// For 360, same texture, just alias with a 3nd sampler.
// This sampler needs to have an exponent bias of -2.
FxaaTex fxaaConsole360TexExpBiasNegTwo,
//
// Only used on FXAA Quality.
// This must be from a constant/uniform.
// {x_} = 1.0/screenWidthInPixels
// {_y} = 1.0/screenHeightInPixels
FxaaFloat2 fxaaQualityRcpFrame,
//
// Only used on FXAA Console.
// This must be from a constant/uniform.
// This effects sub-pixel AA quality and inversely sharpness.
//     Where N ranges between,
//         N = 0.50 (default)
//         N = 0.33 (sharper)
// {x___} = -N/screenWidthInPixels
// {_y__} = -N/screenHeightInPixels
// {__z_} =  N/screenWidthInPixels
// {___w} =  N/screenHeightInPixels
FxaaFloat4 fxaaConsoleRcpFrameOpt,
//
// Only used on FXAA Console.
// Not used on 360, but used on PS3 and PC.
// This must be from a constant/uniform.
// {x___} = -2.0/screenWidthInPixels
// {_y__} = -2.0/screenHeightInPixels
// {__z_} =  2.0/screenWidthInPixels
// {___w} =  2.0/screenHeightInPixels
FxaaFloat4 fxaaConsoleRcpFrameOpt2,
//
// Only used on FXAA Console.
// Only used on 360 in place of fxaaConsoleRcpFrameOpt2.
// This must be from a constant/uniform.
// {x___} =  8.0/screenWidthInPixels
// {_y__} =  8.0/screenHeightInPixels
// {__z_} = -4.0/screenWidthInPixels
// {___w} = -4.0/screenHeightInPixels
FxaaFloat4 fxaaConsole360RcpFrameOpt2,
//
// Only used on FXAA Quality.
// This used to be the FXAA_QUALITY__SUBPIX define.
// It is here now to allow easier tuning.
// Choose the amount of sub-pixel aliasing removal.
```

```
// This can effect sharpness.
//     1.00 - upper limit (softer)
//     0.75 - default amount of filtering
//     0.50 - lower limit (sharper, less sub-pixel aliasing removal)
//     0.25 - almost off
//     0.00 - completely off
FxaaFloat fxaaQualitySubpix,
//
// Only used on FXAA Quality.
// This used to be the FXAA_QUALITY__EDGE_THRESHOLD define.
// It is here now to allow easier tuning.
// The minimum amount of local contrast required to apply algorithm.
//     0.333 - too little (faster)
//     0.250 - low quality
//     0.166 - default
//     0.125 - high quality
//     0.063 - overkill (slower)
FxaaFloat fxaaQualityEdgeThreshold,
//
// Only used on FXAA Quality.
// This used to be the FXAA_QUALITY__EDGE_THRESHOLD_MIN define.
// It is here now to allow easier tuning.
// Trims the algorithm from processing darks.
//     0.0833 - upper limit (default, the start of visible unfiltered edges)
//     0.0625 - high quality (faster)
//     0.0312 - visible limit (slower)
// Special notes when using FXAA_GREEN_AS_LUMA,
//     Likely want to set this to zero.
//     As colors that are mostly not-green
//     will appear very dark in the green channel!
//     Tune by looking at mostly non-green content,
//     then start at zero and increase until aliasing is a problem.
FxaaFloat fxaaQualityEdgeThresholdMin,
//
// Only used on FXAA Console.
// This used to be the FXAA_CONSOLE__EDGE_SHARPNESS define.
// It is here now to allow easier tuning.
// This does not effect PS3, as this needs to be compiled in.
//     Use FXAA_CONSOLE__PS3_EDGE_SHARPNESS for PS3.
//     Due to the PS3 being ALU bound,
//     there are only three safe values here: 2 and 4 and 8.
//     These options use the shaders ability to a free *|/ by 2|4|8.
// For all other platforms can be a non-power of two.
//     8.0 is sharper (default!!!)
```

```
    //      4.0 is softer
    //      2.0 is really soft (good only for vector graphics inputs)
    FxaaFloat fxaaConsoleEdgeSharpness,
    //
    // Only used on FXAA Console.
    // This used to be the FXAA_CONSOLE__EDGE_THRESHOLD define.
    // It is here now to allow easier tuning.
    // This does not effect PS3, as this needs to be compiled in.
    //      Use FXAA_CONSOLE__PS3_EDGE_THRESHOLD for PS3.
    //      Due to the PS3 being ALU bound,
    //      there are only two safe values here: 1/4 and 1/8.
    //      These options use the shaders ability to a free *|/ by 2|4|8.
    // The console setting has a different mapping than the quality setting.
    // Other platforms can use other values.
    //      0.125 leaves less aliasing, but is softer (default!!!)
    //      0.25 leaves more aliasing, and is sharper
    FxaaFloat fxaaConsoleEdgeThreshold,
    //
    // Only used on FXAA Console.
    // This used to be the FXAA_CONSOLE__EDGE_THRESHOLD_MIN define.
    // It is here now to allow easier tuning.
    // Trims the algorithm from processing darks.
    // The console setting has a different mapping than the quality setting.
    // This only applies when FXAA_EARLY_EXIT is 1.
    // This does not apply to PS3,
    // PS3 was simplified to avoid more shader instructions.
    //      0.06 - faster but more aliasing in darks
    //      0.05 - default
    //      0.04 - slower and less aliasing in darks
    // Special notes when using FXAA_GREEN_AS_LUMA,
    //      Likely want to set this to zero.
    //      As colors that are mostly not-green
    //      will appear very dark in the green channel!
    //      Tune by looking at mostly non-green content,
    //      then start at zero and increase until aliasing is a problem.
    FxaaFloat fxaaConsoleEdgeThresholdMin,
    //
    // Extra constants for 360 FXAA Console only.
    // Use zeros or anything else for other platforms.
    // These must be in physical constant registers and NOT immedates.
    // Immedates will result in compiler un-optimizing.
    // {xyzw} = float4(1.0, -1.0, 0.25, -0.25)
    FxaaFloat4 fxaaConsole360ConstDir
) {
```

```
/*--------------------------------------------------------------------------*/
    FxaaFloat2 posM;
    posM.x = pos.x;
    posM.y = pos.y;
    #if (FXAA_GATHER4_ALPHA == 1)
        #if (FXAA_DISCARD == 0)
            FxaaFloat4 rgbyM = FxaaTexTop(tex, posM);
            #if (FXAA_GREEN_AS_LUMA == 0)
                #define lumaM rgbyM.w
            #else
                #define lumaM rgbyM.y
            #endif
        #endif
        #if (FXAA_GREEN_AS_LUMA == 0)
            FxaaFloat4 luma4A = FxaaTexAlpha4(tex, posM);
            FxaaFloat4 luma4B = FxaaTexOffAlpha4(tex, posM, FxaaInt2(-1, -1));
        #else
            FxaaFloat4 luma4A = FxaaTexGreen4(tex, posM);
            FxaaFloat4 luma4B = FxaaTexOffGreen4(tex, posM, FxaaInt2(-1, -1));
        #endif
        #if (FXAA_DISCARD == 1)
            #define lumaM luma4A.w
        #endif
        #define lumaE luma4A.z
        #define lumaS luma4A.x
        #define lumaSE luma4A.y
        #define lumaNW luma4B.w
        #define lumaN luma4B.z
        #define lumaW luma4B.x
    #else
        FxaaFloat4 rgbyM = FxaaTexTop(tex, posM);
        #if (FXAA_GREEN_AS_LUMA == 0)
            #define lumaM rgbyM.w
        #else
            #define lumaM rgbyM.y
        #endif
        FxaaFloat lumaS = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 0, 1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaE = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 1, 0),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaN = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 0,-1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaW = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(-1, 0),
fxaaQualityRcpFrame.xy));
```

