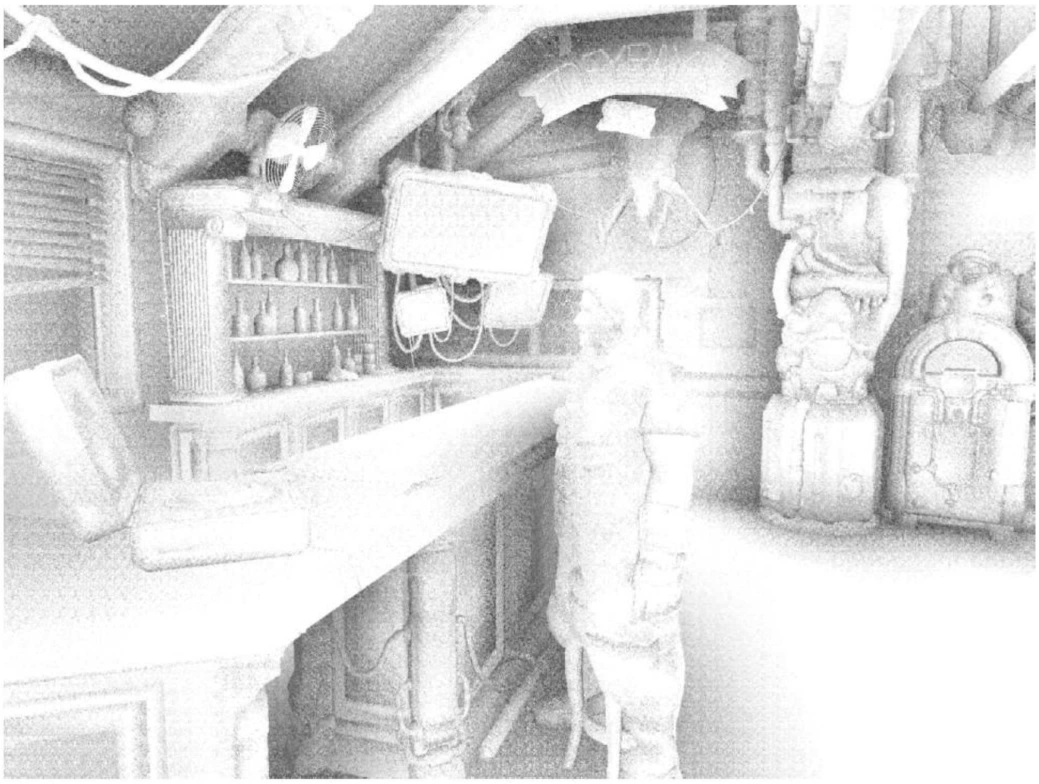
Screen Space Ambient Occlusion in StarCraft II



Introduction

This demo is a implementation of the Screen Space Ambient Occlusion (SSAO) technique used in the game StarCraft II according to the explanation in a SIGGRAPH 2008 game course “StarCraft II Effects & Techniques”.

Screen Space Ambient Occlusion

Screen Space Ambient Occlusion (SSAO) is a popular technique used in recent 3D games to simulate global illumination and it was first introduced in the game Crysis (2007). There are many SSAO implementations but all of them are based on a common main idea, which is approximating the occlusion function at points on visible surfaces by sampling the depth of neighboring pixels in screen space.

SSAO in StarCraft II

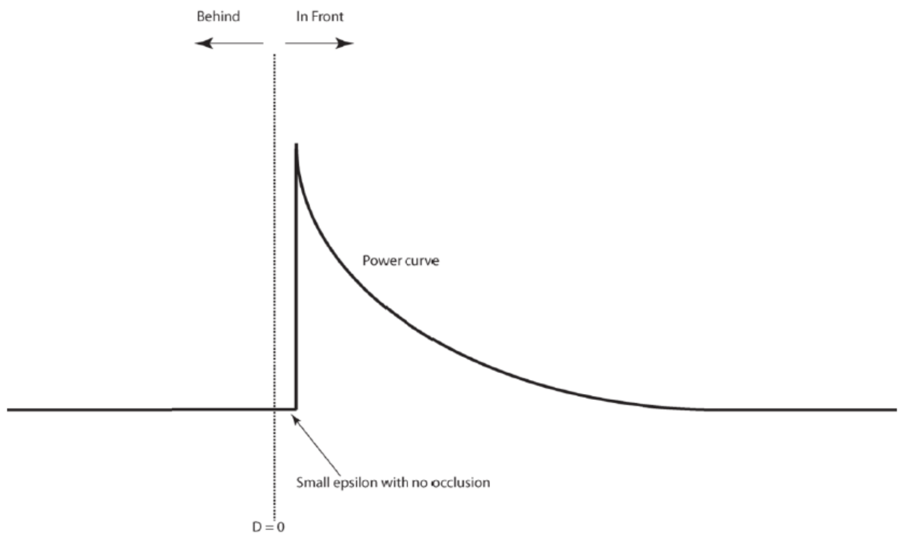
The linearized depth (not the depth stored in Z-Buffer) and surface normal are rendered to the texture in the first pass. In the second pass, a screen-size quad is rendered. In pixel shader of this pass, multiple (in my implementation, 16) samples are taken from neighboring points in the scene. These samples are offset in 3D space from current point being computed. They are then projected back to screen space to sample the depth by accessing to the texture created in the first pass.

