Horizon-Based Ambient Occlusion using Compute Shaders

Louis Bavoil
lbavoil@nvidia.com

March 2011

Document Change History

Version	Date	Responsible	Reason for Change
1	March 14, 2011	Louis Bavoil	Initial release

Overview

This DirectX 11 SDK sample renders Screen-Space Ambient Occlusion (SSAO) in constant time using compute shaders. As with all SSAO algorithms, the input data is the depth buffer of the scene being rendered. (See [Akenine-Moller et al. 08] for an introduction to SSAO). In this sample, the AO is rendered using a jitter-free approximation of the Horizon-Based Ambient Occlusion (HBAO) algorithm [Bavoil and Sainz 08]. Second, the AO result is blurred horizontally and vertically using a depth-aware filter. Both the HBAO and the blur passes are implemented in compute shaders using group-shared memory. As shown on Figure 1, the lack of jittering in the HBAO computation can be hidden by using a large blur kernel. To ease the integration of this technique in DirectX 11 applications, an open-source library is provided with functions for specifying the input depths (linear or not), the associated camera parameters, the output color buffer, and the HBAO parameters.

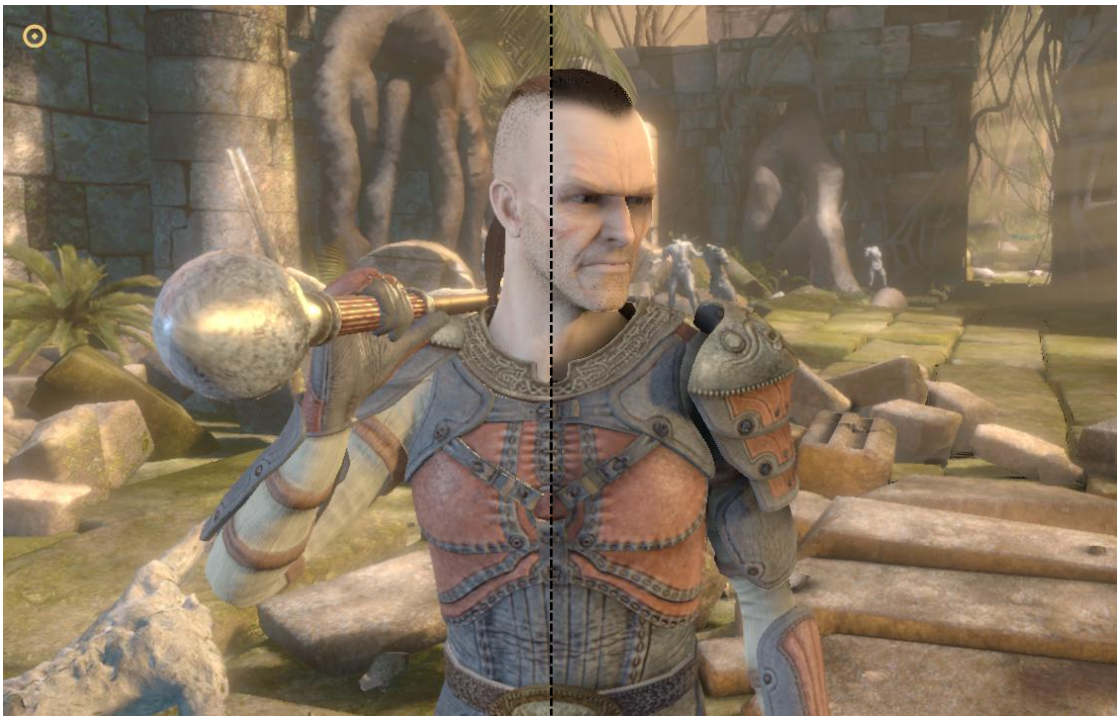

Figure 1. Medusa scene [NVIDIA 2008] with and without HBAO, using 4x8 depth samples per pixel and a 33x33 blur kernel.