```
    #endif
/*--------------------------------------------------------------------------*/
    FxaaFloat maxSM = max(lumaS, lumaM);
    FxaaFloat minSM = min(lumaS, lumaM);
    FxaaFloat maxESM = max(lumaE, maxSM);
    FxaaFloat minESM = min(lumaE, minSM);
    FxaaFloat maxWN = max(lumaN, lumaW);
    FxaaFloat minWN = min(lumaN, lumaW);
    FxaaFloat rangeMax = max(maxWN, maxESM);
    FxaaFloat rangeMin = min(minWN, minESM);
    FxaaFloat rangeMaxScaled = rangeMax * fxaaQualityEdgeThreshold;
    FxaaFloat range = rangeMax - rangeMin;
    FxaaFloat rangeMaxClamped = max(fxaaQualityEdgeThresholdMin, rangeMaxScaled);
    FxaaBool earlyExit = range < rangeMaxClamped;
/*--------------------------------------------------------------------------*/
    if(earlyExit)
        #if (FXAA_DISCARD == 1)
            FxaaDiscard;
        #else
            return rgbyM;
        #endif
/*--------------------------------------------------------------------------*/
    #if (FXAA_GATHER4_ALPHA == 0)
        FxaaFloat lumaNW = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(-1,-1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaSE = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 1, 1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaNE = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 1,-1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaSW = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(-1, 1),
fxaaQualityRcpFrame.xy));
    #else
        FxaaFloat lumaNE = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(1, -1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaSW = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(-1, 1),
fxaaQualityRcpFrame.xy));
    #endif
/*--------------------------------------------------------------------------*/
    FxaaFloat lumaNS = lumaN + lumaS;
    FxaaFloat lumaWE = lumaW + lumaE;
    FxaaFloat subpixRcpRange = 1.0/range;
    FxaaFloat subpixNSWE = lumaNS + lumaWE;
    FxaaFloat edgeHorz1 = (-2.0 * lumaM) + lumaNS;
    FxaaFloat edgeVert1 = (-2.0 * lumaM) + lumaWE;
```

```
/*--------------------------------------------------------------------------*/
    FxaaFloat lumaNESE = lumaNE + lumaSE;
    FxaaFloat lumaNWNE = lumaNW + lumaNE;
    FxaaFloat edgeHorz2 = (-2.0 * lumaE) + lumaNESE;
    FxaaFloat edgeVert2 = (-2.0 * lumaN) + lumaNWNE;
/*--------------------------------------------------------------------------*/
    FxaaFloat lumaNWSW = lumaNW + lumaSW;
    FxaaFloat lumaSWSE = lumaSW + lumaSE;
    FxaaFloat edgeHorz4 = (abs(edgeHorz1) * 2.0) + abs(edgeHorz2);
    FxaaFloat edgeVert4 = (abs(edgeVert1) * 2.0) + abs(edgeVert2);
    FxaaFloat edgeHorz3 = (-2.0 * lumaW) + lumaNWSW;
    FxaaFloat edgeVert3 = (-2.0 * lumaS) + lumaSWSE;
    FxaaFloat edgeHorz = abs(edgeHorz3) + edgeHorz4;
    FxaaFloat edgeVert = abs(edgeVert3) + edgeVert4;
/*--------------------------------------------------------------------------*/
    FxaaFloat subpixNWSWNESE = lumaNWSW + lumaNESE;
    FxaaFloat lengthSign = fxaaQualityRcpFrame.x;
    FxaaBool horzSpan = edgeHorz >= edgeVert;
    FxaaFloat subpixA = subpixNSWE * 2.0 + subpixNWSWNESE;
/*--------------------------------------------------------------------------*/
    if(!horzSpan) lumaN = lumaW;
    if(!horzSpan) lumaS = lumaE;
    if(horzSpan) lengthSign = fxaaQualityRcpFrame.y;
    FxaaFloat subpixB = (subpixA * (1.0/12.0)) - lumaM;
/*--------------------------------------------------------------------------*/
    FxaaFloat gradientN = lumaN - lumaM;
    FxaaFloat gradientS = lumaS - lumaM;
    FxaaFloat lumaNN = lumaN + lumaM;
    FxaaFloat lumaSS = lumaS + lumaM;
    FxaaBool pairN = abs(gradientN) >= abs(gradientS);
    FxaaFloat gradient = max(abs(gradientN), abs(gradientS));
    if(pairN) lengthSign = -lengthSign;
    FxaaFloat subpixC = FxaaSat(abs(subpixB) * subpixRcpRange);
/*--------------------------------------------------------------------------*/
    FxaaFloat2 posB;
    posB.x = posM.x;
    posB.y = posM.y;
    FxaaFloat2 offNP;
    offNP.x = (!horzSpan) ? 0.0 : fxaaQualityRcpFrame.x;
    offNP.y = ( horzSpan) ? 0.0 : fxaaQualityRcpFrame.y;
    if(!horzSpan) posB.x += lengthSign * 0.5;
    if( horzSpan) posB.y += lengthSign * 0.5;
/*--------------------------------------------------------------------------*/
    FxaaFloat2 posN;
```

```
    posN.x = posB.x - offNP.x * FXAA_QUALITY__P0;
    posN.y = posB.y - offNP.y * FXAA_QUALITY__P0;
    FxaaFloat2 posP;
    posP.x = posB.x + offNP.x * FXAA_QUALITY__P0;
    posP.y = posB.y + offNP.y * FXAA_QUALITY__P0;
    FxaaFloat subpixD = ((-2.0)*subpixC) + 3.0;
    FxaaFloat lumaEndN = FxaaLuma(FxaaTexTop(tex, posN));
    FxaaFloat subpixE = subpixC * subpixC;
    FxaaFloat lumaEndP = FxaaLuma(FxaaTexTop(tex, posP));
/*--------------------------------------------------------------------------*/
    if(!pairN) lumaNN = lumaSS;
    FxaaFloat gradientScaled = gradient * 1.0/4.0;
    FxaaFloat lumaMM = lumaM - lumaNN * 0.5;
    FxaaFloat subpixF = subpixD * subpixE;
    FxaaBool lumaMLTZero = lumaMM < 0.0;
/*--------------------------------------------------------------------------*/
    lumaEndN -= lumaNN * 0.5;
    lumaEndP -= lumaNN * 0.5;
    FxaaBool doneN = abs(lumaEndN) >= gradientScaled;
    FxaaBool doneP = abs(lumaEndP) >= gradientScaled;
    if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P1;
    if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P1;
    FxaaBool doneNP = (!doneN) || (!doneP);
    if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P1;
    if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P1;
/*--------------------------------------------------------------------------*/
    if(doneNP) {
        if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
        if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
        if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
        if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
        doneN = abs(lumaEndN) >= gradientScaled;
        doneP = abs(lumaEndP) >= gradientScaled;
        if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P2;
        if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P2;
        doneNP = (!doneN) || (!doneP);
        if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P2;
        if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P2;
/*--------------------------------------------------------------------------*/
        #if (FXAA_QUALITY__PS > 3)
        if(doneNP) {
            if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
            if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
            if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
```

```
            if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
            doneN = abs(lumaEndN) >= gradientScaled;
            doneP = abs(lumaEndP) >= gradientScaled;
            if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P3;
            if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P3;
            doneNP = (!doneN) || (!doneP);
            if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P3;
            if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P3;
/*--------------------------------------------------------------------------*/
            #if (FXAA_QUALITY__PS > 4)
            if(doneNP) {
                if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                doneN = abs(lumaEndN) >= gradientScaled;
                doneP = abs(lumaEndP) >= gradientScaled;
                if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P4;
                if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P4;
                doneNP = (!doneN) || (!doneP);
                if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P4;
                if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P4;
/*--------------------------------------------------------------------------*/
                #if (FXAA_QUALITY__PS > 5)
                if(doneNP) {
                    if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                    if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                    if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                    if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                    doneN = abs(lumaEndN) >= gradientScaled;
                    doneP = abs(lumaEndP) >= gradientScaled;
                    if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P5;
                    if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P5;
                    doneNP = (!doneN) || (!doneP);
                    if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P5;
                    if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P5;
/*--------------------------------------------------------------------------*/
                    #if (FXAA_QUALITY__PS > 6)
                    if(doneNP) {
                        if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                        if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                        if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                        if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                        doneN = abs(lumaEndN) >= gradientScaled;
```

