Technical Report

Coverage Sampled Antialiasing

DEVELOPMENT

Peter Young
pyoung@nvidia.com

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050

October 30, 2006

Introduction

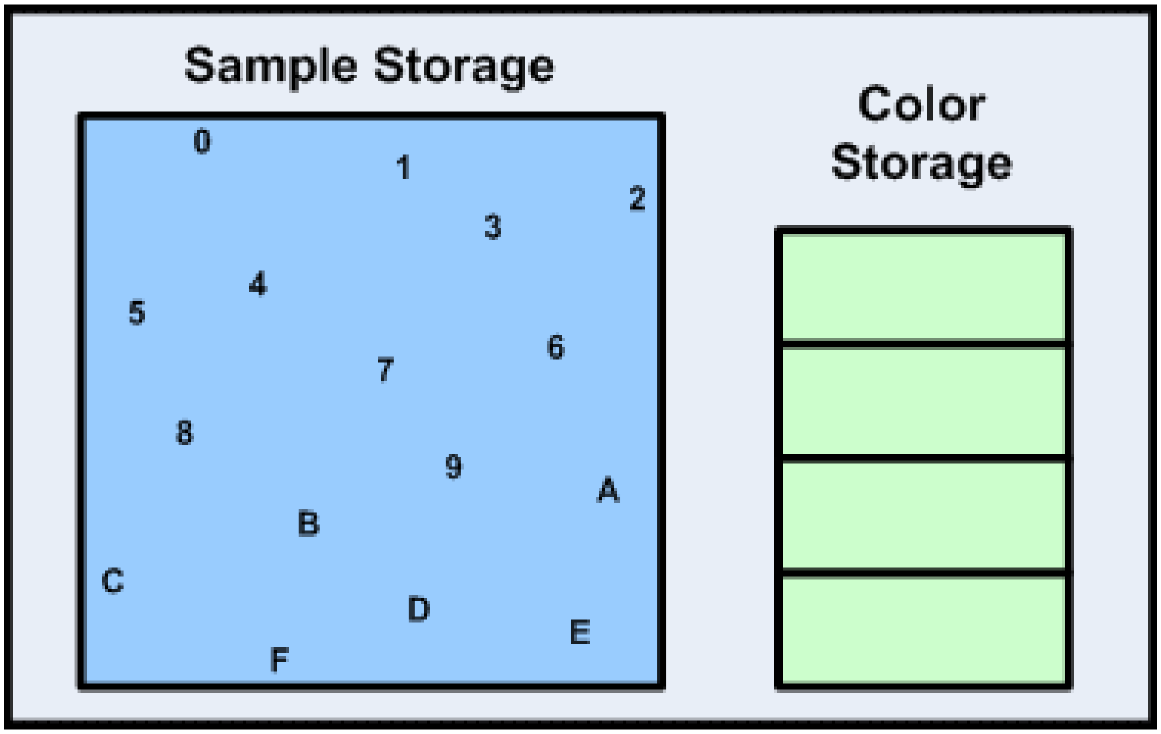
CSAA (Coverage Sampling Antialiasing) is one of the key new features of the GeForce 8 series GPU.

In summary, CSAA produces antialiased images that rival the quality of 8x or 16x MSAA, while introducing only a minimal performance hit over standard (typically 4x) MSAA. It works by introducing the concept of a new sample type: a sample that represents coverage. This differs from previous AA techniques where coverage was always inherently tied to another sample type. In supersampling for example, each sample represents shaded color, stored color/z/stencil, and coverage, which essentially amounts to rendering to an oversized buffer and downfiltering. MSAA reduces the shader overhead of this operation by decoupling shaded samples from stored color and coverage; this allows applications using antialiasing to operate with fewer shaded samples while maintaining the same quality color/z/stencil and coverage sampling. CSAA further optimizes this process by decoupling coverage from color/z/stencil, thus reducing bandwidth and storage costs.