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

The NVSSAO Library

Building with the Library

The NVSSAO library can be integrated as an external library into any DirectX 11 application. The header file and the 32-bit lib files are located in the NVSSAO directory. There are multiple lib files, for different build configurations. The naming conventions are:

- MT for multi-threaded runtime
- MD for multi-threaded DLL runtime
- MTd for multi-threaded debug runtime
- MDd for multi-threaded DLL debug runtime

Alternatively, the library can be integrated to the application by adding the contents of the NVSDK_D3D11_SSAO (including the Shaders subdirectory) to the source code and adding HBAO_DX11_LIB.cpp to the Visual Studio project. This is what the example SSAO11 application in this SDK sample does.

In the two cases, the interface of the library is NVSDK_D3D11_SSAO.h. Only this header needs to be included in the source files that call the library's functions.

Using NVSSAO

Adding HBAO to a DirectX 11 application can be done simply by adding two function calls to the rendering loop, after the opaque objects of the scene have been rendered into the hardware depth buffer, and before the semi-transparent geometry is blended over.

First, the input depth buffer must be specified by calling either `NVSDK_D3D11_SetHardwareDepthsForAO` or `NVSDK_D3D11_SetViewSpaceDepthsForAO`.

Second, calling `NVSDK_D3D11_RenderAO` will render HBAO using default parameters and will output the HBAO to the render target view passed as argument to the function. The RenderAO should not modify the state of the application. It saves all the D3D state objects that it will alter at the beginning of the function and restores them at the end.

The RenderAO function allocates all the D3D resources it needs on first use. To release these resources, `NVSDK_D3D11_ReleaseAOResources` should be added to the application where the screen-dependent resources are released. There is no need to call ReleaseAOResources when the buffers are resized though. RenderAO automatically resizes its render targets when it detects that the input depth buffer has changed size.

Optionally, before calling the `NVSDK_D3D11_RenderAO`, the AO parameters may be configured by calling `NVSDK_D3D11_SetAOParameters`. If the useBlending option is enabled, the AO is multiplied over the RGB colors from the output buffer while preserving destination alpha, otherwise the AO is overwritten.

Running the Sample

The "Show Colors" checkbox toggles drawing the scene colors. The "Show AO" checkbox toggles rendering the HBAO and multiplying the AO over the colors.

There are three scenes selectable from the GUI:

- The "Medusa" scenes are single-frame captures of the NVIDIA Medusa GTX 280 launch demo. For these scene, the camera cannot be moved and the resolution is assumed to be 1280x720 (default resolution).
- The "AT-AT" and the "Leaves" scenes are rendering meshes with a dynamic camera. For these scenes, the camera can be rotated by dragging the left button over the window and the mouse wheel can be used to zoom in and out. MSAA can be enabled by selecting one of the modes from the second combo box.
- The "Sibenik" scene is rendering a mesh with a first-person camera. For this scene, the camera can be moved with the A,D,W,S keys and dragging the mouse.

The following HBAO parameters are exposed in the GUI:

- The "Radius multiplier" slider controls the scale factor that is applied to the AO radius of influence (view-space distance). Increasing the AO radius increases the distance over which objects are casting occlusion.
- The "Num steps" slider controls the number of uniform steps per direction. Increasing the number of steps increases the footprint of the HBAO kernel but may introduce under-sampling artifacts.
- The "Angle bias" slider can be used to remove undesired AO at concave edges of low-tessellated meshes, as well as for hiding precision artifacts.
- The "Power exponent" slider controls the exponent of the power function that is applied right before the HBAO is blended with the destination colors.
- The "Blur width" slider controls the size of the depth-aware blur kernel that is applied to the HBAO, in number of pixels. Increasing the blur width filters out banding artifacts and reduces temporal aliasing artifacts.
- The "Blur sharpness" slider scales the maximum depth delta between the current pixel and a given sample. This parameter should be increased if the blur filter is making the AO bleed across object silhouettes.
- The "Compute-shader HBAO" and "Pixel-shader HBAO" radio buttons are for comparing the compute-shader jitter-free HBAO with the original HBAO algorithm (ground truth).
- The "Half-resolution" and "Full-resolution" buttons control the resolution at which the unfiltered AO is computed. The blur passes are always performed in full resolution, regardless of this parameter.