```
                            doneP = abs(lumaEndP) >= gradientScaled;
                            if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P6;
                            if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P6;
                            doneNP = (!doneN) || (!doneP);
                            if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P6;
                            if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P6;
/*--------------------------------------------------------------------------*/
                            #if (FXAA_QUALITY__PS > 7)
                            if(doneNP) {
                                if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex,
posN.xy));
                                if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex,
posP.xy));
                                if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                                if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                                doneN = abs(lumaEndN) >= gradientScaled;
                                doneP = abs(lumaEndP) >= gradientScaled;
                                if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P7;
                                if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P7;
                                doneNP = (!doneN) || (!doneP);
                                if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P7;
                                if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P7;
/*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PS > 8)
    if(doneNP) {
        if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
        if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
        if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
        if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
        doneN = abs(lumaEndN) >= gradientScaled;
        doneP = abs(lumaEndP) >= gradientScaled;
        if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P8;
        if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P8;
        doneNP = (!doneN) || (!doneP);
        if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P8;
        if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P8;
/*--------------------------------------------------------------------------*/
        #if (FXAA_QUALITY__PS > 9)
        if(doneNP) {
            if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
            if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
            if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
            if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
            doneN = abs(lumaEndN) >= gradientScaled;
```

```
    doneP = abs(lumaEndP) >= gradientScaled;
    if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P9;
    if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P9;
    doneNP = (!doneN) || (!doneP);
    if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P9;
    if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P9;
/*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PS > 10)
    if(doneNP) {
        if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
        if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
        if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
        if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
        doneN = abs(lumaEndN) >= gradientScaled;
        doneP = abs(lumaEndP) >= gradientScaled;
        if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P10;
        if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P10;
        doneNP = (!doneN) || (!doneP);
        if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P10;
        if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P10;
/*--------------------------------------------------------------------------*/
        #if (FXAA_QUALITY__PS > 11)
        if(doneNP) {
            if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
            if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
            if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
            if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
            doneN = abs(lumaEndN) >= gradientScaled;
            doneP = abs(lumaEndP) >= gradientScaled;
            if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P11;
            if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P11;
            doneNP = (!doneN) || (!doneP);
            if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P11;
            if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P11;
/*--------------------------------------------------------------------------*/
            #if (FXAA_QUALITY__PS > 12)
            if(doneNP) {
                if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                doneN = abs(lumaEndN) >= gradientScaled;
                doneP = abs(lumaEndP) >= gradientScaled;
                if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P12;
```

```
                                if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P12;
                                doneNP = (!doneN) || (!doneP);
                                if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P12;
                                if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P12;
/*--------------------------------------------------------------------------*/
                            }
                            #endif
/*--------------------------------------------------------------------------*/
                        }
                        #endif
/*--------------------------------------------------------------------------*/
                    }
                    #endif
/*--------------------------------------------------------------------------*/
                }
                #endif
/*--------------------------------------------------------------------------*/
            }
            #endif
/*--------------------------------------------------------------------------*/
        }
        #endif
/*--------------------------------------------------------------------------*/
                                }
                                #endif
/*--------------------------------------------------------------------------*/
                            }
                            #endif
/*--------------------------------------------------------------------------*/
                        }
                        #endif
/*--------------------------------------------------------------------------*/
                    }
                    #endif
/*--------------------------------------------------------------------------*/
                }
                #endif
/*--------------------------------------------------------------------------*/
            }
            #endif
/*--------------------------------------------------------------------------*/
        }
        #endif
/*--------------------------------------------------------------------------*/
    }
/*--------------------------------------------------------------------------*/
    FxaaFloat dstN = posM.x - posN.x;
    FxaaFloat dstP = posP.x - posM.x;
    if(!horzSpan) dstN = posM.y - posN.y;
    if(!horzSpan) dstP = posP.y - posM.y;
/*--------------------------------------------------------------------------*/
    FxaaBool goodSpanN = (lumaEndN < 0.0) != lumaMLTZero;
    FxaaFloat spanLength = (dstP + dstN);
```

```
    FxaaBool goodSpanP = (lumaEndP < 0.0) != lumaMLTZero;
    FxaaFloat spanLengthRcp = 1.0/spanLength;
/*--------------------------------------------------------------------------*/
    FxaaBool directionN = dstN < dstP;
    FxaaFloat dst = min(dstN, dstP);
    FxaaBool goodSpan = directionN ? goodSpanN : goodSpanP;
    FxaaFloat subpixG = subpixF * subpixF;
    FxaaFloat pixelOffset = (dst * (-spanLengthRcp)) + 0.5;
    FxaaFloat subpixH = subpixG * fxaaQualitySubpix;
/*--------------------------------------------------------------------------*/
    FxaaFloat pixelOffsetGood = goodSpan ? pixelOffset : 0.0;
    FxaaFloat pixelOffsetSubpix = max(pixelOffsetGood, subpixH);
    if(!horzSpan) posM.x += pixelOffsetSubpix * lengthSign;
    if( horzSpan) posM.y += pixelOffsetSubpix * lengthSign;
    #if (FXAA_DISCARD == 1)
        return FxaaTexTop(tex, posM);
    #else
        return FxaaFloat4(FxaaTexTop(tex, posM).xyz, lumaM);
    #endif
}
/*==========================================================================
================*/
#endif
```

```
//------------------------------------------------------------------------
-----
// File:            es3-kepler\FXAA\assets\shaders/FXAA_Fastest.frag
// SDK Version: v2.0
// Email:           gameworks@nvidia.com
// Site:            http://developer.nvidia.com/
//
// Copyright (c) 2014, NVIDIA CORPORATION. All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions
// are met:
//  * Redistributions of source code must retain the above copyright
//    notice, this list of conditions and the following disclaimer.
//  * Redistributions in binary form must reproduce the above copyright
//    notice, this list of conditions and the following disclaimer in the
//    documentation and/or other materials provided with the distribution.
```

```
//    * Neither the name of NVIDIA CORPORATION nor the names of its
//      contributors may be used to endorse or promote products derived
//      from this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS ``AS IS'' AND ANY
// EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
// PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
// CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
// EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
// PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
// PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
// OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
//------------------------------------------------------------------------------
-----
//#version 100

precision highp float;

uniform sampler2D uSourceTex;
uniform vec2 RCPFrame;
varying vec2 vTexCoord;

void main(void)
{
    gl_FragColor = FxaaPixelShader(vTexCoord,
                    FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f),     // FxaaFloat4 fxaaConsolePosPos,
        uSourceTex,                                 // FxaaTex tex,
        uSourceTex,                                 // FxaaTex
fxaaConsole360TexExpBiasNegOne,
        uSourceTex,                                 // FxaaTex
fxaaConsole360TexExpBiasNegTwo,
        RCPFrame,                                   // FxaaFloat2 fxaaQualityRcpFrame,
        FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f),     // FxaaFloat4 fxaaConsoleRcpFrameOpt,
        FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f),     // FxaaFloat4 fxaaConsoleRcpFrameOpt2,
        FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f),     // FxaaFloat4 fxaaConsole360RcpFrameOpt2,
        0.75f,                                          // FxaaFloat fxaaQualitySubpix,
        0.166f,                                         // FxaaFloat
fxaaQualityEdgeThreshold,
        0.0833f,                                    // FxaaFloat
fxaaQualityEdgeThresholdMin,
```

```
        0.0f,                                   // FxaaFloat fxaaConsoleEdgeSharpness,
        0.0f,                                   // FxaaFloat fxaaConsoleEdgeThreshold,
        0.0f,                                   // FxaaFloat
fxaaConsoleEdgeThresholdMin,
        FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f)      // FxaaFloat fxaaConsole360ConstDir,
    );
}
/////////////////////////////FXAA_High_Quality.frag//////////////////////////////////////
#define FXAA_PC 1
#define FXAA_GLSL_130 1
#define FXAA_QUALITY__PRESET 29

#define FXAA_GREEN_AS_LUMA 1


/*--------------------------------------------------------------------------*/
#ifndef FXAA_PC_CONSOLE
    //
    // The console algorithm for PC is included
    // for developers targeting really low spec machines.
    // Likely better to just run FXAA_PC, and use a really low preset.
    //
    #define FXAA_PC_CONSOLE 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_GLSL_120
    #define FXAA_GLSL_120 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_GLSL_130
    #define FXAA_GLSL_130 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_HLSL_3
    #define FXAA_HLSL_3 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_HLSL_4
    #define FXAA_HLSL_4 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_HLSL_5
    #define FXAA_HLSL_5 0
#endif
```