On the GeForce 8800, the following CSAA modes are supported:

CSAA Mode	# of Color/Z/Stencil Samples	# of Coverage Samples
8x	4	8
8xQ (Quality)	8	8
16x	4	16
16xQ (Quality)	8	16

Figure 1: Color and coverage storage/layout in 16x CSAA



CSAA can be enabled in one of the two following scenarios:

- When the application specifically enables CSAA.
- When CSAA is turned on in the control panel by the end user. The control panel offers users one of two options:
    1. "Enhance" mode: All application-created MSAA surfaces are converted to CSAA surfaces using the control panel-specified CSAA mode.
    2. "Override" mode: The control panel forces a CSAA-enabled backbuffer. Note that this mode only works for applications that render directly to the backbuffer.

It is recommended that applications enable CSAA directly. The process of implementing CSAA is incredibly simple (almost exactly the same as implementing MSAA), and the benefit to your users is enormous; they get much higher image quality at almost zero performance loss, without the fuss and hassle of manipulating the control panel.

Implementation – DirectX 10

Creating CSAA-compatible Surfaces

CSAA is invoked through the use of the Quality field in the multisample descriptor structure:

```
typedef struct DXGI_SAMPLE_DESC {
    UINT Count;
    UINT Quality;
} DXGI_SAMPLE_DESC, *LPDXGI_SAMPLE_DESC;
```

Since this structure already provides a mechanism to specify the number of stored color/z/stencil samples through the Count field, and the number of shaded color samples is always 1 in a multisampled buffer, we can specify the number of coverage samples by setting the Quality field to the desired value. Below is a table listing the various CSAA modes, and the appropriate Count/Quality values used to enable them:

Table 1: CSAA modes and corresponding sample/coverage counts for DX10

Desired CSAA Mode	***Count*** **(# Color/z/stencil samples)**	***QualityValue***
8x	**4**	**8**
8xQ (Quality)	**8**	**8**
16x	**4**	**16**
16xQ (Quality)	**8**	**16**

Determining Hardware Support for CSAA

Checking for CSAA support is simple:

- Check the vendorid of the current device to ensure it is NVIDIA hardware
- Use `CheckMultisampleQualityLevels()` to check the max quality level. Pass the desired color/z/stencil count as *SampleCount*, and if *pNumQualityLevels* is greater than the desired *Quality* value, then CSAA is supported.

For example, if you pass a *SampleCount* of 4, a *pNumQualityLevels* value of 9 or greater indicates that the 8x and 8xQ CSAA modes are available. A *pNumQualityLevels* value of 17 or greater indicates that the 16x and 16xQ CSAA modes are available.

Implementation – DirectX 9

Creating CSAA-compatible Surfaces

The method of implementing CSAA in DX9 differs from DX10. This is due to a limitation in the DX9 runtime, which prohibits the driver from exposing multisample quality values greater than 7. For this reason, instead of specifying the number of coverage samples with the q*uality* value, we simply set *quality* to a predetermined value which will be interpreted as a specific CSAA mode by the driver.

When creating a D3D device or render target, use the D3DMULTISAMPLE_TYPE parameter to specify the number of color/z/stencil samples, and use the *quality* parameter to set the CSAA mode. Refer to the following table:

Table 2: CSAA modes and corresponding sample/coverage counts for DX9

Desired CSAA Mode	***Multisample* Value**	***Quality Value***
8x	`D3DMULTISAMPLE_4_SAMPLES`	**2**
8xQ (Quality)	`D3DMULTISAMPLE_8_SAMPLES`	**0**
16x	`D3DMULTISAMPLE_4_SAMPLES`	**4**
16xQ (Quality)	`D3DMULTISAMPLE_8_SAMPLES`	**2**

Determining Hardware Support for CSAA

Checking for CSAA support is simple:

- Check the vendorid of the current device to ensure it is NVIDIA hardware
- Use `CheckDeviceMultisampleType()` to check the max quality level. Set the MultiSampleType parameter to reflect the desired number of color/z/stencil samples, and if *pQualityLevels* is greater than the desired *Quality* value, then CSAA is supported.

For example, if you pass a sample count of 4, a *pQualityLevels* value of 5 or greater indicates that both 8x and 16x CSAA modes are available.

