My progress this week. I finished implementing the method. The result had really a lot of artifacts, of which most of them were due to incorrect sampling of textures. Close to the borders, in fact, there are some error due to discretizing everything to a texture that caused the mentioned artifacts.

The three main artifacts were caused by:

- Incorrect sampling of the G-buffer
- Incorrect sampling of the cubemap
- Shadow bias (when sampling the depth of the texture).

I have applied the following fixes:

1 Incorrect sampling of the G-buffer

We modify the world coordinate of the point $\mathbf{x}_o$ with normal $\vec{n}_o$ in order to "shrink' a little bit towards the inside of the object according to the light direction $\vec{\omega}_l$:

$$\mathbf{x}'_o = \mathbf{x}_o - \epsilon_g(\vec{n}_o - \vec{\omega}_l(\vec{\omega}_l \cdot \vec{n}_o))$$

2 Incorrect sampling of the cubemap

Since we are sampling a cubemap, we do not need to account for the light direction in this case. In addition, the cubemap needs a 3D vector to be sampled with. So, instead of using $\mathbf{x}_o$, we use a point slightly intruded (i.e. displaced along the normal) in the calculations, according to:

$$\mathbf{x}'_o = \mathbf{x}_o - \epsilon_c \vec{n}_o$$

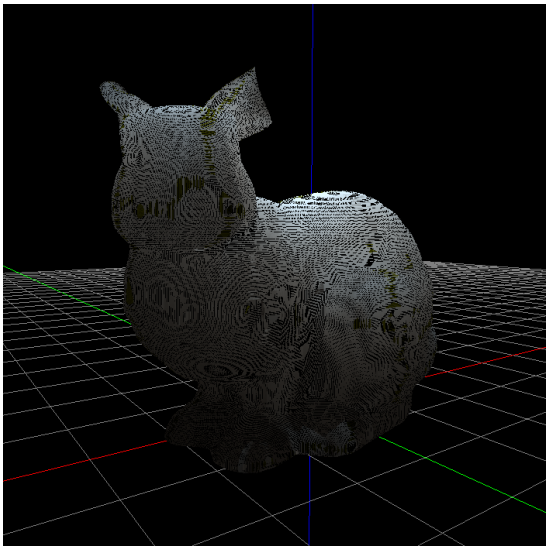
3 Shadow bias

In order to avoid artifacts such as shadows acne, we use a bias when comparing the z values of a point to determine if the point is in shadow or not. This implies that we need to convert the depth value from texture space ($\mathbf{z}_{tex}$) to world space again ($\mathbf{z}_{world}$). Since we are using an orthographics camera, the z value is the same in clip coordinates and in normalized device coordinates. Then, we simply use the camera projection properties ($\mathbf{z}_{far}$, $\mathbf{z}_{near}$) to convert the depth into the camera local space, in order to finally add the camera position transformed z value ($\mathbf{z}_{camera}$) and reconstruct the depth in world space:

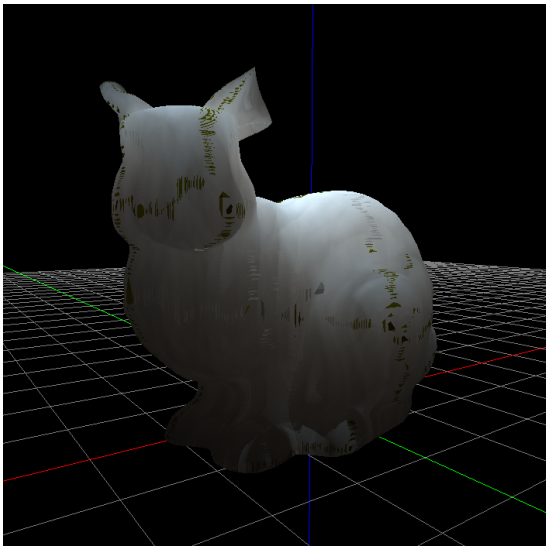
$$\mathbf{z}_{world} = \mathbf{z}_{camera} - \frac{\mathbf{z}_{far} - \mathbf{z}_{near}}{2}\left(2\mathbf{z}_{tex} - 1 + \frac{\mathbf{z}_{far} + \mathbf{z}_{near}}{\mathbf{z}_{far} - \mathbf{z}_{near}}\right)$$

And finally compare the obtained z with the z position in world space of the point ($\mathbf{z}$) using the bias ϵ_b, so a point is lit iff:

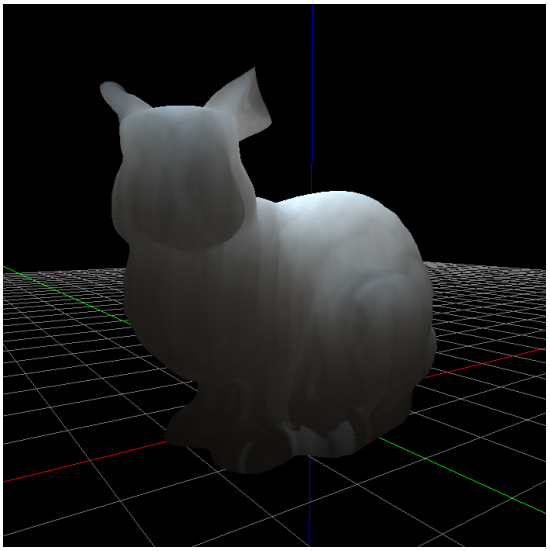
$$\mathbf{z}_{world} - \epsilon_b < \mathbf{z}$$



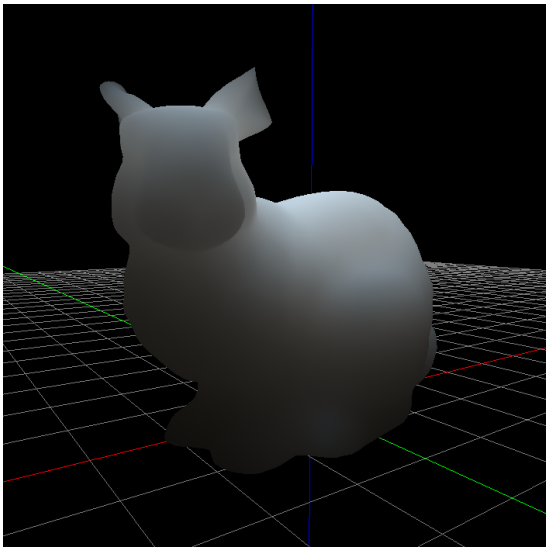
(a) Without shadow bias and sampling fixes



(b) With shadow bias and without sampling fixes

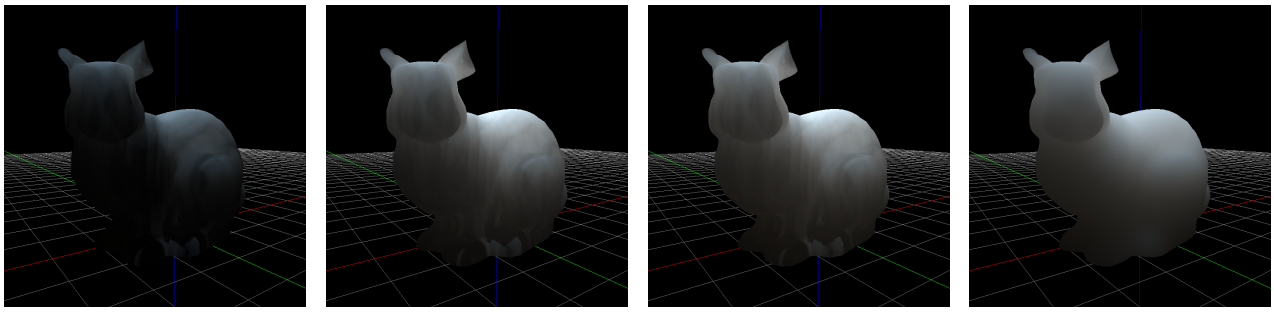


(c) With both shadow bias and sampling fixes



(d) Reference

Figure 1: Progressively removing artifacts to get to the final image.



(a) 10 samples **(b)** 150 samples **(c)** 300 samples **(d)** Reference

Figure 2: Different results according to the number of samples.

4 Samples number

The samples have been arranged to take first the closest to the center of the disc. We can see that with this little thing around 150 samples we stop improving the result, with a radius of 0.5 (in texture space).

5 Future work

My schedule for next week:

- Solve banding artifacts that are still present (seem to be caused by sampling too close to the point of interest)
- Try to add the directional dipole (at the moment I am testing Jensen)
- Find reasonable formulas for the epsilons (at the moment they must be set manually)
- Focus a little bit more on the report (deliver previous work section and reorganizing some notes on the implementation)