```
/*============================================================
================*/
#ifndef FXAA_GREEN_AS_LUMA
    //
    // For those using non-linear color,
    // and either not able to get luma in alpha, or not wanting to,
    // this enables FXAA to run using green as a proxy for luma.
    // So with this enabled, no need to pack luma in alpha.
    //
    // This will turn off AA on anything which lacks some amount of green.
    // Pure red and blue or combination of only R and B, will get no AA.
    //
    // Might want to lower the settings for both,
    //    fxaaConsoleEdgeThresholdMin
    //    fxaaQualityEdgeThresholdMin
    // In order to insure AA does not get turned off on colors
    // which contain a minor amount of green.
    //
    // 1 = On.
    // 0 = Off.
    //
    #define FXAA_GREEN_AS_LUMA 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_EARLY_EXIT
    //
    // Controls algorithm's early exit path.
    // On PS3 turning this ON adds 2 cycles to the shader.
    // On 360 turning this OFF adds 10ths of a millisecond to the shader.
    // Turning this off on console will result in a more blurry image.
    // So this defaults to on.
    //
    // 1 = On.
    // 0 = Off.
    //
    #define FXAA_EARLY_EXIT 1
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_DISCARD
    //
    // Only valid for PC OpenGL currently.
    // Probably will not work when FXAA_GREEN_AS_LUMA = 1.
    //
    // 1 = Use discard on pixels which don't need AA.
```

```
    //        For APIs which enable concurrent TEX+ROP from same surface.
    // 0 = Return unchanged color on pixels which don't need AA.
    //
    #define FXAA_DISCARD 0
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_FAST_PIXEL_OFFSET
    //
    // Used for GLSL 120 only.
    //
    // 1 = GL API supports fast pixel offsets
    // 0 = do not use fast pixel offsets
    //
    #ifdef GL_EXT_gpu_shader4
        #define FXAA_FAST_PIXEL_OFFSET 1
    #endif
    #ifdef GL_NV_gpu_shader5
        #define FXAA_FAST_PIXEL_OFFSET 1
    #endif
    #ifdef GL_ARB_gpu_shader5
        #define FXAA_FAST_PIXEL_OFFSET 1
    #endif
    #ifndef FXAA_FAST_PIXEL_OFFSET
        #define FXAA_FAST_PIXEL_OFFSET 0
    #endif
#endif
/*--------------------------------------------------------------------------*/
#ifndef FXAA_GATHER4_ALPHA
    //
    // 1 = API supports gather4 on alpha channel.
    // 0 = API does not support gather4 on alpha channel.
    //
    #if (FXAA_HLSL_5 == 1)
        #define FXAA_GATHER4_ALPHA 1
    #endif
    #ifdef GL_ARB_gpu_shader5
        #define FXAA_GATHER4_ALPHA 1
    #endif
    #ifdef GL_NV_gpu_shader5
        #define FXAA_GATHER4_ALPHA 1
    #endif
    #ifndef FXAA_GATHER4_ALPHA
        #define FXAA_GATHER4_ALPHA 0
    #endif
```

```
#endif


/*============================================================
==================
                        FXAA QUALITY - TUNING KNOBS
------------------------------------------------------------------------------
NOTE the other tuning knobs are now in the shader function inputs!
============================================================
==================*/
#ifndef FXAA_QUALITY__PRESET
    //
    // Choose the quality preset.
    // This needs to be compiled into the shader as it effects code.
    // Best option to include multiple presets is to
    // in each shader define the preset, then include this file.
    //
    // OPTIONS
    // -----------------------------------------------------------------
    // 10 to 15 - default medium dither (10=fastest, 15=highest quality)
    // 20 to 29 - less dither, more expensive (20=fastest, 29=highest quality)
    // 39       - no dither, very expensive
    //
    // NOTES
    // -----------------------------------------------------------------
    // 12 = slightly faster then FXAA 3.9 and higher edge quality (default)
    // 13 = about same speed as FXAA 3.9 and better than 12
    // 23 = closest to FXAA 3.9 visually and performance wise
    //  _ = the lowest digit is directly related to performance
    // _  = the highest digit is directly related to style
    //
    #define FXAA_QUALITY__PRESET 12
#endif


/*============================================================
==================

                        FXAA QUALITY - PRESETS

============================================================
==================*/

/*============================================================
```

```
==================
                                FXAA QUALITY - MEDIUM DITHER PRESETS
============================================================================
==================*/
#if (FXAA_QUALITY__PRESET == 10)
    #define FXAA_QUALITY__PS 3
    #define FXAA_QUALITY__P0 1.5
    #define FXAA_QUALITY__P1 3.0
    #define FXAA_QUALITY__P2 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 11)
    #define FXAA_QUALITY__PS 4
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 3.0
    #define FXAA_QUALITY__P3 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 12)
    #define FXAA_QUALITY__PS 5
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 4.0
    #define FXAA_QUALITY__P4 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 13)
    #define FXAA_QUALITY__PS 6
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 4.0
    #define FXAA_QUALITY__P5 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 14)
    #define FXAA_QUALITY__PS 7
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
```

```
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 4.0
    #define FXAA_QUALITY__P6 12.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 15)
    #define FXAA_QUALITY__PS 8
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 2.0
    #define FXAA_QUALITY__P6 4.0
    #define FXAA_QUALITY__P7 12.0
#endif

/*============================================================================
==================
                        FXAA QUALITY - LOW DITHER PRESETS
============================================================================
==================*/
#if (FXAA_QUALITY__PRESET == 20)
    #define FXAA_QUALITY__PS 3
    #define FXAA_QUALITY__P0 1.5
    #define FXAA_QUALITY__P1 2.0
    #define FXAA_QUALITY__P2 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 21)
    #define FXAA_QUALITY__PS 4
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 22)
    #define FXAA_QUALITY__PS 5
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 8.0
```

```
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 23)
    #define FXAA_QUALITY__PS 6
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 24)
    #define FXAA_QUALITY__PS 7
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 3.0
    #define FXAA_QUALITY__P6 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 25)
    #define FXAA_QUALITY__PS 8
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 2.0
    #define FXAA_QUALITY__P6 4.0
    #define FXAA_QUALITY__P7 8.0
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_QUALITY__PRESET == 26)
    #define FXAA_QUALITY__PS 9
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.5
    #define FXAA_QUALITY__P2 2.0
    #define FXAA_QUALITY__P3 2.0
    #define FXAA_QUALITY__P4 2.0
    #define FXAA_QUALITY__P5 2.0
    #define FXAA_QUALITY__P6 2.0
```

```
        #define FXAA_QUALITY__P7 4.0
        #define FXAA_QUALITY__P8 8.0
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PRESET == 27)
        #define FXAA_QUALITY__PS 10
        #define FXAA_QUALITY__P0 1.0
        #define FXAA_QUALITY__P1 1.5
        #define FXAA_QUALITY__P2 2.0
        #define FXAA_QUALITY__P3 2.0
        #define FXAA_QUALITY__P4 2.0
        #define FXAA_QUALITY__P5 2.0
        #define FXAA_QUALITY__P6 2.0
        #define FXAA_QUALITY__P7 2.0
        #define FXAA_QUALITY__P8 4.0
        #define FXAA_QUALITY__P9 8.0
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PRESET == 28)
        #define FXAA_QUALITY__PS 11
        #define FXAA_QUALITY__P0 1.0
        #define FXAA_QUALITY__P1 1.5
        #define FXAA_QUALITY__P2 2.0
        #define FXAA_QUALITY__P3 2.0
        #define FXAA_QUALITY__P4 2.0
        #define FXAA_QUALITY__P5 2.0
        #define FXAA_QUALITY__P6 2.0
        #define FXAA_QUALITY__P7 2.0
        #define FXAA_QUALITY__P8 2.0
        #define FXAA_QUALITY__P9 4.0
        #define FXAA_QUALITY__P10 8.0
    #endif
    /*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PRESET == 29)
        #define FXAA_QUALITY__PS 12
        #define FXAA_QUALITY__P0 1.0
        #define FXAA_QUALITY__P1 1.5
        #define FXAA_QUALITY__P2 2.0
        #define FXAA_QUALITY__P3 2.0
        #define FXAA_QUALITY__P4 2.0
        #define FXAA_QUALITY__P5 2.0
        #define FXAA_QUALITY__P6 2.0
        #define FXAA_QUALITY__P7 2.0
        #define FXAA_QUALITY__P8 2.0
```

