FRR - Assignment 2 - SSAO

Diego Campos Milian

June 1, 2020

1 Screen Space Ambient Occlusion

The application allows to visualize three different screen space ambient occlusion implementations. First simple and straightforward SSAO, a 3D SSAO and a separable version of the first one.All the implementations use the same vectors to produce it's sample. This vectors are separated in two groups, first a set/kernel of 64 random vectors and a second set of noise vectors set as a texture of 4 by 4 repeating along the screen. Also, in all implementations, are passed position and normal information as textures.
In all of them is possible to change the following parameters:

- Maximum sample radius .
- Maximum depth difference between a fragment and a sample.
- Occlusion intensity.
- Number of samples.

1.1 Standard SSAO using normals

This implementation can be found in the ssao.frag fragment shader. This is a stantdard implementation of SSAO, so it takes the fragment texture coordinates as the initial point p and calculates it's occlusion from different samples reflected by a noise vector, which upon projected creates the sample point p'. Then, if the depth of p' is closer to the camera than p it will contribute to occlusion, not otherwise.
Usually the main problems of this kind of implementation are two. First, fragments that are far a part may be occluded incorrectly, this is simple solve by adding a range check that limits how far the depth of p and p' can be from one another. And second, can happen that surfaces that are not aligned with the image plane will self occlude, this is solve by comparing the normalized normals of p and p' with a dot product, this way if the normals are too similar we will consider that they pertain to the same plane and therefore that sample will not contribute to occlusion.

1.2 3D SSAO

The 3D SSAO implementation can be found in 3dssao.frag fragment shader. This imlpementation takes profits of the 3D geometry information of the fragment, position and normal, to properly reorient the kernel vector with the surface normal providing more meaningfull samples. However, the main different between this and the prior implementation is that to decide if a given sample will increase the occlusion or not, we compare the depth of the sample and the depth of it's projection point p'. If the sample is closer to the camera, smaller depth, than it's projection it will contribute to occlusion.

1.3 Separable SSAO

This cab be found at the sao.frag fragment shader. This implementation perform a separable version of the first algorithm, that is that instead of doing n^2 samples will only perform n samples in the x axis and another n in the y axis as it's perform in a multi-pass Gaussian filter. In this case though, due to the nature of the AO algorithm, this can be perform in a single pass, and the axis will differ from fragment to fragment. To obtain those axis we use the noise vector texture that we have to obtain what will be the x axis, so we have only to calculate the orthogonal y axis from that.

1.4 Blur

The blur fragment shader can be found in blur.frag. One of the downsides of the SSAO it's that the image may suffer from some pointillism/acne/noise effect, a simple blur pass helps avoid the issue. Then, what this implementation those is make each fragment be the average of it's surrounded ones. The amount of surrounding fragments taken into account is the same as the size of the noise texture, surprisingly using a different number, even greater, may increase the pointilism effect.