Guidelines for Project in EDAN35 High Performance Computer Graphics

Adam Backstrom och Ragnar Wernersson*

Lund University
Sweden

Abstract

In this project we have been looking in to two different effects, ambient occlusion and volumetric lighting. The ambient occlusion is implemented through the Alchemy AO algorithm with the use of pseudo random sampling and a Gaussian filter. The volumetric lightning is implemented through our own algorithm also using pseudo random volume sampling.

1 Introduction

Our first goal was to create a game. The concept of the game was to hide in the shadows, to have patroling guards with spotlights that you would have to avoid. To do this we needed ambient occlusion(AO) to get a realistic feel without light sources and volumetric light(VL) sources so that the player could see the light sources properly.

Unfortunately we were not able to finish the game dynamics but we were able to implement both AO and VL. So instead we ended up creating a demo to show these effects.

2 Algorithms

2.1 Ambient Occlusion

The algorithm we choose for AO was Alchemy AO [1]. It uses the geometry buffer to sample points in a circle with radius r around the pixel. The depth buffer is then used so that only samples that are inside the half-sphere with radius r outwards from the normal of the current pixel are used.

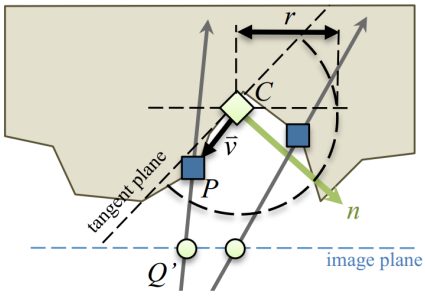

Figure 1: Description of AlchemyAO. Picture from http://graphics.cs.williams.edu/papers/AlchemyHPG11/VV11AlchemyAO.pdf

The following algorithm is used to implement the effect:

$$A = max\left(0, 1 - \frac{2\sigma}{s}\sum_{i=1}^{s}\frac{max(0, \bar{v_i}\cdot\bar{n} + z_C\beta)}{\bar{v_i}\cdot\bar{v_i} + \epsilon}\right)^k \quad (1)$$

The overall idea of Alchemy AO is that the amount of light given by a sample point is dependent on the angle between the normal and the vector to the sample point ($\bar{v}$). A high angle, 90 degrees and above, will not give any ambient shadow while small angle represent that the point is more covered by the sample and will get less light, hence more ambient shadow.

σ represents the overall strength, s the amount of samples and ϵ is to avoid division by zero. The $beta$ is to avoid self-occlusion which gives artefacts represented by lines orthogonal to the viewer's z-direction. z_c is the linear depth in view coordinates.

The random function used for sampling is a pseudor-andom function in the following format:

$$X = fract(sin(\bar{a}\cdot\bar{b}))*k) \quad (2)$$

Where $\bar{a}$ is sent in to the function and $\bar{b}$ is a constant 2-vetor and k is a scalar. For AO we start by using the pixels position as the $\bar{a}$ vector and two different values on k to get two different generated values for the same pixel. These values define the radius and angle of the first random sample point. For the next sample generation, the previous two values are sent in as the new $\bar{a}$. The resulting sequence will always be the same for each pixel. In this case this effect is beneficial since the noise will be stationary on the screen. The resulting buffer image can be seen in figure 4. To remove the high frequency noise from the buffer a Gaussian filter is applied at the resolve phase of the rendering. Final results can be seen in figures 2 and 3.

2.2 Volumetric Light

The overall idea of our volumetric lightning is to simulate particles in the air by sampling along lines, from the view point to the current pixel, into the scene. Since we already render the bounding volumes for each light with the DeferredSpotlight shader this effect can be added to this. The only thing that need be changed is to remove depth test for the shader. Instead of sampling the whole distance to the current depth value, one can use that the light has a limit volume by the shaped of a cone. First we transform the view position and the direction into coordinate system of the current spotlight. This gives the view position (x_0, y_0, z_0) and the line direction (a, b, c).

The cone's equation can now be expressed using the following form:

$$x^2 + y^2 = (\tan(\alpha)z)^2 \quad (3)$$

Where α is the angle of the cone.

If we combine this with the line equation of the view direction:

$$\begin{aligned} x &= a\cdot t + x_0 \\ y &= b\cdot t + y_0 \\ z &= c\cdot t + z_0 \end{aligned} \quad (4)$$

These equations can be combined into a quadratic equation of t which gives two solutions, if the line intersects the cone. These represent the two intersection points. These points are compared with the depth buffer to see if there are objects in front or in between the two points to remove any unnecessary sampling. There are a lot of different special cases depending on the view position and the direction that has to be taken in to account apart from this. None of which we will go deeper into.