Implementation - OpenGL

Implementing CSAA in OpenGL is done through a new extension:

`GL_NV_framebuffer_multisample_coverage`

When support for this extension is verified, you may use the following function to create CSAA render buffers:

```
void RenderbufferStorageMultisampleCoverageNV(
            enum target, sizei coverageSamples,
            sizei colorSamples, enum internalformat,
            sizei width, sizei height);
```

Use of this function is identical to the older **RenderbufferStorageMultisampleEXT()**; the only difference is the ability to specify distinct color/z/stencil sample counts and coverage sample counts. Like DX10, the OpenGL function to specify color samples and coverage samples takes exact sample counts as arguments. Refer to the table below:

CSAA Mode	*colorSamples* value	*coverageSamples* value
8x	4	8
8xQ (Quality)	8	8
16x	4	16
16xQ (Quality)	8	16

CSAA Guidelines

Performance

CSAA performance is generally very similar to that of typical MSAA performance, given the same color/z/stencil sample count. For example, both the 8x and 16x CSAA modes (which use 4 color/z/stencil samples) perform similarly or identical to 4x MSAA.

CSAA is also extremely efficient in terms of storage, as the coverage samples require very little memory.

Transparency Antialiasing

Transparency AA is 100% compatible with CSAA. However, keep in mind that transparency AA operates at a lower resolution; it works at color/stencil/z resolution instead of coverage resolution, since transparency AA requires the color/z/stencil sample information to work.

Development Considerations

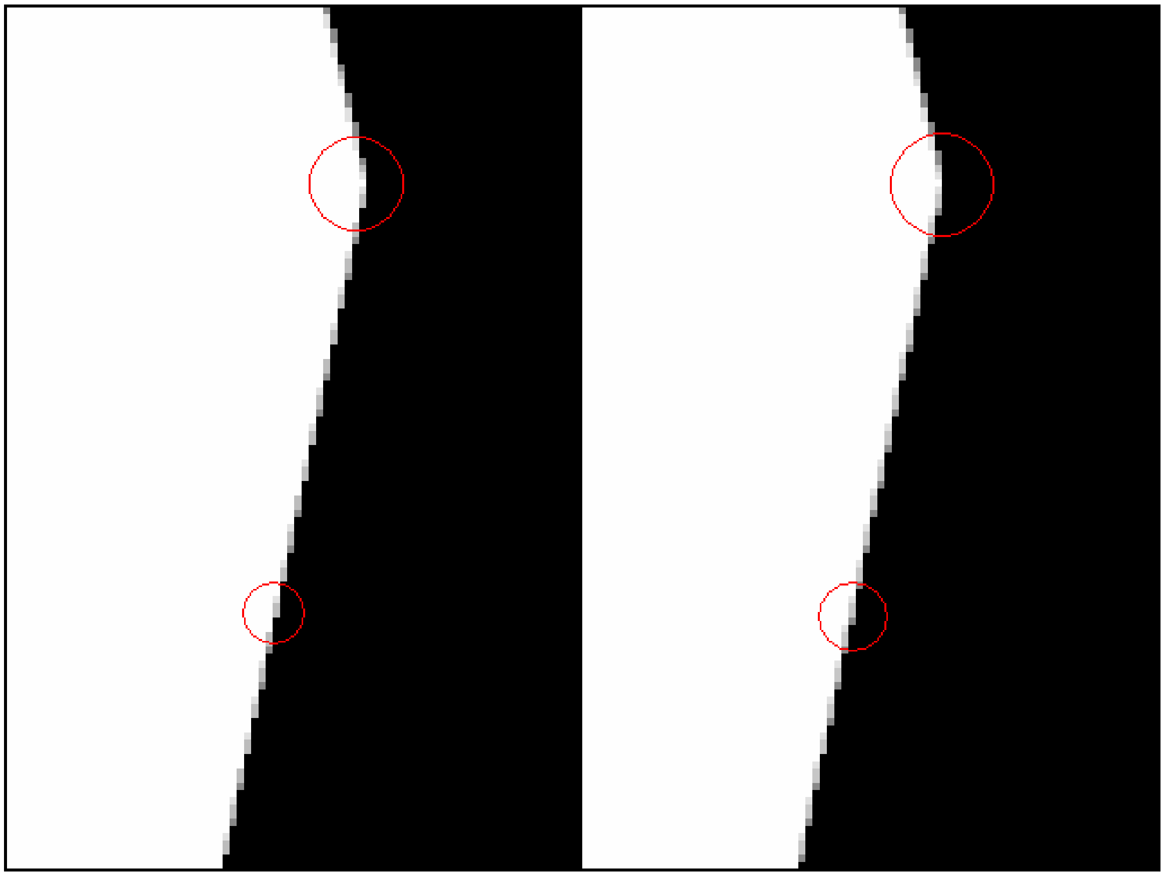
For best image quality results, try to avoid the following scenarios when implementing CSAA:

