Early Film Simulation

Maria Jose Cepeda Garcia

April 30, 2018

Abstract

In this paper, I explore how to apply procedural textures for simulation of graininess and scratches on a scene. Furthermore, I implement a photographic tone operator. The objective of this project is to provide the silent movie appearance to a synthetic image.

1 Introduction

In this project, I explore the photographic development process for the simulation of black and white photographic prints [2]. Since this process fits in the tone reproduction stage, I incorporate the algorithms described here to my ray tracer that I have been working on throughout the course. My goal is to recreate the same image used for our ray tracer checkpoints as a black and white image with the appearance of a silent movie. The old movie appearance process consists of two phases; 1) the exposure phase, where I add the grain and the scratches to the scene, and 2) the tone reproduction phase, where I apply a photographic tone operator to map simulated scene luminance to display luminance.

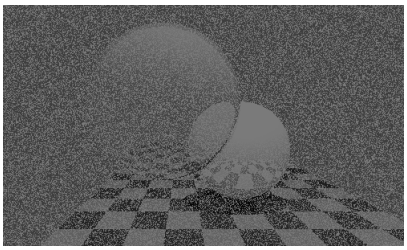
2 Implementation

The implementation of this project is written in Java 8.

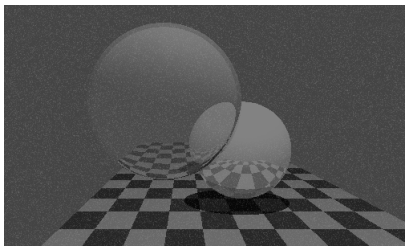
2.1 Exposure

In the exposure phase, I add the grain and the scratches to our scene. The generation of noise that follows a Gaussian distribution is often used to accurate model film grain [2]. For this project, I create a Gaussian noise generator that adds independent noise values to each pixel in the scene. To control the amount of grain in our scene, the noise generator receives as a parameter the variance for the Gaussian distribution. The mean value is set to zero by default. Setting the variance value close to one results in images with lots of grain (Figure 1a). If we want a more subtle grain, we need to specify a variance close to zero (Figure 1b). Figure 2 depicts a scene from The Incredibles animation movie where grain is also added to get the silent movie appearance.

In a first attempt to simulate scratches on the scene, I implemented the generation of a random number of straight lines with random positions and I added them to my ray tracer scene. Unfortunately, the result was not very convincing. Film scratches usually appear like vertical lines across the entire



(a) Gaussian noise, variance = 1



(b) Gaussian noise, variance = 0.08

Figure 1: Example of Gaussian noise with different variance values

Figure 2: The Incredibles movie, example of grain and scratches simulation

scene (e.g. Figure 2 shows a scratch in the middle of the scene), but they are not really sharp straight lines. Some of them are curved, others have some slope. To draw lines with those characteristics, I implement Perlin noise [3]. The idea is to generate lines with a fixed x coordinate position and a random y coordinate for each point of the line. The y coordinate is the result of adding Perlin noise to a given y coordinate.

My implementation of Perlin noise for one dimension consists of two methods; 1) the pseudo-random number generator (PRNG), and 2) the interpolation method. The PRNG generates random numbers using a seed. That seed guarantees that for a giving number, we will obtain the same random number. In Java, it is possible to set the seed value to the method Math.random(). The interpolation method will create a new y coordinate between two given values. To get smooth curved lines, I implement the cosine interpolation.

Algorithm 1 interpolation (float *a*, float *b*, float *blend*)

▷ Input: (a) lower bound, (b) upper bound, (c) Value between 0 and 1
▷ Output: Value between a and b

```
1: double theta = blend * Math.PI;
2: float f = (float) (1f - Math.cos(theta)) * 0.5f;
3: return a * (1f - f) + b * f;
```

There are two parameters that characterize the shape of a line; 1) the amplitude, is the distance from the right to the left of the wave (vertical lines), and 2) the wavelength, the distance between the peak of two consecutive waves. If we want a line with short twisted curves, amplitude and wavelength values must be small. In our case, since film scratches look almost straight lines, we draw lines with large values for both parameters. Figure 3 shows an example of a scratch generated using my implementation of Perlin noise. In this case, the x coordinate is 400, the length of the line is equal to the height of the canvas, the amplitude is 50, and the wavelength is 600.

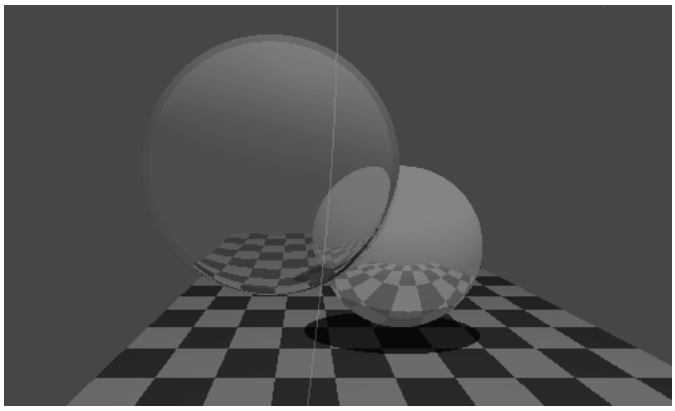
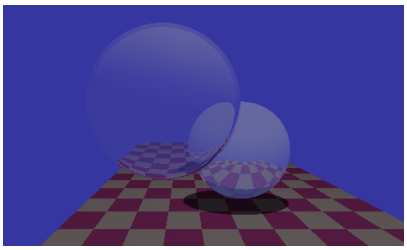


