Screen Space Cone Tracing for Glossy Reflections

Lukas Herrmanns Tobias Alexander Franke

Fraunhofer IGD & TU-Darmstadt

Figure 1: *Glossy reflections at varying surface roughness in the Crytek Sponza Atrium computed with SSCT.*

1 Introduction

A typical modern engine has a postprocessing pipeline which can be used to augment the final image from a previous render process with several effects. These usually include depth-of-field, crepuscular rays, tonemapping or morphological antialiasing. Such effects can be easily added to any existing renderer, since they usually rely only on information readily available in screen space. Recently, global illumination algorithms have been mapped to post-processing effects, such as the wide selection of Screen Space Ambient Occlusion methods. An insight in [Ritschel et al. 2009] is that screen space algorithms can sample more information than just occlusion: in addition to visibility Screen Space Direct Occlusion samples neighboring pixels to gather indirect bounces. Soler et al. [Soler et al. 2010] use mipmapped buffers to sample diffuse far-field indirect illumination and importance sample specular cones for glossy reflections, but do not consider to use mipmaps to access prefiltered bounces for specular cones. We present Screen Space Cone Tracing (SSCT), a method to simulate glossy and specular reflections. Instead of regular screen-space ray tracing, we adopt the concept of cone-tracing on hierarchical, prefiltered buffers to reduce integration costs.

2 Technical Approach

The idea of cone-tracing is to replace rays with cones in a ray-tracer. By doing so, the solid angle over which incident radiance is gathered usually by sampling many rays can now be defined through one cone which has an equivalent opening angle. A key idea in Crassin et al. [Crassin et al. 2011] is to exploit hardware filtering on 3D volumetric textures which save the first bounce of direct illumination injected from a Reflective Shadow Map into the voxels of such a texture. Integrating over a cone's opening angle in a volumetric texture is now equivalent to a lookup on a higher level mipmap of the filtered volumetric texture. We apply this idea to the view-space representation of a rendered scene: For each pixel, we reflect the view direction along its normal and cone-trace along the reflected direction. We assume that after the initial direct hit each pixel in the rendered camera view represents a diffuse reflector. As long as a reflected ray will hit another part of the rendered image, we can gather this first bounce and add it as indirect contribution onto the pixel currently processed.

We start out with a regular render pass from the camera view into a GBuffer which includes linear depth, normals and colors. In a deferred pass we compute the shaded image, after which we generate mipmaps for the linear depth, the normal and the shaded image and use them as input for SSCT. For each pixel, we reconstruct its position P and its normal N. We step through the image from P in the reflected viewing direction V_r for a maximum of 200 steps. To converge onto the hitpoint faster, we increase the step-size in image-space after each step by a factor of 2. If we overstep the hitpoint, we go back in the opposite direction of the ray at half the step-size. At each step, we compute the subtended angle of the cone and a mipmap level from this area which we use to access depth, normals and the shaded buffer. If we find a hit before leaving the visible space of the camera perspective, the buffer with the shaded image is queried for the indirect contribution. A larger cone opening will yield a higher mipmap level, thus returning the pre-integrated incident radiance over its solid angle.

By relying on mipmaps of the camera view space certain integration errors are unavoidable. The most apparent is that a solid angle can subtend either flat spaces or multiple pieces of geometry visible only at grazing angles, which can manifest as alias or temporal inconsistency when moving the camera view. As the cone angle size increases to simulate more glossy appearances, this error will increase notably. Another issue is that mipmapping itself is only a coarse approximation of the actual filtering process. We can achieve better results with manual filtering, albeit at much higher costs.

References

CRASSIN, C., NEYRET, F., SAINZ, M., GREEN, S., AND EISEMANN, E. 2011. Interactive indirect illumination using voxel cone tracing. *Computer Graphics Forum*.

RITSCHEL, T., GROSCH, T., AND SEIDEL, H.-P. 2009. Approximating dynamic global illumination in image space. In *Symposium on Interactive 3D Graphics and Games*, I3D '09, 75–82.

SOLER, C., HOEL, O., AND ROCHET, F. 2010. A deferred shading pipeline for real-time indirect illumination. In *ACM SIGGRAPH 2010 Talks*, SIGGRAPH '10, 18.

SIGGRAPH 2014, August 10 – 14, 2014, Vancouver, British Columbia, Canada.

ACM 978-1-4503-2958-3/14/08