*e-mail: ragnarwernersson@gmail.com, adamb3.14@gmail.com

[1] http://graphics.cs.williams.edu/papers/AlchemyHPG11/VV11AlchemyAO.pdf

The distance between the intersecting points are sampled. The light in each sample is determined by the angular and distance fall-off as well as weather the point is in shadow and added together. In order to get a realistic effect one need small step sizes and this take a lot of computing power.

2.2.1 Optimization

Firstly it is unnecessary to sample where the distance fall-off have removed most of the light. A max radius from the light position is used to remove this.

The second optimization is by using random sample distances to avoid repeating patterns from the volumetric shadows. Since it is the step size that is randomized the texture caching will still be working. The random function used is the same as for AO but now it is the scalar k that changes between samples for the same pixel. By adding a dependence on the distance to the light cone for k the previous effect where the same pixel always had the same pattern is removed. In this case it is better if the noise varies over time because it then gives the effect of small particles moving in the air, rather than something that is stuck on the screen. This noise effect is also reduced by applying a cross-filter on the volumetric light buffer. The cross-filer also adds a better lighting effect where strong lights bleed out into a cross, which means that the strength of the filter can be increased without destroying the effect itself.

3 Results

The values used in the AO algorithm were:

$$\begin{aligned} \sigma &= 1.0 \\ s &= 12 \\ \beta &= 0.005 \\ \epsilon &= 0.01 \end{aligned} \tag{5}$$

The result can be seen in the following images. The following pictures show the complete effects with AO and VL. With our current ditance cap and random step-size sampling we currently have an average of 10 samples and never more than 20.

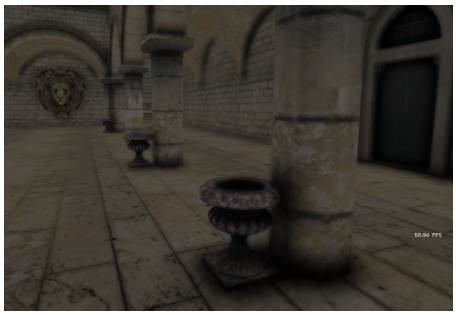

Figure 2: Ambient occlusion

Figure 3: Ambient occlusion

Figure 4: Ambient occlusion buffer

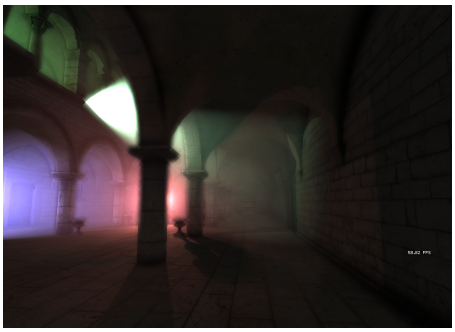

Figure 5: Volumetric Light

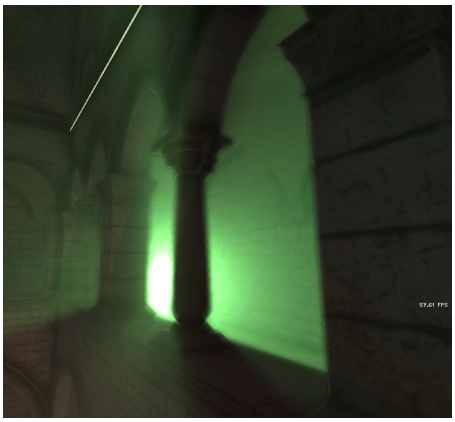

Figure 6: Volumetric Light

4 Discussion

Even though we didn't have time to implement the game mechanics we had planned we are pleased with the results, it should not be too hard to create the game out of what we have accomplished.

We still get some frame rate drop when the whole screen is blinded by light. This is probably due to the long distance that needs to be sampled. Also even though the effect is very pretty it

Figure 7: Volumetric Light

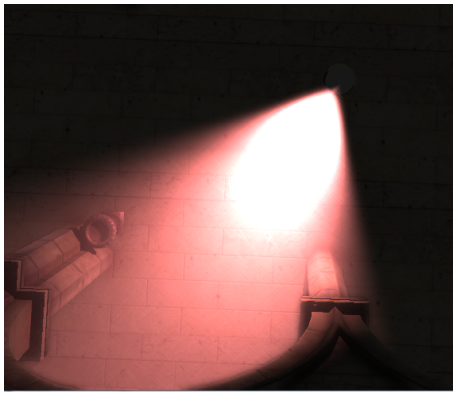

Figure 8: Volumetric Light

Figure 9: Volumetric Light

would be nice to reduce the noise even more, especially where the difference in light strength is high.

5 References

McGuire, M., Osman, B., Bukowski, M., Hennessy, P. 2011 The Alchemy Screen-Space Ambient Obscurance Algorithm, http://graphics.cs.williams.edu/papers/AlchemyHPG11/VV11AlchemyAO.pdf