Overview

Screen space ambient occlusion (SSAO) is a simple and predictable approximation to global illumination. We have extended it by replacing its depth buffer comparisons with a sampling of a volumetric discretization of the scene, while still evaluating it in image-space and thus retaining its predictable performance.

- Szirmay-Kalos et al. [2] substitute AO’s integration over the hemisphere with integration of points within an inscribed tangent sphere. We apply their theoretical insight to a practical volumetric scene proxy.
- The ratio of volumes $A$ and $A+B$ defined by volumetric AO is approximated by counting bits in a binary solid voxelization.

Visibility False Positives

- Naive SSAO suffers from a “black halo” artifact. The nearer column (labeled A) seems fringed with darkness. It is when evaluating shade points on the further object such as the one indicated that it is incorrectly taken to be occluding.
- Our algorithm avoids this artifact naturally without any specific fix.

Visibility False Negatives

- Conditionalizing use of SSAO depth samples based on relative distance can solve false positives.
- E.g., bottom of well is highly occluded but camera can’t see sides (dashed). False positive fix results in over-bright well bottom.
- Our approach avoids these issues by capturing a solid volumetric representation of the scene.

Example Image

- Zoomed-in views show contact shadows at multiple ranges and complex geometry visible in shadow forms.

Results on GTX 280

Summary of performance in fps. The algorithm variants are: 2d: a simple count of depth tests using a 2d kernel, sc: follows Starcraft 2 without hemisphere flip, get_voxel implements the tangent sphere sampling by looping around a get_voxel() function, dense128: examines all 280 tan-sphere voxels using shifting and masking to access voxels in column batches, stipple128: refines this by taking only half the samples, dense8: reorders the slicemap into a 8-bit per texel format before doing AO from that, stipple8: enhances this in the same way as stipple128, static128: omits the generation pass, static8: omits this too and also the reordering one, generate: just the generation pass, reorder: this single pass. Each model uses a slicemap resolution that is tuned to it. The dimensions are, ballista: 512, car: 632, dragon: 192, rhino: 512, spheres: 192. The “half” and “quarter” columns refer to these numbers.

Slicemap Generation

- Costs of generation are flat for small numbers of render target, then rise super-linearly with increased slicemap resolution.
- Generation using LUT in texture much closer to ideal.
- Upper bound on fragment shader perf. (writes zeros to all targets):
- Additional cost of reordering is model-independent (diff. vert. scale):
- This steep increase limits our technique, but many models work well with low resolutions.

AO on a Reordered Slicemap

- Total frame time is dominated by the costs of generation and reordering for wide range of screen resolutions. The reordering pass (solid lines) is always superior to performing AO directly on 128 bit texel slicemap (dashed lines of corresponding color).
- When generation and reordering are omitted, frame-time is largely insensitive to slicemap resolution.

References

- Szirmay-Kalos and Farkas (2007) \textit{Virtual Environments and Computer Graphics, University College London}
- Szirmay-Kalos, Sziróczki, and Lázár (2010) \textit{Virtual Environments and Computer Graphics, University College London}
- Andrew Cox and Jan Kautz (2010) \textit{Journal of WSCG}