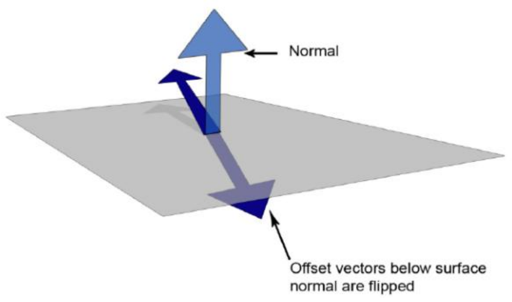
The objective is to check if the depth sampled at the point is closer or further away than the depth of the sample point itself. If the depth sampled is closer, then there is a surface covering the sample point.

The occlusion function is defined based on the following criteria, in which the "depth delta" is the difference between the depth of sample point and depth sampled at the point.

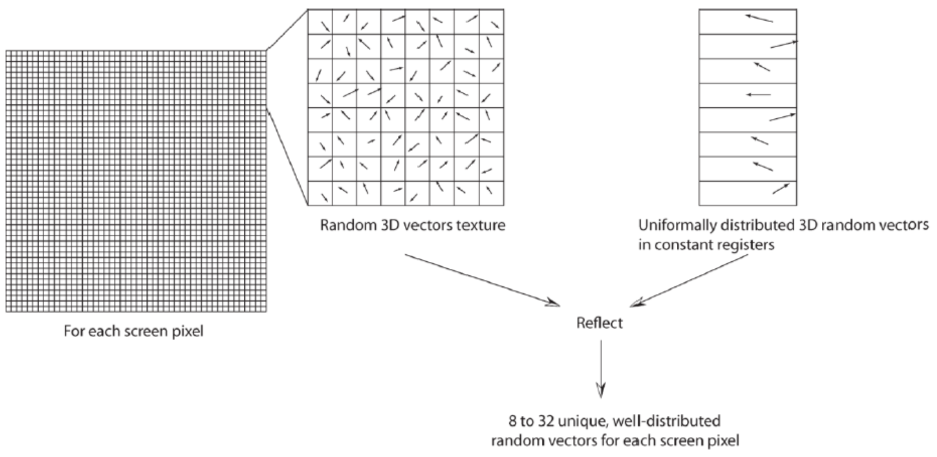
- Negative depth deltas give zero occlusion.
- Smaller depth deltas give higher occlusion.
- The occlusion value falls to zero again beyond a certain depth delta value.

In my implementation, the occlusion value drops along a quadratic function of depth delta and it is also associated with the dot product of the normal of the current pixel and the normal o f the sample pixel. Smaller dot product gives bigger occlusion value.





The sample points must be randomized thoroughly. Initially, we have several uniformly distributed 3D vectors and a 2D random normal texture. In pixel shader we lookup this texture in screen space and retrieve a unique random normal texture per pixel. All the 3D vectors are reflected by the random normal to get the final offset vectors. The sample points are obtained by adding these offset vectors to the position of the current pixel. To solve the self-occlusion problem, any offset vector below surface normal will be fillipped.

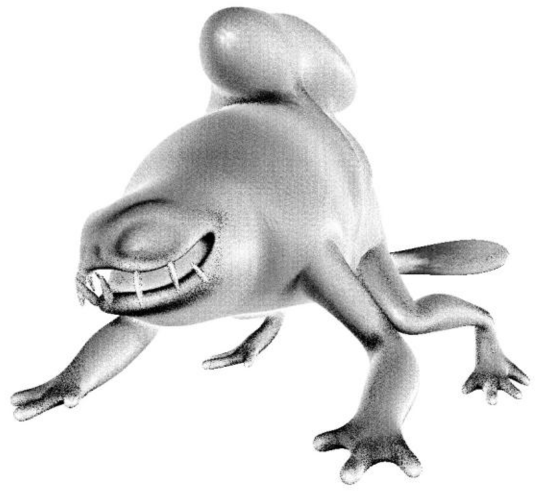
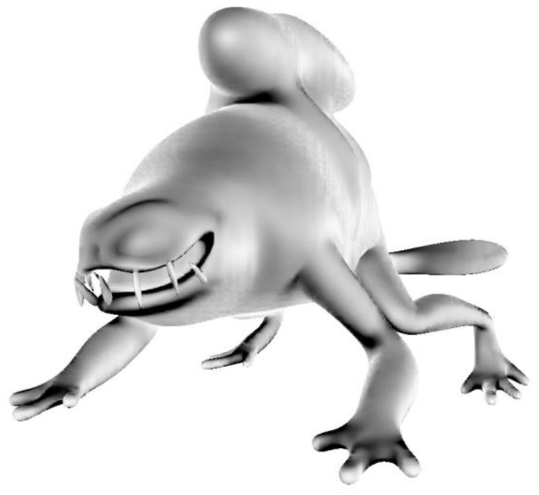