In the top-left part of the screen are displayed the average frame rate, the current resolution and GPU name, as well as the following GPU times:

- "Total" is the GPU time spent by `NVSDK_D3D11_RenderAO`, including:
- "Z" is for linearization the input depths (when using hardware depths as input) ,
- "AO" is for rendering the unfiltered HBAO,
- "BlurX" and "BlurY" are for the horizontal and vertical filter respectively,
- "Comp" is for blending the final AO to the output render target view.

Parameter Tuning

AO Radius

As shown on Figure 2, using a small AO radius multiplier, the output AO essentially darkens the concave areas of the surfaces. Using a larger radius multiplier makes the AO look more like a global illumination approximation.

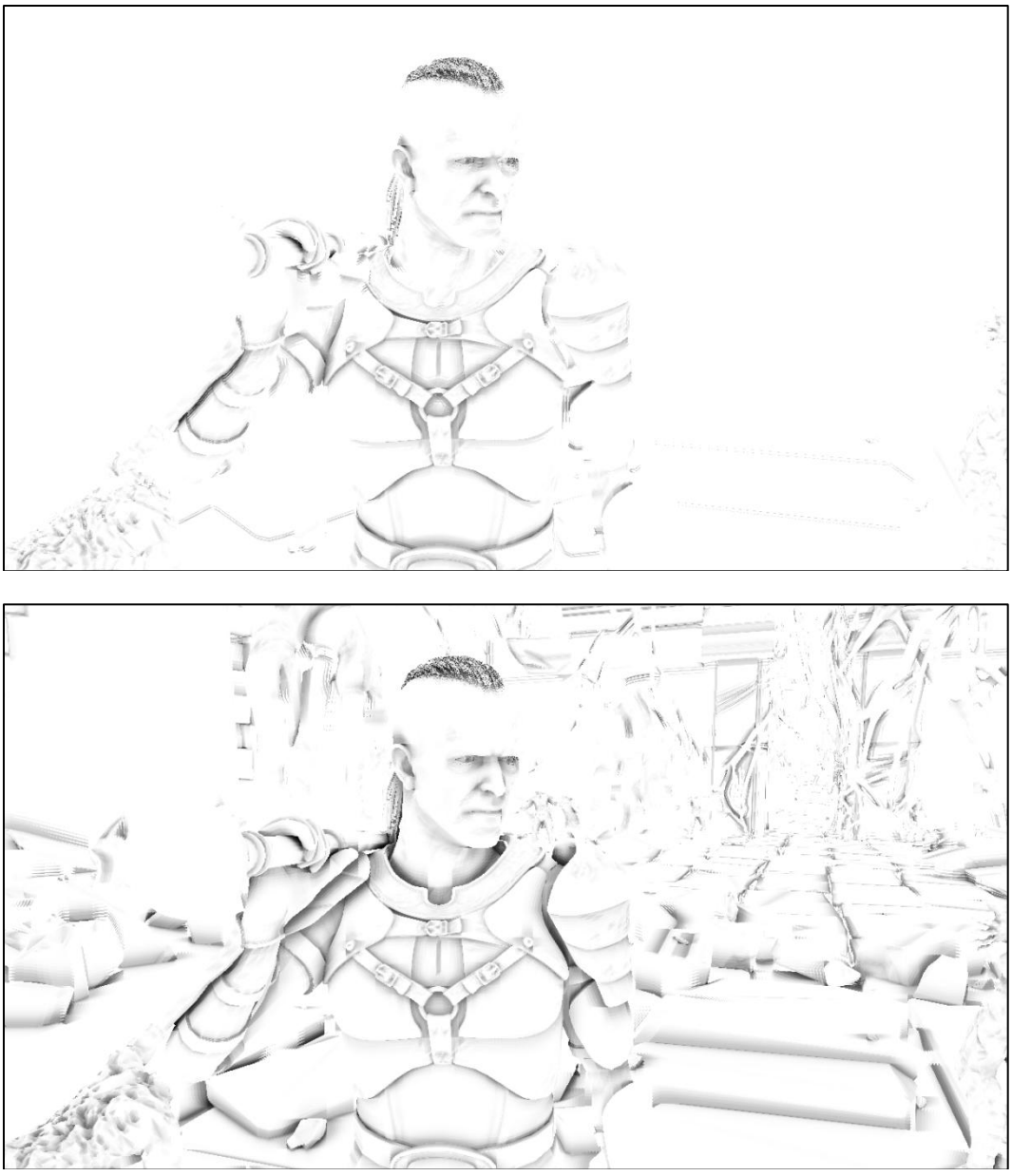

Figure 2. Unfiltered HBAO with R=0.2 (top) and R=4.0 (bottom).

Blur Size

Figure 3 shows that the banding artifacts from using the 4 axis-aligned directions can be hidden by a 33x33 blur kernel.

