CS562 Project 4

Synopsis

Continue from the previous projects, now adding an Ambient Occlusion/Obscurance algorithm, with a bilateral filter as an edge-aware blur to smooth out the noise in the calculated ambient occlusion factor. Apply the resulting ambient occlusion factor to the ambient light (from the previous IBL project), but not to any direct light calculations.

References

The Alchemy paper (my preference):

https://faculty.digipen.edu/~gherron/references/References/AmbientOcclusion/VV11AlchemyAO.pdf

Bilateral filter:

http://en.wikipedia.org/wiki/Bilateral_filter

AO Instructions

For a given pixel with a world space position and normal P, N (from the gbuffer), carefully choose (as shown in the next section) a selection of n points P_i around P within a specified range of influence R. The ambient occlusion at P is approximated by summing up the occlusion of the n points P_i thusly:

$$S = \frac{2\pi\, c}{n} \sum_{i=1}^{n} \frac{max(0, N \cdot \omega_i - \delta\, d_i)\;\; H(R - ||\omega_i||)}{max(c^2, \omega_i \cdot \omega_i)}$$

where

- $\omega_i = P_i - P$
- R is the range of influence, past which points P_i will no longer occlude P, (about 1 meter for human scale scenes)
- $c = 0.1R$ (or so) is the point at which the fall-off function reaches its maximum of one,
- n (approx 10 to 20) is the number of points being used to approximate the integral,
- $\delta = 0.001$ (or so) and $d = world\ space\ depth\ of\ P_i$ provide a depth based threshold preventing neighboring polygons from occluding each other,
- $H(R - \|v(\omega_i)\|)$ is the Heaviside step function – a mathematical version of an "if" statement, returning 0 for negative arguments and 1 otherwise, used here to exclude points outside the range of influence R.

The final ambient factor is:

$$A = (1 - s\,S)_+^k$$

where s and k are adjustable scale and contrast factors, and the + subscript says to clamp away any negative values. This value scales down all ambient light – which for this sequence of projects is the IBL from project 3. It does not scale down any direct light where shadows play a similar role.

Selection of points P_i

See equations 6-8 in https://faculty.digipen.edu/~gherron/references/References/AmbientOcclusion/McGuire12AO.pdf

Given:

- A pixel with
  - **integer** coefficients (x', y') (from gl_FragCoord.xy)
  - and **floating** coordinates (x, y) (from $(x'/Width, y'/Height)$
  - P, N: world position and normal from the gbuffer at (x, y)
  - d: camera space depth, also from the gbuffer at (x, y)
- A world space range of influence R.
- A number of points to sample n.

we will compute n points in a spiral around the pixel, distributed widely enough to look random, and with a pseudo-random rotation between neighboring pixels.

The pseudo-random rotation is the following hash function on the **integer** pixel coefficients:

$\phi = (30 * x' \wedge y') + 10 x' y'$ where ^ means XOR

The n points:

for each i in 0 to n-1:

$\alpha = (i + 0.5)/n$ // In range 0 ... 1

$h = \alpha R/d$ // Spiral radius, in range 0 ... R/d (= R projected to screen)

$\theta = 2\pi\alpha(7n/9) + \phi$ // Spiral angle: 7 turns for every 9 points

Read P_i from gbuffer at $(x, y) + h * (\cos\theta, \sin\theta)$

Bilateral blur filter

Much like the blur filter done earlier in the semester, each output pixel will be a weighted average of a range of input pixels. In this case however, the weights will be *only partially* known ahead of time. The bilateral filter is not separable into horizontal and vertical passes, but we'll ignore this technicality and do so anyway.

Let $\Omega(x)$ represent a window of pixels around x. Then each filtered pixel is computed as

$$\hat{I}(x) = \frac{\sum_{x_i \in \Omega} W(x_i, x)\, I(x_i)}{\sum_{x_i \in \Omega} W(x_i, x)}$$

where the weights are based on both the distance between pixels x and x_i as well as the difference in values at those two pixels:

$$W(x_i,\, x) = R(x_i,\, x)\; S(x_i,\, x)$$

where

- $S(...)$ is the *spatial* kernel: a Gaussian based on the distance between the two pixels – exactly the same as in the blur from earlier in the semester. This can stored in a pre-calculated table of Gaussian weights.
- $R(...)$ Is the *range* kernel: This measures how similar two pixels are in geometry – i.e., whether they are on the same surface or across a boundary between two surfaces. It uses both the normals and the depths (from the eye) to make this judgment.

  $$R(x_i,\, x) = (N_i \cdot N)_+ \; \frac{1}{\sqrt{2\pi s}}\; e^{-\frac{(d_i - d)^2}{2s}}$$

  where

  N, d, N_i, d_i are the normals and depths (from the g-buffer) of x and x_i respectively,

  s (try 0.01) is the variance (width) of the Gaussian, and

  the notation $(...)_+$ means $max(0, ...)$

What to display

You should have a way to display your ambient occlusion factor, both before and after each blur pass, and of course, a display of the final image with the ambient occlusion applied to the ambient light (as calculated by the IBL project).

Demonstration

Submit a zip file containing the relevant code and a project report. Your report should contain sufficient screen captures (and accompanying text) to demonstrate the correctness of your project.