```
    #define FXAA_QUALITY__P9 2.0
    #define FXAA_QUALITY__P10 4.0
    #define FXAA_QUALITY__P11 8.0
#endif

/*============================================================
==================
                    FXAA QUALITY - EXTREME QUALITY
============================================================
==================*/
#if (FXAA_QUALITY__PRESET == 39)
    #define FXAA_QUALITY__PS 12
    #define FXAA_QUALITY__P0 1.0
    #define FXAA_QUALITY__P1 1.0
    #define FXAA_QUALITY__P2 1.0
    #define FXAA_QUALITY__P3 1.0
    #define FXAA_QUALITY__P4 1.0
    #define FXAA_QUALITY__P5 1.5
    #define FXAA_QUALITY__P6 2.0
    #define FXAA_QUALITY__P7 2.0
    #define FXAA_QUALITY__P8 2.0
    #define FXAA_QUALITY__P9 2.0
    #define FXAA_QUALITY__P10 4.0
    #define FXAA_QUALITY__P11 8.0
#endif



/*============================================================
==================

                    API PORTING

============================================================
==================*/
#if (FXAA_GLSL_120 == 1) || (FXAA_GLSL_130 == 1)
    #define FxaaBool bool
    #define FxaaDiscard discard
    #define FxaaFloat float
    #define FxaaFloat2 vec2
    #define FxaaFloat3 vec3
    #define FxaaFloat4 vec4
    #define FxaaHalf float
    #define FxaaHalf2 vec2
```

```
    #define FxaaHalf3 vec3
    #define FxaaHalf4 vec4
    #define FxaaInt2 ivec2
    #define FxaaSat(x) clamp(x, 0.0, 1.0)
    #define FxaaTex sampler2D
#else
    #define FxaaBool bool
    #define FxaaDiscard clip(-1)
    #define FxaaFloat float
    #define FxaaFloat2 float2
    #define FxaaFloat3 float3
    #define FxaaFloat4 float4
    #define FxaaHalf half
    #define FxaaHalf2 half2
    #define FxaaHalf3 half3
    #define FxaaHalf4 half4
    #define FxaaSat(x) saturate(x)
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_GLSL_120 == 1)
    // Requires,
    //  #version 120
    // And at least,
    //  #extension GL_EXT_gpu_shader4 : enable
    //  (or set FXAA_FAST_PIXEL_OFFSET 1 to work like DX9)
    #define FxaaTexTop(t, p) texture2DLod(t, p, 0.0)
    #if (FXAA_FAST_PIXEL_OFFSET == 1)
        #define FxaaTexOff(t, p, o, r) texture2DLodOffset(t, p, 0.0, o)
    #else
        #define FxaaTexOff(t, p, o, r) texture2DLod(t, p + (o * r), 0.0)
    #endif
    #if (FXAA_GATHER4_ALPHA == 1)
        // use #extension GL_ARB_gpu_shader5 : enable
        #define FxaaTexAlpha4(t, p) textureGather(t, p, 3)
        #define FxaaTexOffAlpha4(t, p, o) textureGatherOffset(t, p, o, 3)
        #define FxaaTexGreen4(t, p) textureGather(t, p, 1)
        #define FxaaTexOffGreen4(t, p, o) textureGatherOffset(t, p, o, 1)
    #endif
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_GLSL_130 == 1)
    // Requires "#version 130" or better
    #define FxaaTexTop(t, p) textureLod(t, p, 0.0)
    #define FxaaTexOff(t, p, o, r) textureLodOffset(t, p, 0.0, o)
```

```
    #if (FXAA_GATHER4_ALPHA == 1)
        // use #extension GL_ARB_gpu_shader5 : enable
        #define FxaaTexAlpha4(t, p) textureGather(t, p, 3)
        #define FxaaTexOffAlpha4(t, p, o) textureGatherOffset(t, p, o, 3)
        #define FxaaTexGreen4(t, p) textureGather(t, p, 1)
        #define FxaaTexOffGreen4(t, p, o) textureGatherOffset(t, p, o, 1)
    #endif
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_HLSL_3 == 1) || (FXAA_360 == 1) || (FXAA_PS3 == 1)
    #define FxaaInt2 float2
    #define FxaaTex sampler2D
    #define FxaaTexTop(t, p) tex2Dlod(t, float4(p, 0.0, 0.0))
    #define FxaaTexOff(t, p, o, r) tex2Dlod(t, float4(p + (o * r), 0, 0))
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_HLSL_4 == 1)
    #define FxaaInt2 int2
    struct FxaaTex { SamplerState smpl; Texture2D tex; };
    #define FxaaTexTop(t, p) t.tex.SampleLevel(t.smpl, p, 0.0)
    #define FxaaTexOff(t, p, o, r) t.tex.SampleLevel(t.smpl, p, 0.0, o)
#endif
/*--------------------------------------------------------------------------*/
#if (FXAA_HLSL_5 == 1)
    #define FxaaInt2 int2
    struct FxaaTex { SamplerState smpl; Texture2D tex; };
    #define FxaaTexTop(t, p) t.tex.SampleLevel(t.smpl, p, 0.0)
    #define FxaaTexOff(t, p, o, r) t.tex.SampleLevel(t.smpl, p, 0.0, o)
    #define FxaaTexAlpha4(t, p) t.tex.GatherAlpha(t.smpl, p)
    #define FxaaTexOffAlpha4(t, p, o) t.tex.GatherAlpha(t.smpl, p, o)
    #define FxaaTexGreen4(t, p) t.tex.GatherGreen(t.smpl, p)
    #define FxaaTexOffGreen4(t, p, o) t.tex.GatherGreen(t.smpl, p, o)
#endif


/*==========================================================================
==================
                        GREEN AS LUMA OPTION SUPPORT FUNCTION
============================================================================
================*/
#if (FXAA_GREEN_AS_LUMA == 0)
    FxaaFloat FxaaLuma(FxaaFloat4 rgba) { return rgba.w; }
#else
    FxaaFloat FxaaLuma(FxaaFloat4 rgba) { return rgba.y; }
```

```
#endif


/*============================================================
==================

                        FXAA3 QUALITY - PC

============================================================
==================*/
#if (FXAA_PC == 1)
/*--------------------------------------------------------------------------*/
FxaaFloat4 FxaaPixelShader(
    //
    // Use noperspective interpolation here (turn off perspective interpolation).
    // {xy} = center of pixel
    FxaaFloat2 pos,
    //
    // Used only for FXAA Console, and not used on the 360 version.
    // Use noperspective interpolation here (turn off perspective interpolation).
    // {xy__} = upper left of pixel
    // {__zw} = lower right of pixel
    FxaaFloat4 fxaaConsolePosPos,
    //
    // Input color texture.
    // {rgb_} = color in linear or perceptual color space
    // if (FXAA_GREEN_AS_LUMA == 0)
    //     {___a} = luma in perceptual color space (not linear)
    FxaaTex tex,
    //
    // Only used on the optimized 360 version of FXAA Console.
    // For everything but 360, just use the same input here as for "tex".
    // For 360, same texture, just alias with a 2nd sampler.
    // This sampler needs to have an exponent bias of -1.
    FxaaTex fxaaConsole360TexExpBiasNegOne,
    //
    // Only used on the optimized 360 version of FXAA Console.
    // For everything but 360, just use the same input here as for "tex".
    // For 360, same texture, just alias with a 3nd sampler.
    // This sampler needs to have an exponent bias of -2.
    FxaaTex fxaaConsole360TexExpBiasNegTwo,
    //
```