Random 3D vectors texture
Uniformally distributed 3D random vectors in constant registers
For each screen pixel
Reflect
8 to 32 unique, well-distributed random vectors for each screen pixel

Blur

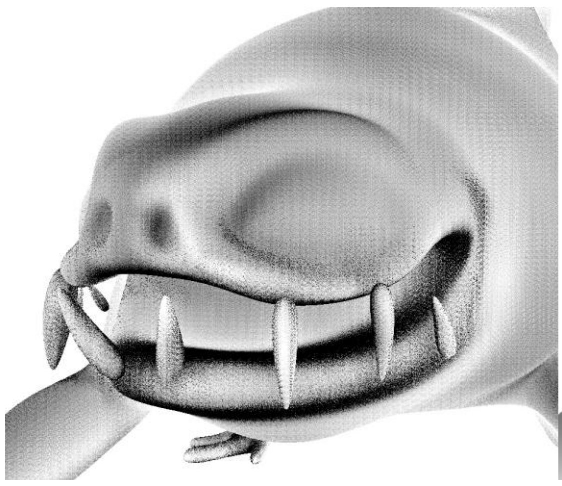
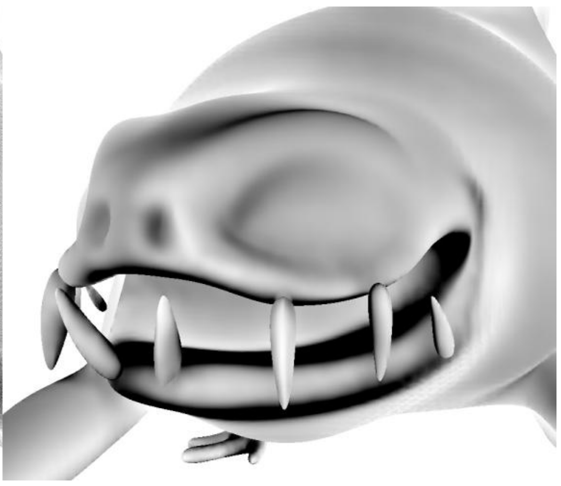
After the SSAO calculation pass, blurring is needed to reduce noises. It is not a simple blur algorithm but a smart blur algorithm considering the depth and normal variance.

Either a 2D Gaussian kernel or a 1D Gaussian kernel (for doing bilateral blurring in X and Y direction respectively, 2 passes needed) can be used. In order to preserve the sharp edges after blurring, for any sample pixel in the kernel, if its depth differs from the current (center) pixel by a certain threshold, or the dot product of normal of the sample pixel and the current (center) pixel is less than a certain threshold, then the sample pixel has zero weight in this kernel.

In my implementation, a Gaussian kernel with size 31 is used for bilateral blurring. The following two figures demonstrate the occlusion values before blurring and after blurring.

From the following two figures you can see that the edge information at the eye brows and maxima is preserved after blurring.

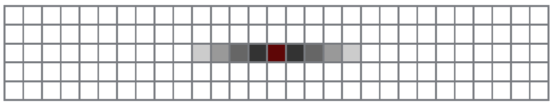
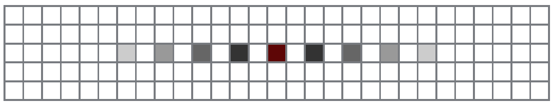



Performance Improvement

The occlusion values before blurring can be rendered to a smaller render target (or framebuffer) but render blurring result to an original-size render target (or framebuffer).

Interleaved sampling can be used to reduce the number of samples in the kernel without obvious loss of quality.

The left figure below is a normal 1D kernel and the right one is an interleaved sampling kernel. If we have same number of samples, in most of cases interleaved sampling gives a better blurring result.

Rendering with SSAO

Ambient occlusion value is usually used to occlude ambient light but you can also use it to occlude diffuse and/or specular light if it gives a special effect you want.

The following figures demonstrate a comparison of a simple Phong Shading effect without SSAO and that with SSAO. Only ambient light is occluded.

Diffuse light contribution: 0.6
Specular light contribution: 0.9
Ambient light contribution: 0.3
Shininess factor: 64