Antialiasing

Last time we wrote a very simple ray tracer to render a sphere. Looking at the output image, we can see that the result is less than ideal and contains sharp edges instead of curved boundaries for the sphere. Why is this happening?

To fully understand this, we must first discuss the fact that computers are *discrete* devices that can only display a *finite* number of pixels. Spheres (and objects in general) are continuous entities. Therefore, any representation of an object in the computer will be an *approximation* of the object (or surface, or texture) and as a result is subject to *aliasing*. In our case, the "jaggies" that we're seeing are the result of approximating the curvature of the sphere with a finite number of pixels. Other examples of aliasing include:

- Incorrect rendering of small details,
- moire patterns,
- colour banding.

We will discuss some simple techniques to reduce aliasing in our images. The process is therefore known as *antialiasing*. It is important to note that we cannot completely eliminate aliasing, we can just reduce it to "acceptable" levels (with the definition of acceptable varying depending on the application) or we can replace it with noise.

The techniques we will be discussing involve sampling pixels with multiple rays, which makes them very easy to implement in our current code. We will later discuss multisampling in greater detail.

Aliasing Effects

Many aliasing effects in ray tracing are a direct result of the fact that ray tracing is a *point-sampling* process. In other words, we sample a point with an infinitely small ray, and then apply that value to the entire surface of the pixel. To better understand this, consider a square image with a diagonal drawn in it. The lower half will be yellow, and the upper half black. Now lay a regular grid on top of it. You will notice that the diagonal does not always cover the *entire* pixel, it instead covers part of the pixel. Now take a sample only on the centre of each pixel. What you get is a jagged staircase pattern instead of a smooth line.

Another case where this occurs is when the intensity (think brightness) of the image varies and is rendered at resolutions small enough that the smallest details cannot be reproduced. See sinusoid demo.

Remedy 1: Increase the Image Resolution

The first option for reducing antialiasing is to simply increase the image resolution. This will reduce the *appearance* of jaggies, but it will not eliminate them. The fact will still remain that any curved edges will still have jagged edges, they will just be smaller. Now there is an argument to be made that having the jagged

edges be small enough that we can't see them anymore is sufficient, however this involves a number of other factors that will determine how small is enough. Furthermore, the increase in resolution will also affect performance as we will have even more pixels to sample.

Remedy 2: Regular Sampling

If the issue is that we're only taking a single sample per pixel, then a possible solution to the problem is to simply take *more* samples per pixel. A very straightforward way of accomplishing this is to divide each pixel into a regular grid, and sample at the centre of each cell. The final pixel colour is therefore the average of the colours in the grid. Fortunately, this is very easy to implement in our code, and so we can obtain a better result without any extra effort. The downside of regular sampling is that aliasing is still present and we can still see moire artifacts (see the sinusoid demo).

Remedy 2.1: Random Sampling

If regular sampling doesn't work, then what other option do we have? Well, what we can do is to do *irregular* sampling. Since we do not want to display any kind of bias towards a particular region of a pixel, we will use *random* sampling. In other words, we will fire several rays from random starting positions within the pixels and then average the results. While the final outcome is almost indistinguishable from the regular sampling in the case of the sphere, the results change wildly in the case of the sinusoid, as the moire patterns have been replaced with noise. While this may not seem to be a good alternative, it turns out that the way or eyes work makes it less susceptible to noise than patterns. In fact, humans have a very high tolerance for noise, but not so much for repeated patterns. This means that we can replace aliasing with noise and the image will look *better*, even though it does not represent the scene any more accurately. That being said, they can look significantly worse at lower resolutions.

Remedy 2.2: Jittered Sampling

Using random sampling generally results in clumps and areas left out of a particular pixel. A way to improve this is to force the rays to be evenly distributed over the surface of the pixel, while still maintaining the randomness. We can achieve this easily by subdividing each pixel into a regular grid (similar to regular sampling) but instead of firing a ray from the centre, we fire one from a random location in the cell. This ensures that while there may be clumps and gaps, it is not as severe as with normal random sampling.

More to come...

There are other ways of reducing aliasing. For example, none of the techniques we have discussed will solve the problem of a checkerboard plane stretching

infinitely into the horizon, as the area that the tiles cover tends to infinity as they reach the horizon. In this case we can apply a technique called *filtering* which involves sampling both the pixel we're interested in, as well as pixels around it. We will discuss this later on in more detail.