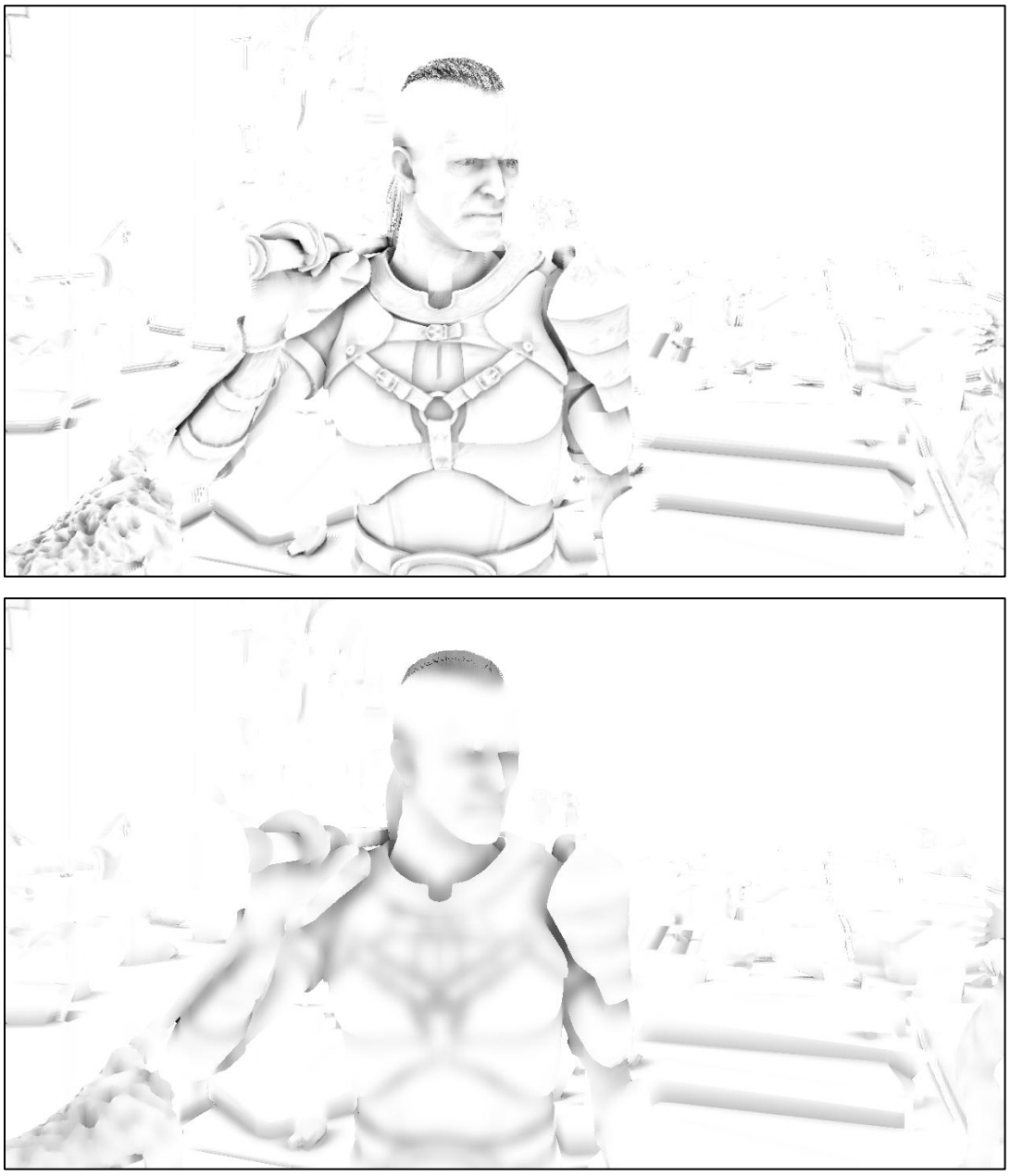

Figure 3. With no blur (top) and with a 33x33 blur kernel (bottom).

Code Organization

The C++ code of the library is located in the NVSDK_D3D11_SSAO directory. The API is in NVSSAO\NVSDK_D3D11_SSAO.h and the implementation is in HBAO_DX11_LIB.h and HBAO_DX11_LIB.cpp.

The HLSL source code of the shaders is located in the Shaders\Source\ directory. The associated HLSL byte-code files are in Shaders\Bin\ and are included in HBAO_DX11_LIB.cpp. When changing the HLSL source code, the byte code can be recompiled by launching Shaders\compile_all.bat.

Performance

Table 1 below presents an analysis of the NVSSAO GPU cost when running the SDK sample in 1900x1200 1xMSAA with the default settings (compute-shader HBAO). The HBAO is computed in full resolution with 4 directions and 8 steps per direction. The resulting AO is then filtered with depth-aware horizontal and vertical blur passes.

For the Medusa scene, the input depths are set with NVSDK_D3D11_SetViewSpaceDepthsForAO, whereas for the other scenes the depths are passed with NVSDK_D3D11_SetHardwareDepthsForAO. Therefore, there is no GPU time spent on linearizing depths for the Medusa scene.