- Avoid sharing z-buffers between different multisample render targets. This can affect coverage information and potentially cause a loss in antialiasing quality. If your application renders to multiple render targets, the following loop for each RT is safe for CSAA:
    - Clear depth and color
    - Render to multisample target
    - Blit to non-multisample target
- Avoid partial depth-clears, as they can affect coverage information. If a triangle edge intersects the edge of the cleared region, antialiasing quality may be reduced at the point of intersection.
- Try to avoid updating color and z independently, as CSAA coverage information is only updated when z is being written. The exception to this is when the geometry for each pass is identical (such as in typical z-only prepasses), in which case image quality will be preserved.

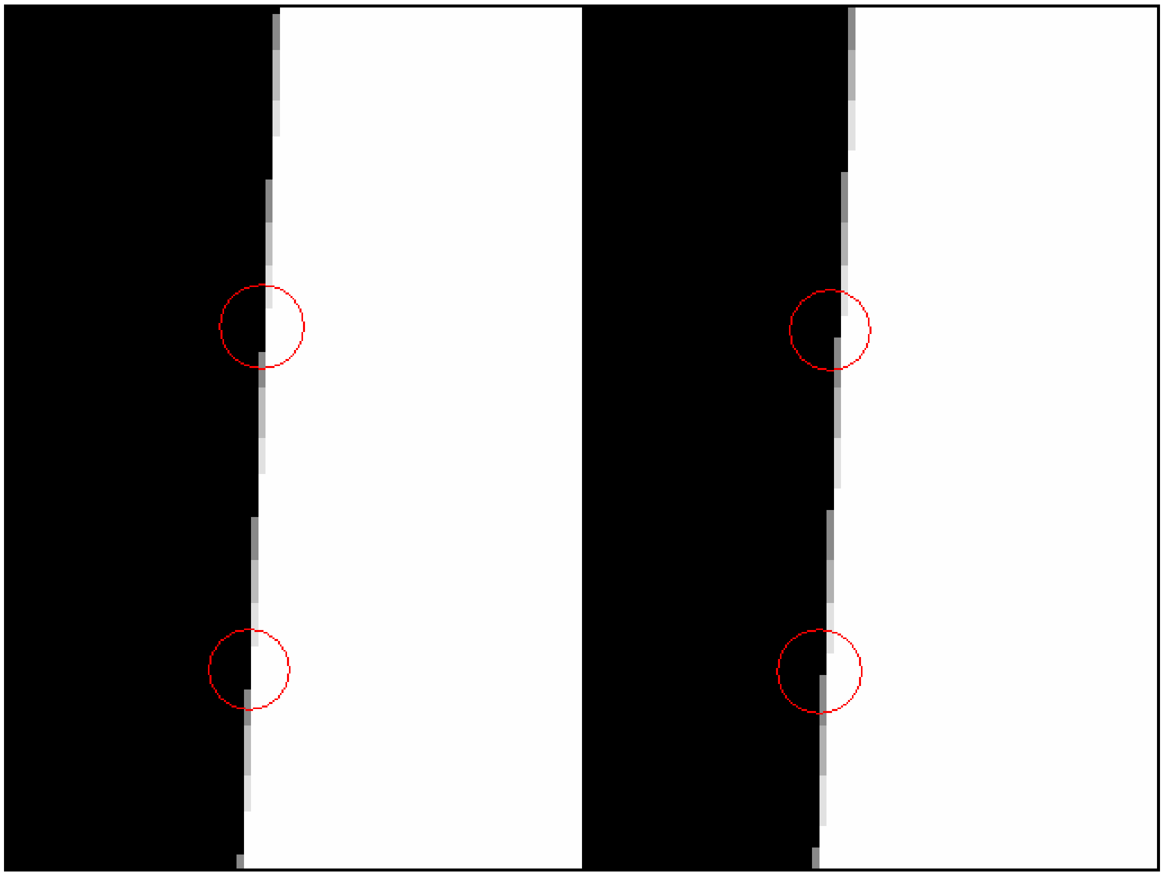
CSAA – Results

Below are images highlighting the difference in quality between standard 4x MSAA and 16x CSAA modes. Observe the improvement in image quality for 16x, and note the fact that 16x CSAA typically performs similarly to 4x MSAA.