Figure 3: Scratches simulation using Perlin Noise 1D

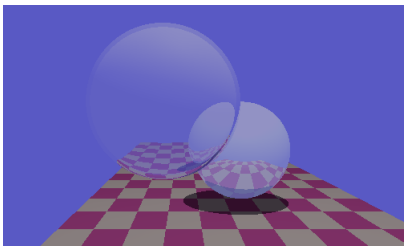
2.2 Tone reproduction

For tone reproduction, I use Reinhard's global mapping operator [4]. This operator applies photographic techniques to map simulated scene luminance to display luminance. Since we are following a photographic development process to simulate old movie appearance, this is an appropriate tone operator to implement. There are three terms to consider in Reinhard's tone operator; 1) the middle-grey color, that represents the middle brightness region of a scene, 2) the key value, that indicates if the simulated scene is bright, normal, or dark, and 3) the illuminance range, the difference between the highest and lowest scene luminance. In general, the middle-grey color is mapped to the Zone V in the Zone System. In a simplified implementation of this tone operator, Zone V value is usually set to 18% of grey and the key value is the log average scene luminance. In order to get a more photographic appearance, I implement this global mapping operator so you can specify the key value and the middle-grey color. Furthermore, you can specify the maximum luminance value in the scene (Lmax). Specifying different values for these three terms, we can achieve different effects. Figure 4 shows the same scene choosing different values for these terms. As you can see, using higher values for Zone V and Lmax, the scene becomes brighter. Choosing different middle-grey colors, the color appearance of the entire scene change.

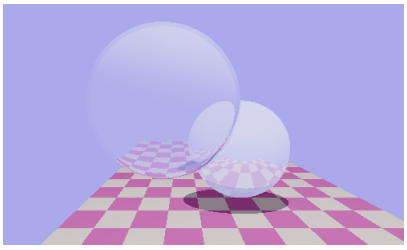
For this version, Reinhard's dodging-and-burning printing technique is not implemented.



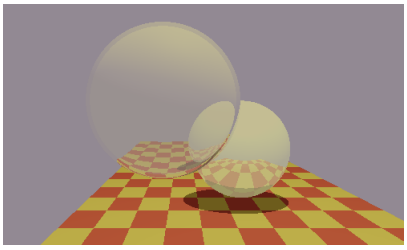
(a) Zone V = 9% of grey, middle-grey = Blue, Lmax= 100 nits



(b) Zone V = 18% of grey, middle-grey = Blue, Lmax= 100 nits



(c) Zone V = 18% of grey , middle-grey = Blue, Lmax= 1000 nits



(d) Zone V = 18% of grey , middle-grey = Red, Lmax= 100 nits

Figure 4: Example of my raytracer image using Reinhard's tone operator with different values for Zone V, middle-grey color, and Lmax parameters.

3 Results

In this section, I present some images generated for my solution (Figure 5). In all images, Reinhard's Zone V is 18% of gray and Lmax is 100 nits. Furthermore, the number of scratches is eight in all images. Although the scratches have the same x coordinate position in all the images, they look different. That is due to the random Perlin noise added to the y coordinate.

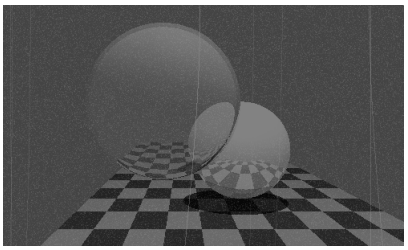
4 Conclusions & Future work

I explore the process of simulating silent movies following a photographic approach. Although my implementation is a simplified version of what it is really done, it is able to simulate old movies adequately. However, further work must be done to generate more realistic scratches. Using a procedural texture to define the scratch shape is not enough. It also necessary to consider the light reflection at every point of the scratch as well as the scratch geometry [1].

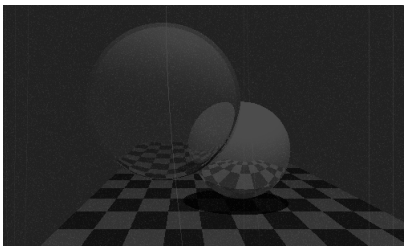
References

[1] C. Bosch, X. Pueyo, S. Mérillou, and D. Ghazanfarpour. A physically-based model for rendering realistic scratches. *Computer Graphics Forum*, 23(3):361–370.

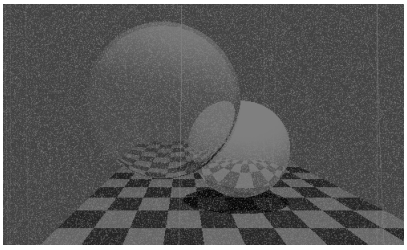
[2] J. Geigel and F. K. Musgrave. A model for simulating the photographic development process on digital images. In *Proceedings of the 24th Annual*



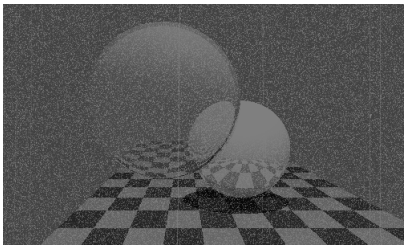
(a) Reinhard's middle-grey = Blue, Gaussina noise variance = 0.008



(b) Reinhard's middle-grey = Red, Gaussina noise variance = 0.008



(c) Reinhard's middle-grey = Blue, Gaussian noise variance = 0.5



(d) Reinhard's middle-grey = Blue, Gaussian noise variance = 0.8

Figure 5: Example of images generated using my solution

Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '97, pages 135–142, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[3] K. Perlin. An image synthesizer. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, pages 287–296, New York, NY, USA, 1985. ACM.

[4] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. Photographic tone reproduction for digital images. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 267–276, New York, NY, USA, 2002. ACM.