```
// Only used on FXAA Quality.
// This must be from a constant/uniform.
// {x_} = 1.0/screenWidthInPixels
// {_y} = 1.0/screenHeightInPixels
FxaaFloat2 fxaaQualityRcpFrame,
//
// Only used on FXAA Console.
// This must be from a constant/uniform.
// This effects sub-pixel AA quality and inversely sharpness.
//     Where N ranges between,
//         N = 0.50 (default)
//         N = 0.33 (sharper)
// {x___} = -N/screenWidthInPixels
// {_y__} = -N/screenHeightInPixels
// {__z_} =    N/screenWidthInPixels
// {___w} =    N/screenHeightInPixels
FxaaFloat4 fxaaConsoleRcpFrameOpt,
//
// Only used on FXAA Console.
// Not used on 360, but used on PS3 and PC.
// This must be from a constant/uniform.
// {x___} = -2.0/screenWidthInPixels
// {_y__} = -2.0/screenHeightInPixels
// {__z_} =    2.0/screenWidthInPixels
// {___w} =    2.0/screenHeightInPixels
FxaaFloat4 fxaaConsoleRcpFrameOpt2,
//
// Only used on FXAA Console.
// Only used on 360 in place of fxaaConsoleRcpFrameOpt2.
// This must be from a constant/uniform.
// {x___} =    8.0/screenWidthInPixels
// {_y__} =    8.0/screenHeightInPixels
// {__z_} = -4.0/screenWidthInPixels
// {___w} = -4.0/screenHeightInPixels
FxaaFloat4 fxaaConsole360RcpFrameOpt2,
//
// Only used on FXAA Quality.
// This used to be the FXAA_QUALITY__SUBPIX define.
// It is here now to allow easier tuning.
// Choose the amount of sub-pixel aliasing removal.
// This can effect sharpness.
//     1.00 - upper limit (softer)
//     0.75 - default amount of filtering
//     0.50 - lower limit (sharper, less sub-pixel aliasing removal)
```

```
//     0.25 - almost off
//     0.00 - completely off
FxaaFloat fxaaQualitySubpix,
//
// Only used on FXAA Quality.
// This used to be the FXAA_QUALITY__EDGE_THRESHOLD define.
// It is here now to allow easier tuning.
// The minimum amount of local contrast required to apply algorithm.
//     0.333 - too little (faster)
//     0.250 - low quality
//     0.166 - default
//     0.125 - high quality
//     0.063 - overkill (slower)
FxaaFloat fxaaQualityEdgeThreshold,
//
// Only used on FXAA Quality.
// This used to be the FXAA_QUALITY__EDGE_THRESHOLD_MIN define.
// It is here now to allow easier tuning.
// Trims the algorithm from processing darks.
//     0.0833 - upper limit (default, the start of visible unfiltered edges)
//     0.0625 - high quality (faster)
//     0.0312 - visible limit (slower)
// Special notes when using FXAA_GREEN_AS_LUMA,
//     Likely want to set this to zero.
//     As colors that are mostly not-green
//     will appear very dark in the green channel!
//     Tune by looking at mostly non-green content,
//     then start at zero and increase until aliasing is a problem.
FxaaFloat fxaaQualityEdgeThresholdMin,
//
// Only used on FXAA Console.
// This used to be the FXAA_CONSOLE__EDGE_SHARPNESS define.
// It is here now to allow easier tuning.
// This does not effect PS3, as this needs to be compiled in.
//     Use FXAA_CONSOLE__PS3_EDGE_SHARPNESS for PS3.
//     Due to the PS3 being ALU bound,
//     there are only three safe values here: 2 and 4 and 8.
//     These options use the shaders ability to a free *|/ by 2|4|8.
// For all other platforms can be a non-power of two.
//     8.0 is sharper (default!!!)
//     4.0 is softer
//     2.0 is really soft (good only for vector graphics inputs)
FxaaFloat fxaaConsoleEdgeSharpness,
//
```

```
    // Only used on FXAA Console.
    // This used to be the FXAA_CONSOLE__EDGE_THRESHOLD define.
    // It is here now to allow easier tuning.
    // This does not effect PS3, as this needs to be compiled in.
    //     Use FXAA_CONSOLE__PS3_EDGE_THRESHOLD for PS3.
    //     Due to the PS3 being ALU bound,
    //     there are only two safe values here: 1/4 and 1/8.
    //     These options use the shaders ability to a free *|/ by 2|4|8.
    // The console setting has a different mapping than the quality setting.
    // Other platforms can use other values.
    //     0.125 leaves less aliasing, but is softer (default!!!)
    //     0.25 leaves more aliasing, and is sharper
    FxaaFloat fxaaConsoleEdgeThreshold,
    //
    // Only used on FXAA Console.
    // This used to be the FXAA_CONSOLE__EDGE_THRESHOLD_MIN define.
    // It is here now to allow easier tuning.
    // Trims the algorithm from processing darks.
    // The console setting has a different mapping than the quality setting.
    // This only applies when FXAA_EARLY_EXIT is 1.
    // This does not apply to PS3,
    // PS3 was simplified to avoid more shader instructions.
    //     0.06 - faster but more aliasing in darks
    //     0.05 - default
    //     0.04 - slower and less aliasing in darks
    // Special notes when using FXAA_GREEN_AS_LUMA,
    //     Likely want to set this to zero.
    //     As colors that are mostly not-green
    //     will appear very dark in the green channel!
    //     Tune by looking at mostly non-green content,
    //     then start at zero and increase until aliasing is a problem.
    FxaaFloat fxaaConsoleEdgeThresholdMin,
    //
    // Extra constants for 360 FXAA Console only.
    // Use zeros or anything else for other platforms.
    // These must be in physical constant registers and NOT immedates.
    // Immedates will result in compiler un-optimizing.
    // {xyzw} = float4(1.0, -1.0, 0.25, -0.25)
    FxaaFloat4 fxaaConsole360ConstDir
) {
/*--------------------------------------------------------------------------*/
    FxaaFloat2 posM;
    posM.x = pos.x;
    posM.y = pos.y;
```

```
    #if (FXAA_GATHER4_ALPHA == 1)
        #if (FXAA_DISCARD == 0)
            FxaaFloat4 rgbyM = FxaaTexTop(tex, posM);
            #if (FXAA_GREEN_AS_LUMA == 0)
                #define lumaM rgbyM.w
            #else
                #define lumaM rgbyM.y
            #endif
        #endif
        #if (FXAA_GREEN_AS_LUMA == 0)
            FxaaFloat4 luma4A = FxaaTexAlpha4(tex, posM);
            FxaaFloat4 luma4B = FxaaTexOffAlpha4(tex, posM, FxaaInt2(-1, -1));
        #else
            FxaaFloat4 luma4A = FxaaTexGreen4(tex, posM);
            FxaaFloat4 luma4B = FxaaTexOffGreen4(tex, posM, FxaaInt2(-1, -1));
        #endif
        #if (FXAA_DISCARD == 1)
            #define lumaM luma4A.w
        #endif
        #define lumaE luma4A.z
        #define lumaS luma4A.x
        #define lumaSE luma4A.y
        #define lumaNW luma4B.w
        #define lumaN luma4B.z
        #define lumaW luma4B.x
    #else
        FxaaFloat4 rgbyM = FxaaTexTop(tex, posM);
        #if (FXAA_GREEN_AS_LUMA == 0)
            #define lumaM rgbyM.w
        #else
            #define lumaM rgbyM.y
        #endif
        FxaaFloat lumaS = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 0, 1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaE = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 1, 0),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaN = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 0,-1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaW = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(-1, 0),
fxaaQualityRcpFrame.xy));
    #endif
/*--------------------------------------------------------------------------*/
    FxaaFloat maxSM = max(lumaS, lumaM);
    FxaaFloat minSM = min(lumaS, lumaM);
```

