User Guide

Advanced Shadow Mapping

This code sample is an implementation of several techniques for handling wide-area fullscene directional light shadow mapping, including: simple orthographic projection, Perspective Shadow Maps (using the improvements described in Simon Kozlov's article in NVIDIA's *GPU Gems*, "Perspective Shadow Maps: Care and Feeding"), Light-Space Perspective Shadow Maps, and a variant of Trapezoidal Shadow Maps.

Perspective shadow mapping (Stamminger & Drettakis, SIGGRAPH 2002) is similar to Lance Williams' original shadow mapping; however, all objects rendered into the shadow map are first transformed through the eye's view and projection matrices, bringing them into post-projective space. This way, objects closer to the viewer are larger than objects far away, so more texel resolution is given to them, reducing perspective aliasing. Also, by operating in post-projective space, directional lights can be handled in the same manner as spot lights (as directional lights become point lights on the infinity plane in post-projective space).

Unfortunately, perspective shadow maps as originally presented had a number of shortcomings, severely limiting their utility for interactive scenes:

- Lights from behind the viewer were handled by "virtually" shifting the shadow camera (causing severe, instantaneous changes in effective shadow map resolution)
- Shadow quality was dependent on the relative positions and directions of the viewer and the light sources, creating temporal artifacts when anything in the scene moves.
- Shadow quality was fantastic near the viewer (when a "virtual" shift wasn't required), but severely degrades further away
- Obtaining optimal results for any light/viewer combination required substantial CPU interaction.

Simon Kozlov's article introduces many improvements to the original perspective shadow mapping algorithm (including an *inverted* projection matrix), that makes perspective shadow maps quite a bit more practical for use in interactive situations.

Light-Space Perspective Shadow Maps (Wimmer, Eurographics 2004) are another way to reformulate the original PSM projection, to avoid many of the singularity artifacts that afflict post-projective space, as well as providing a better balance between near and far shadow quality.

Trapezoidal Shadow Maps (Martin, Eurographics 2004) are a third alternative, that rely on 2D homogeneous operations that optimize the shadow map resolution based on the shadow of the view frustum. Note that this code sample does not implement TSMs exactly as described in the original paper; instead of finding the actual trapezoid vertices, the algorithm in the code sample operates directly on the edges. The resulting quality should be similar (if not identical).

This code sample was written using Microsoft DirectX™ 9 API, and requires a GPU with hardware shadow map support (a.k.a., depth/stencil textures), and at least pixel shader 1.1 support, or a GPU with R32F texture and pixel shader 2.0 support (GeForce 3+, Radeon 9500+).

Code Sample Features

- Low-angle viewer, with Znear=1m, Zfar=800m, approximating a generic outdoor FPS-style scenario.
- Single directional lightsource, traveling in a line across the upper hemisphere
- 40 shadow-casting objects located over a 2.56km^2 terrain, consisting of >2m triangles
- All shadowing handled with a single 1536x1536 (code-changeable) shadow map (note: on ps 1.x hardware, the shadowmap resolution is 1024x1024)
- View-box clipping and virtual-slideback implemented, to improve effective shadowmap resolution
- User controls over a variety of PSM, LSPSM, and TSM-quality parameters.
- Ability to toggle between PSMs, LSPSMs, TSMs, and orthographic shadow maps.

Shortcomings:

- Some fudge factors are used for the shadow camera's near and far clip planes in the PSM code.
- Shadow acne problems are not handled when run on hardware that does not support depth/stencil textures.
- Trapezoid space causes some extreme skewing of the Z axis, such that the original authors suggest using pixel shader 2.0 depth replace to avoid shadow acne artifacts. This code sample does not implement this.

Controls

Movement is performed using the standard right-handed FPS key bindings (WASD + mouse). To change the view direction, hold the left mouse button while moving the mouse. Various rendering options are available via buttons and sliders in the GUI.

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