# ANUCG HW7 report

u6740827

October 2023

## 1 Acknowledgement

Heming Zhu's own work, no other help gained. Code built based upon given code skeleton. I am the smartest ass on the planet.

## 2 Jittered Sampling

Heming Zhu implemented jittered sampling to achieve anti-aliasing for this path-tracing program. The main idea of jittered sampling is that for one pixel, multiple rays are cast with random distributions in direction or location. The methodology used here is to get random jittered weight $w_x, w_y \in [-1, 1]$ via standard distribution for every sample light on each pixel. When calculating light direction from the pixel index, we apply the jittered weight to the pixel index to randomize the eye-ray direction.

Compare picture 1 and 2 to observe the result of super-sampling.

## 3 Complexity and Correctness Analysis

The procedure of calculating the color of one pixel is by looping over $[0, spp]$, for each loop, enter recursive function $\phi$. Within each level of $\phi$ recursion, it has a $RR$ possibility to invoke another $\phi$.

We conclude the complexity of calculating one pixel's color of $spp, RR$ as:

$$\mathcal{O}(spp \cdot complexity(\phi)) \tag{1}$$

We now derive the complexity of $\phi$. The event to stop recursion is to get a random number greater than $RR$, the probability of such event is $1 - RR$. The probability of keeping recursing is $RR$. We state here that the complexity of $\phi$ is linearly proportional to the depth of recursion, so we now derive the expectation value of depth of recursion in terms of $RR$.

We define the random variable $D$ to represent the depth of recursion. The probability of $D$ happening can be stated as below:

$$P(D) = (RR)^{D-1} \cdot (1 - RR) \tag{2}$$

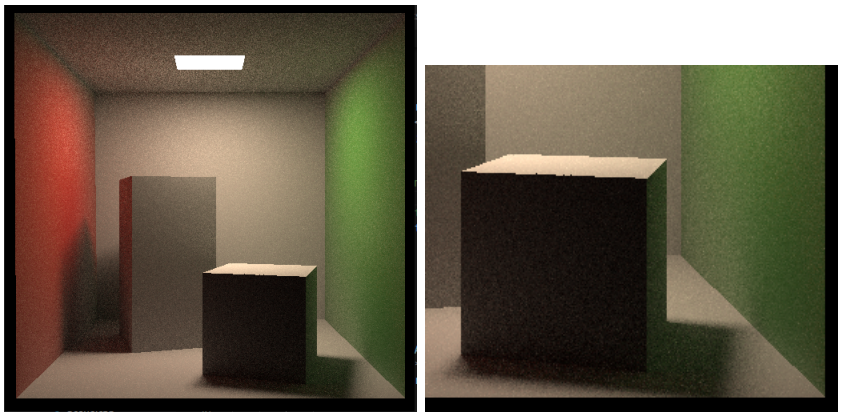

Figure 1: Rendered without jitter sampling, ssp = 64, RR = 0.8

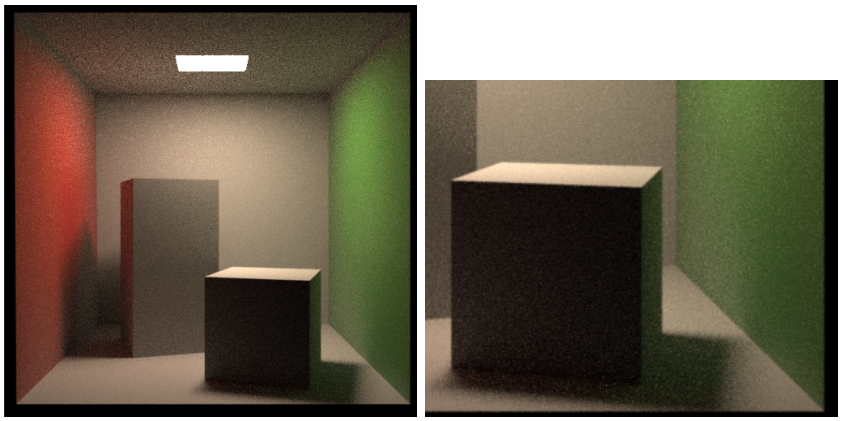

Figure 2: Rendered with jitter sampling, ssp = 64, RR = 0.8

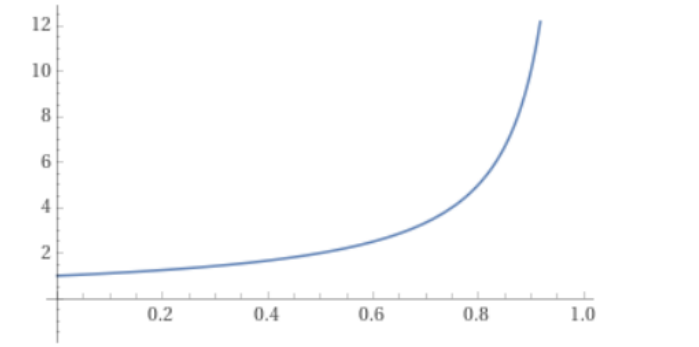


Figure 3: predicted recursion depth, i.e., complexity versus RR curve

Since that for $\phi$ to reach $D$th recursion, $D-1$ times of RR happened, and one time of (1-RR) happened.