Table 1. GPU times on GeForce GTX 580 in 1920x1200 1xMSAA.

GPU Times (ms)	*Medusa*	*AT-AT*	*Leaves*
Linearization	0.0	0.1	0.1
AO	2.1	2.1	2.1
Horizontal Blur	0.6	0.6	0.6
Vertical Blur	0.6	0.6	0.6
Compositing	0.2	0.1	0.1
Total	3.5	3.5	3.5

Acknowledgments

The AT-AT model used in this sample was created by Brad Blackburn and can be downloaded from http://www.scifi3d.com. The Medusa scene was captured from the NVIDIA Medusa demo [NVIDIA 2008].

References

[Akenine-Moller et al. 08] Tomas Akenine-Möller, Eric Haines, Naty Hoffman. "Real-Time Rendering - Third Edition". 2008.

[Bavoil and Sainz 08] Louis Bavoil, Miguel Sainz. "Image-Space Horizon-Based Ambient Occlusion". ShaderX7 - Advanced Rendering Techniques. 2008.

[NVIDIA 2008] Eugene d'Eon. "Life on the Bleeding Edge: More Secrets of the NVIDIA Demo Team". NVISION08. 2008.

Notice

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks

NVIDIA, the NVIDIA logo, GeForce, NVIDIA Quadro, and NVIDIA CUDA are trademarks or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2011 NVIDIA Corporation. All rights reserved.