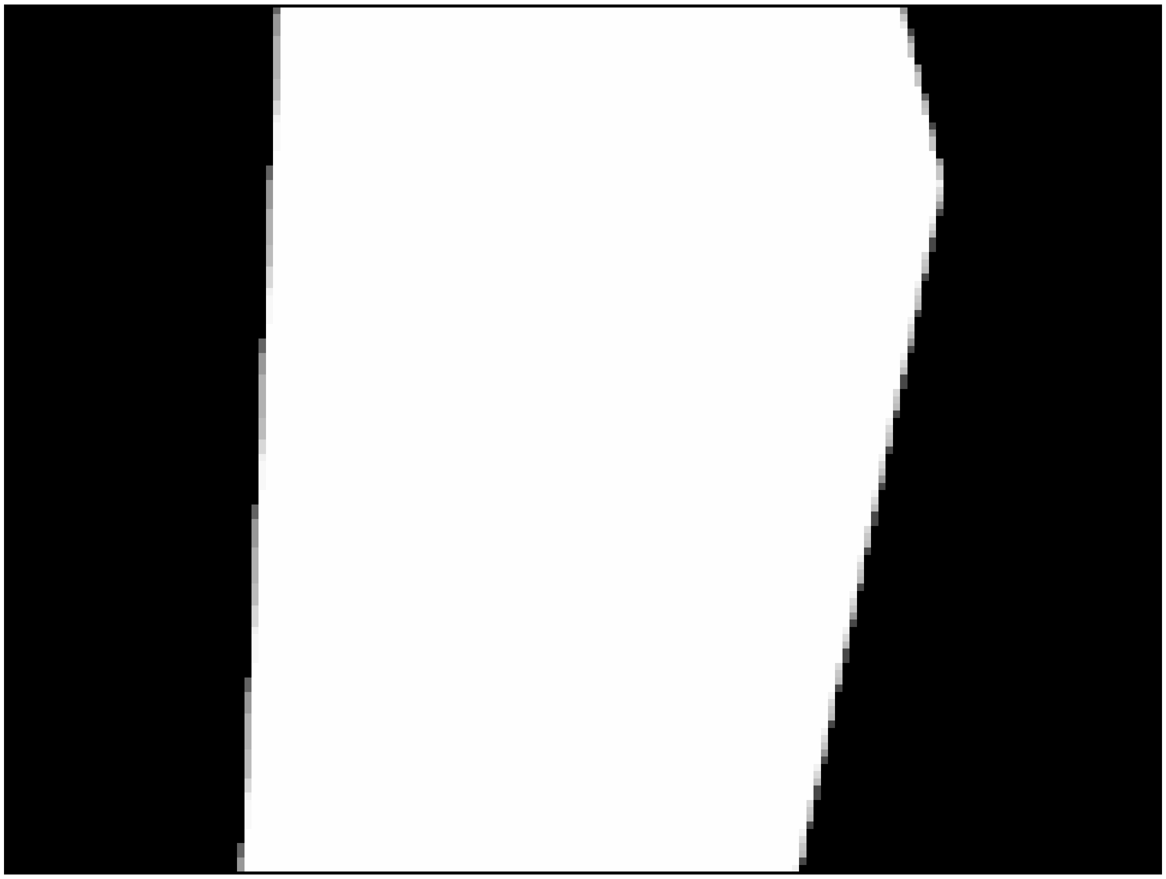
CSAA – 4x vs. 16x (note additional color samples in 16x)



CSAA – 4x vs 16x (note tighter edge resolution in 16x)



CSAA – 16xQ



NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com