Therefore, the expectation value of recursion depth is:

$$E(D) = \sum_{x=1}^{\infty} D \cdot (RR)^{D-1} \cdot (1 - RR) \tag{3}$$

We perform a transformation of variables to cope with Geometric (p) distribution, define $RRc = 1 - RR$, we get:

$$E(D) = \sum_{x=1}^{\infty} D \cdot (1 - RRc)^{D-1} \cdot (RRc) \tag{4}$$

Thus, the expected value of depth is $\frac{1}{1-RR}$.

We conclude the complexity of getting one pixel's color as:

$$\mathcal{O}(spp \cdot \frac{1}{1 - RR}) \tag{5}$$

We expect the spp to influence computational time positively linearly, while complexity has a reciprocal relationship with $RR$ shown in Fig (3).

We restrict $RR$ and vary *spp* to find the relationship between *spp* and runtime.

We perform experiments to observe the exact behavior we predicted above in fig 4.

In terms of graphical influences, we predict that *spp* is responsible for the correctness of both direct lighting and reflections. We calculate the color of one surface via Monte Carlo integration, while *spp* is the prime factor of the accuracy of the integration.

We predict that $RR$ is responsible for the correctness of reflections. Since it is the probability one ray shot again. The higher the RR, the reflection would be more close the correct result, and lower the noises on the screen.

We paint two pairs of graph fig 6 5 to illustrate the difference.

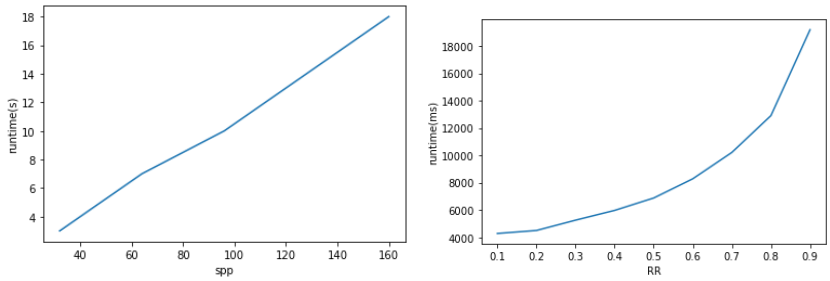


Figure 4: Experimental plots to show the relationship between computational time and *spp* or *RR*

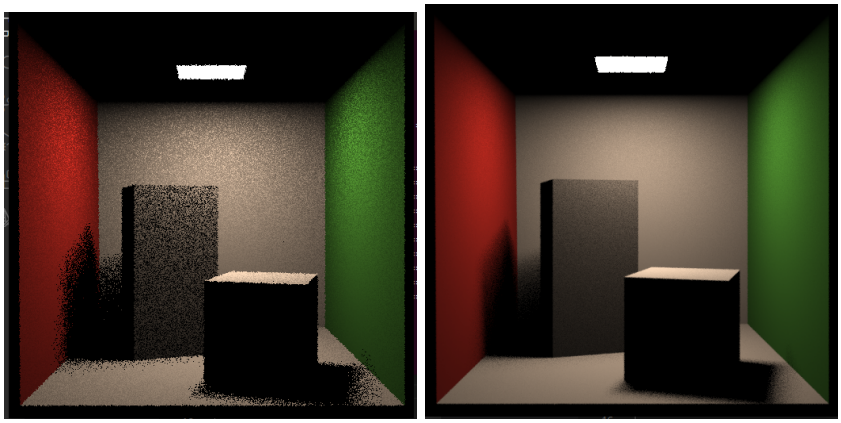

Figure 5: Rendered with no reflections at all, difference with spp=1 and spp=32, monte-carlo integration fails to predict true value when spp=1

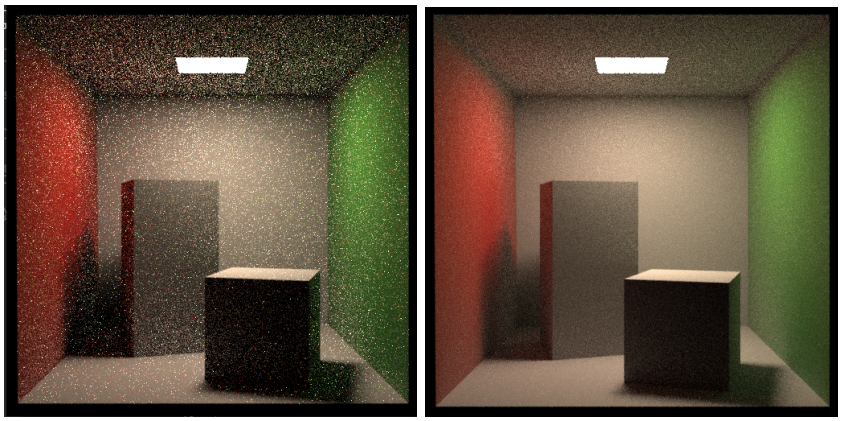

Figure 6: Rendered with RR=0.1 and RR=0.9, noise reduced as RR increases, for the correctness of integrating reflections.