```
    FxaaFloat maxESM = max(lumaE, maxSM);
    FxaaFloat minESM = min(lumaE, minSM);
    FxaaFloat maxWN = max(lumaN, lumaW);
    FxaaFloat minWN = min(lumaN, lumaW);
    FxaaFloat rangeMax = max(maxWN, maxESM);
    FxaaFloat rangeMin = min(minWN, minESM);
    FxaaFloat rangeMaxScaled = rangeMax * fxaaQualityEdgeThreshold;
    FxaaFloat range = rangeMax - rangeMin;
    FxaaFloat rangeMaxClamped = max(fxaaQualityEdgeThresholdMin, rangeMaxScaled);
    FxaaBool earlyExit = range < rangeMaxClamped;
/*------------------------------------------------------------------------*/
    if(earlyExit)
        #if (FXAA_DISCARD == 1)
            FxaaDiscard;
        #else
            return rgbyM;
        #endif
/*------------------------------------------------------------------------*/
    #if (FXAA_GATHER4_ALPHA == 0)
        FxaaFloat lumaNW = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(-1,-1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaSE = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 1, 1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaNE = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2( 1,-1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaSW = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(-1, 1),
fxaaQualityRcpFrame.xy));
    #else
        FxaaFloat lumaNE = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(1, -1),
fxaaQualityRcpFrame.xy));
        FxaaFloat lumaSW = FxaaLuma(FxaaTexOff(tex, posM, FxaaInt2(-1, 1),
fxaaQualityRcpFrame.xy));
    #endif
/*------------------------------------------------------------------------*/
    FxaaFloat lumaNS = lumaN + lumaS;
    FxaaFloat lumaWE = lumaW + lumaE;
    FxaaFloat subpixRcpRange = 1.0/range;
    FxaaFloat subpixNSWE = lumaNS + lumaWE;
    FxaaFloat edgeHorz1 = (-2.0 * lumaM) + lumaNS;
    FxaaFloat edgeVert1 = (-2.0 * lumaM) + lumaWE;
/*------------------------------------------------------------------------*/
    FxaaFloat lumaNESE = lumaNE + lumaSE;
    FxaaFloat lumaNWNE = lumaNW + lumaNE;
    FxaaFloat edgeHorz2 = (-2.0 * lumaE) + lumaNESE;
```

```
    FxaaFloat edgeVert2 = (-2.0 * lumaN) + lumaNWNE;
/*--------------------------------------------------------------------------*/
    FxaaFloat lumaNWSW = lumaNW + lumaSW;
    FxaaFloat lumaSWSE = lumaSW + lumaSE;
    FxaaFloat edgeHorz4 = (abs(edgeHorz1) * 2.0) + abs(edgeHorz2);
    FxaaFloat edgeVert4 = (abs(edgeVert1) * 2.0) + abs(edgeVert2);
    FxaaFloat edgeHorz3 = (-2.0 * lumaW) + lumaNWSW;
    FxaaFloat edgeVert3 = (-2.0 * lumaS) + lumaSWSE;
    FxaaFloat edgeHorz = abs(edgeHorz3) + edgeHorz4;
    FxaaFloat edgeVert = abs(edgeVert3) + edgeVert4;
/*--------------------------------------------------------------------------*/
    FxaaFloat subpixNWSWNESE = lumaNWSW + lumaNESE;
    FxaaFloat lengthSign = fxaaQualityRcpFrame.x;
    FxaaBool horzSpan = edgeHorz >= edgeVert;
    FxaaFloat subpixA = subpixNSWE * 2.0 + subpixNWSWNESE;
/*--------------------------------------------------------------------------*/
    if(!horzSpan) lumaN = lumaW;
    if(!horzSpan) lumaS = lumaE;
    if(horzSpan) lengthSign = fxaaQualityRcpFrame.y;
    FxaaFloat subpixB = (subpixA * (1.0/12.0)) - lumaM;
/*--------------------------------------------------------------------------*/
    FxaaFloat gradientN = lumaN - lumaM;
    FxaaFloat gradientS = lumaS - lumaM;
    FxaaFloat lumaNN = lumaN + lumaM;
    FxaaFloat lumaSS = lumaS + lumaM;
    FxaaBool pairN = abs(gradientN) >= abs(gradientS);
    FxaaFloat gradient = max(abs(gradientN), abs(gradientS));
    if(pairN) lengthSign = -lengthSign;
    FxaaFloat subpixC = FxaaSat(abs(subpixB) * subpixRcpRange);
/*--------------------------------------------------------------------------*/
    FxaaFloat2 posB;
    posB.x = posM.x;
    posB.y = posM.y;
    FxaaFloat2 offNP;
    offNP.x = (!horzSpan) ? 0.0 : fxaaQualityRcpFrame.x;
    offNP.y = ( horzSpan) ? 0.0 : fxaaQualityRcpFrame.y;
    if(!horzSpan) posB.x += lengthSign * 0.5;
    if( horzSpan) posB.y += lengthSign * 0.5;
/*--------------------------------------------------------------------------*/
    FxaaFloat2 posN;
    posN.x = posB.x - offNP.x * FXAA_QUALITY__P0;
    posN.y = posB.y - offNP.y * FXAA_QUALITY__P0;
    FxaaFloat2 posP;
    posP.x = posB.x + offNP.x * FXAA_QUALITY__P0;
```

```
    posP.y = posB.y + offNP.y * FXAA_QUALITY__P0;
    FxaaFloat subpixD = ((-2.0)*subpixC) + 3.0;
    FxaaFloat lumaEndN = FxaaLuma(FxaaTexTop(tex, posN));
    FxaaFloat subpixE = subpixC * subpixC;
    FxaaFloat lumaEndP = FxaaLuma(FxaaTexTop(tex, posP));
/*---------------------------------------------------------------------------*/
    if(!pairN) lumaNN = lumaSS;
    FxaaFloat gradientScaled = gradient * 1.0/4.0;
    FxaaFloat lumaMM = lumaM - lumaNN * 0.5;
    FxaaFloat subpixF = subpixD * subpixE;
    FxaaBool lumaMLTZero = lumaMM < 0.0;
/*---------------------------------------------------------------------------*/
    lumaEndN -= lumaNN * 0.5;
    lumaEndP -= lumaNN * 0.5;
    FxaaBool doneN = abs(lumaEndN) >= gradientScaled;
    FxaaBool doneP = abs(lumaEndP) >= gradientScaled;
    if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P1;
    if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P1;
    FxaaBool doneNP = (!doneN) || (!doneP);
    if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P1;
    if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P1;
/*---------------------------------------------------------------------------*/
    if(doneNP) {
        if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
        if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
        if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
        if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
        doneN = abs(lumaEndN) >= gradientScaled;
        doneP = abs(lumaEndP) >= gradientScaled;
        if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P2;
        if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P2;
        doneNP = (!doneN) || (!doneP);
        if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P2;
        if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P2;
/*---------------------------------------------------------------------------*/
        #if (FXAA_QUALITY__PS > 3)
        if(doneNP) {
            if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
            if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
            if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
            if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
            doneN = abs(lumaEndN) >= gradientScaled;
            doneP = abs(lumaEndP) >= gradientScaled;
            if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P3;
```

```
            if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P3;
            doneNP = (!doneN) || (!doneP);
            if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P3;
            if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P3;
/*--------------------------------------------------------------------------*/
            #if (FXAA_QUALITY__PS > 4)
            if(doneNP) {
                if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                doneN = abs(lumaEndN) >= gradientScaled;
                doneP = abs(lumaEndP) >= gradientScaled;
                if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P4;
                if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P4;
                doneNP = (!doneN) || (!doneP);
                if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P4;
                if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P4;
/*--------------------------------------------------------------------------*/
                #if (FXAA_QUALITY__PS > 5)
                if(doneNP) {
                    if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                    if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                    if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                    if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                    doneN = abs(lumaEndN) >= gradientScaled;
                    doneP = abs(lumaEndP) >= gradientScaled;
                    if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P5;
                    if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P5;
                    doneNP = (!doneN) || (!doneP);
                    if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P5;
                    if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P5;
/*--------------------------------------------------------------------------*/
                    #if (FXAA_QUALITY__PS > 6)
                    if(doneNP) {
                        if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                        if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                        if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                        if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                        doneN = abs(lumaEndN) >= gradientScaled;
                        doneP = abs(lumaEndP) >= gradientScaled;
                        if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P6;
                        if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P6;
                        doneNP = (!doneN) || (!doneP);
```

