GrateShader Demo — User Guide

Frank Jargstorff
fjargstorff@nvidia.com

April 21, 2004

1 Introduction

This user guide explains the six different shaders integrated in the Grate-Shader demo:

1. Transparency using alpha test.
2. Discarding transparent fragments.
3. Compensate lighting for self shadowing.
4. Removing the overshooting artifacts by clipping ends.
5. Anti aliasing
6. Applying textures to the "virtual geometry"

2 Controlls and Features

There are two basic kinds of controlls for the application:

- Those that can be activated through a menu or keyboard shortcuts. Right-clicking on the application window pops-up a menu with all the options. The keyboard shortcuts are given with each menu option.
- Interacting with the application by moving the mouse with the left or middle mouse-button down.

  Left Mouse Button rotates around the geometry in the center of the scene.

  Middle Mouse Button zoomes in and out on the scene.

2.1 General Navigation and Controll

Common to all experiments is the

- navigation capability,
- the **w**-key for wire-frame mode,
- the **h**-key to bring up the parameter-control sliders, and
- the **q**-key to terminate the application.

2.2 Controlls for specific experiments

Controlls that are for specific experiments are:

- The number keys **1**–**6** for changing to different shaders.
- The **a**-key to toggle the rendering order (bars last vs. bars first).
- The **f**-key to toggle rendering the "window frame".
- The **s**-key to toggle rendering a sphere intersecting the "virtual geometry".

3 Experiments

The first experiment should be to fire up that app and switch to wire frame mode. All the slats are "virtual geometry": They are rendered using a single quad.

3.1 Alpha Test Shader

The app should be in Alpha Shader [1] mode.

Using alpha-test for transparancy has the disadvantage that the resulting image is order dependent, i.e. surfaces with alpha need to be rendered after all other geometry in back-to-front order. Enable the sphere (**s**-key) and toggle the rendering order usign the **a**-key. In the initial mode parts of the sphere are visible through the gaps in the slats. When rendering the slats first the sphere behind the quad becomes invisible.

A possible solution to this problem is to use the hardware to discard fragments that are known to be transparent so that they do not modify the z-buffer value. This is shown in the next subsection.

3.2 Discard Transparent Fragment Shader

Using the same setup as in the previous subsection on the "Alpha Test Shader" you will find that the sphere will always be nicely visible independently from the render order.

3.3 Self Shadowing Shader

Apart from intersecting the slats/bars the red sphere also helps to determine the light direction. Orient the frame to be parallel with the light direction such that the slats are perpendicular to the light. What you see is that the slats are brightly lit despite the fact that first the frame and then each slat should cast a shadow on all the other slats.

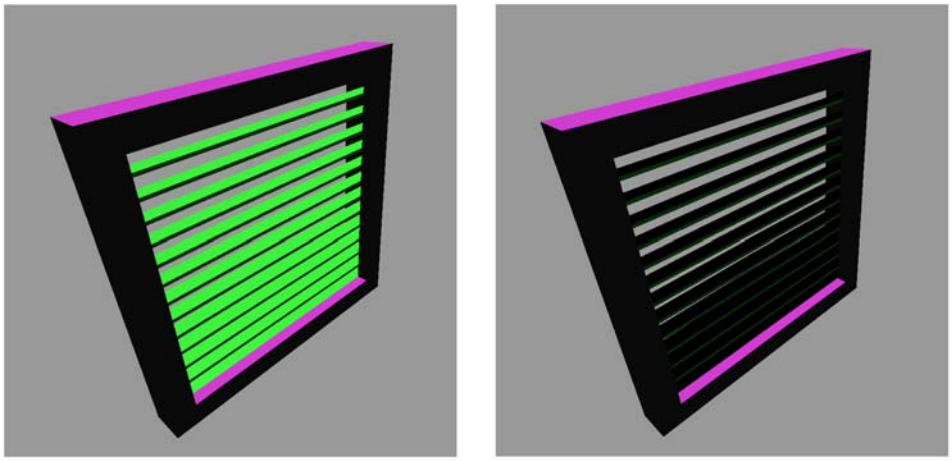

Figure 1: In both pictures the light source is straight above the frame. As one can see the top of the frame is brightly lit. In the first image all the slats below are also brightly lit since the shader does not take self shadowing into account. The second image uses the statistical self shadowing formula and the slats appear much darker.

The self-shadowing shader (Self Shadowing Shades [3]) takes care of this problem by compensating for the self shadowing.

3.4 Clipped Ends Shader

Use any one of the previously discussed shaders. Bring up the parameter sliders (**h**-key) and crank up the slat height and decrease the number of slats. Looking along the slats at a low angle you'll find that it seems as if the slats extend beyond the frame. This is because the math is set up to treat

the slats as infinitely long and give us a view onto these slats through the quad that's being shaded.

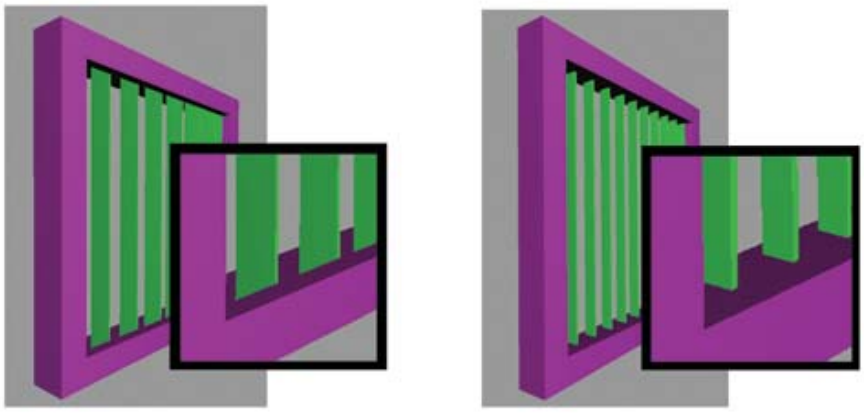

Figure 2: Unclipped ends of the bars cause visual artifacts.

By adding an extra test the ends of the slats can be clipped. Toggle between the alpha test shades and the Clipped ends shades [6] to see the difference.

3.5 Anti-Aliased Shader

When in any of the shaders discussed so far you crank up the number of slats and move the quad away from the view point you'll see massive Moiré patterns. By anti-aliasing the shader results these artifacts almost completely go away. Play with the Anti aliased shades [5] to see the difference.

3.6 Shader with Texture

The last of the shaders Textured shades [5] texture maps the NVIDIA logo onto the slats.

Figure 3: Screen shot of texture applied to a bunch of rectangular bars.