```
                            if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P6;
                            if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P6;
/*--------------------------------------------------------------------------*/
                            #if (FXAA_QUALITY__PS > 7)
                            if(doneNP) {
                                if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex,
posN.xy));
                                if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex,
posP.xy));
                                if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                                if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                                doneN = abs(lumaEndN) >= gradientScaled;
                                doneP = abs(lumaEndP) >= gradientScaled;
                                if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P7;
                                if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P7;
                                doneNP = (!doneN) || (!doneP);
                                if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P7;
                                if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P7;
/*--------------------------------------------------------------------------*/
    #if (FXAA_QUALITY__PS > 8)
    if(doneNP) {
        if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
        if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
        if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
        if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
        doneN = abs(lumaEndN) >= gradientScaled;
        doneP = abs(lumaEndP) >= gradientScaled;
        if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P8;
        if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P8;
        doneNP = (!doneN) || (!doneP);
        if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P8;
        if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P8;
/*--------------------------------------------------------------------------*/
        #if (FXAA_QUALITY__PS > 9)
        if(doneNP) {
            if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
            if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
            if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
            if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
            doneN = abs(lumaEndN) >= gradientScaled;
            doneP = abs(lumaEndP) >= gradientScaled;
            if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P9;
            if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P9;
            doneNP = (!doneN) || (!doneP);
```

```
            if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P9;
            if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P9;
/*--------------------------------------------------------------------------*/
            #if (FXAA_QUALITY__PS > 10)
            if(doneNP) {
                if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                doneN = abs(lumaEndN) >= gradientScaled;
                doneP = abs(lumaEndP) >= gradientScaled;
                if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P10;
                if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P10;
                doneNP = (!doneN) || (!doneP);
                if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P10;
                if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P10;
/*--------------------------------------------------------------------------*/
                #if (FXAA_QUALITY__PS > 11)
                if(doneNP) {
                    if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                    if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                    if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                    if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                    doneN = abs(lumaEndN) >= gradientScaled;
                    doneP = abs(lumaEndP) >= gradientScaled;
                    if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P11;
                    if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P11;
                    doneNP = (!doneN) || (!doneP);
                    if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P11;
                    if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P11;
/*--------------------------------------------------------------------------*/
                    #if (FXAA_QUALITY__PS > 12)
                    if(doneNP) {
                        if(!doneN) lumaEndN = FxaaLuma(FxaaTexTop(tex, posN.xy));
                        if(!doneP) lumaEndP = FxaaLuma(FxaaTexTop(tex, posP.xy));
                        if(!doneN) lumaEndN = lumaEndN - lumaNN * 0.5;
                        if(!doneP) lumaEndP = lumaEndP - lumaNN * 0.5;
                        doneN = abs(lumaEndN) >= gradientScaled;
                        doneP = abs(lumaEndP) >= gradientScaled;
                        if(!doneN) posN.x -= offNP.x * FXAA_QUALITY__P12;
                        if(!doneN) posN.y -= offNP.y * FXAA_QUALITY__P12;
                        doneNP = (!doneN) || (!doneP);
                        if(!doneP) posP.x += offNP.x * FXAA_QUALITY__P12;
                        if(!doneP) posP.y += offNP.y * FXAA_QUALITY__P12;
```

```
/*--------------------------------------------------------------------------*/
                        }
                        #endif
/*--------------------------------------------------------------------------*/
                    }
                    #endif
/*--------------------------------------------------------------------------*/
                }
                #endif
/*--------------------------------------------------------------------------*/
            }
            #endif
/*--------------------------------------------------------------------------*/
        }
        #endif
/*--------------------------------------------------------------------------*/
    }
    #endif
/*--------------------------------------------------------------------------*/
                            }
                            #endif
/*--------------------------------------------------------------------------*/
                        }
                        #endif
/*--------------------------------------------------------------------------*/
                    }
                    #endif
/*--------------------------------------------------------------------------*/
                }
                #endif
/*--------------------------------------------------------------------------*/
            }
            #endif
/*--------------------------------------------------------------------------*/
    }
/*--------------------------------------------------------------------------*/
    FxaaFloat dstN = posM.x - posN.x;
    FxaaFloat dstP = posP.x - posM.x;
    if(!horzSpan) dstN = posM.y - posN.y;
    if(!horzSpan) dstP = posP.y - posM.y;
/*--------------------------------------------------------------------------*/
    FxaaBool goodSpanN = (lumaEndN < 0.0) != lumaMLTZero;
    FxaaFloat spanLength = (dstP + dstN);
    FxaaBool goodSpanP = (lumaEndP < 0.0) != lumaMLTZero;
    FxaaFloat spanLengthRcp = 1.0/spanLength;
/*--------------------------------------------------------------------------*/
    FxaaBool directionN = dstN < dstP;
```

```
    FxaaFloat dst = min(dstN, dstP);
    FxaaBool goodSpan = directionN ? goodSpanN : goodSpanP;
    FxaaFloat subpixG = subpixF * subpixF;
    FxaaFloat pixelOffset = (dst * (-spanLengthRcp)) + 0.5;
    FxaaFloat subpixH = subpixG * fxaaQualitySubpix;
/*--------------------------------------------------------------------------*/
    FxaaFloat pixelOffsetGood = goodSpan ? pixelOffset : 0.0;
    FxaaFloat pixelOffsetSubpix = max(pixelOffsetGood, subpixH);
    if(!horzSpan) posM.x += pixelOffsetSubpix * lengthSign;
    if( horzSpan) posM.y += pixelOffsetSubpix * lengthSign;
    #if (FXAA_DISCARD == 1)
        return FxaaTexTop(tex, posM);
    #else
        return FxaaFloat4(FxaaTexTop(tex, posM).xyz, lumaM);
    #endif
}
/*==========================================================================
================*/
#endif




//-----------------------------------------------------------------------------
-----
// File:            es3-kepler\FXAA\assets\shaders/FXAA_High_Quality.frag
// SDK Version: v2.0
// Email:           gameworks@nvidia.com
// Site:            http://developer.nvidia.com/
//
// Copyright (c) 2014, NVIDIA CORPORATION. All rights reserved.
//
// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions
// are met:
//  * Redistributions of source code must retain the above copyright
//    notice, this list of conditions and the following disclaimer.
//  * Redistributions in binary form must reproduce the above copyright
//    notice, this list of conditions and the following disclaimer in the
//    documentation and/or other materials provided with the distribution.
//  * Neither the name of NVIDIA CORPORATION nor the names of its
//    contributors may be used to endorse or promote products derived
//    from this software without specific prior written permission.
//
```

```
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS ``AS IS'' AND ANY
// EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
// IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
// PURPOSE ARE DISCLAIMED.    IN NO EVENT SHALL THE COPYRIGHT OWNER OR
// CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
// EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
// PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
// PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
// OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
//------------------------------------------------------------------------
-----
//#version 100

precision highp float;

uniform sampler2D uSourceTex;
uniform vec2 RCPFrame;
varying vec2 vTexCoord;

void main(void)
{
    gl_FragColor = FxaaPixelShader(vTexCoord,
                    FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f),     // FxaaFloat4 fxaaConsolePosPos,
        uSourceTex,                                 // FxaaTex tex,
        uSourceTex,                                 // FxaaTex
fxaaConsole360TexExpBiasNegOne,
        uSourceTex,                                 // FxaaTex
fxaaConsole360TexExpBiasNegTwo,
        RCPFrame,                                   // FxaaFloat2 fxaaQualityRcpFrame,
        FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f),     // FxaaFloat4 fxaaConsoleRcpFrameOpt,
        FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f),     // FxaaFloat4 fxaaConsoleRcpFrameOpt2,
        FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f),     // FxaaFloat4 fxaaConsole360RcpFrameOpt2,
        0.75f,                                          // FxaaFloat fxaaQualitySubpix,
        0.166f,                                         // FxaaFloat
fxaaQualityEdgeThreshold,
        0.0833f,                                    // FxaaFloat
fxaaQualityEdgeThresholdMin,
        0.0f,                                       // FxaaFloat fxaaConsoleEdgeSharpness,
        0.0f,                                       // FxaaFloat fxaaConsoleEdgeThreshold,
        0.0f,                                       // FxaaFloat
fxaaConsoleEdgeThresholdMin,
```

```
        FxaaFloat4(0.0f, 0.0f, 0.0f, 0.0f)        // FxaaFloat fxaaConsole360ConstDir,
    );
}
```