Distribution Ray Tracing

Team 8

JAHNVI KUMARI

1 INTRODUCTION

Due to the particle nature of light and non-zero light source area, many phenomena like non-sharp edges, soft shadows and motion blur are experienced in nature. A simple way to achieve these visual effects is to oversample in a regular ray tracer which would average many points on the object for a single pixel. A better technique which reduces artefacts like Moire patterns is to distribute the oversampled rays spatially. The idea of distributing the direction of the rays need not be restricted to spatial sampling and can be any analytic function. This is called distribution ray tracing, and it can be used to incorporate fuzzy phenomena.
The aim of this project is to implement a distributed ray tracer and demonstrate antialiasing, soft shadows, depth of field and motion blur.

2 LITERATURE REVIEW

Cook and Porter [2] introduced the distributed ray tracing algorithm in 1984 and demonstrated it's various application. They discussed some of its applications like shading where they point sample the function's value by distributing the rays - Illumination rays according to the illumination function L and Reflected rays according to the reflectance function R. They also discuss translucency and gloss which require integral of the transmitted and reflected lights respectively. They also discuss the role of secondary rays in implementing penumbras similar to what I will do and gave the idea of distributing the sample points in time to solve the motion blur problem. Marshner and Shirley, in their book [7], have explained the concepts of distributed ray tracing and given pseudocodes for antialiasing and soft shadows. They have also discussed how to accelerate the ray tracer and the underlying precision tradeoffs to make the computations fast and efficient. A few effects they have discussed in their work are as follows:-

(1) Antialiasing: Spatially sampling the rays to jitter or antialias. They discuss stratified sampling which is a hybrid of uniform and random sampling strategies and making it computationally inexpensive by sampling only for the edges.

(2) Soft shadows: Sampling the solid angle of the light sources. To feasibly simulate an area light source, they jittered the light rays and the viewing rays independently so as to form soft shadows and avoid clear belts of shadows.

(3) Depth of field: Sampling the viewing rays from the lens. The aim is to capture the rays going towards the focal plane to simulate a focusing effect similar to using a camera lens.

(4) Motion blur. Time sampling the viewing rays. To simulate the aperture being open for some time interval during which the object moves, they setup a time variable ranging from T0 to T1 and choose a random time for each viewing ray.

Author's address: Jahnvi Kumari, jahnvi19469@iiitd.ac.in.

The direct lighting component in distribution ray tracing is calculated by performing a numerical integration, accounting for all potentially visible luminaires which leads to a high time complexity. Further research on this topic is primarily towards lowering the costs overheads of integration and oversampling and introducing interactivity.

Keller et al. [4] worked towards achieving low sampling rates using interleaved sampling, which tries to replace low-frequency artifacts with dithering-like structured error in screen space. Lee et al. [5] showed that the number of rays required does not depend on the dimensionality of the space being sampled but only on the variance of the multi-dimensional image function and optimized the Distributed Ray tracing through the use of statistical testing and stratified sampling. Wang [9] discussed the method for calculating the contribution from spherical and triangular luminaires. With the development in compute power, DRT has become a default and is used in most rendering applications like those presented by Wald et al. [8] and Buolos et al. [1]. The former work describes interactive ray tracing where the image is rendered in real-time while users rotate or zoom in them. Buolos et al compared interactive whitted ray tracer based renderer with interactive DRT based ones.

3 MILESTONES

S. No.	Milestone	Member	Milestone achieved?
	Mid evaluation		
0	Literature Review and code setup	Jahnvi	✓
1	Implement Antialiasing	Jahnvi	✓
2	Implement depth of field	Jahnvi	✓
	Final evaluation		
3	Implement soft shadows	Jahnvi	✓
4	Implement motion blur	Jahnvi	✓
5	(If time permits) Glossy reflection	Jahnvi	

4 APPROACH

Antialiasing was performed similar to the approach mentioned in [7]. A pixel was divided into $n \times n$ cells and n^2 square rays were emanated from a random distance $\xi$from the left boundary of each of the cells.

Depth of field was implemented in an approach similar to the one described in [6]. Focal plane is calculated as a spherical plane at distance of focal length from the camera's eye. The objects on or near the focal plane would be in focus and those nearer or further would be out of focus. To model haziness, a new origin is calculated by perturbing the ray origin according to the aperture. Angularly distributed rays are then emanated towards the focal plane from this new origin and the average colour is taken as the pixel colour.

Soft shadow was implemented by modelling an area light source by a circle. In this case, the intersection of the shadow ray was tested with this area. An alternative technique I tried was to distribute the shadow rays along the light direction which lead to shadow falling on points which were previously fully visible. I've used Phong shading as taught in class.

Motion blur was implemented by ray tracing a pixel repeatedly over several frames ie. distribution across time and taking the average colour. This lead to a fainter color of a pixel from which the object has moved. The technique was similar to as demonstrated in [3].

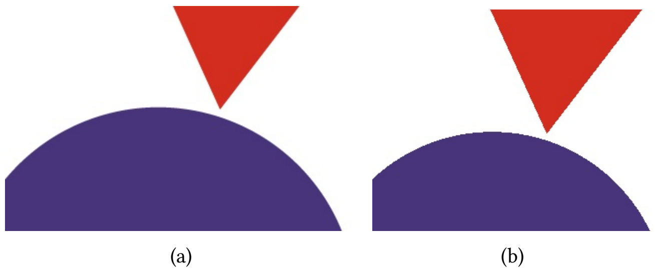


Fig. 1. (a) With Antialiasing (b) Without Antialiasing

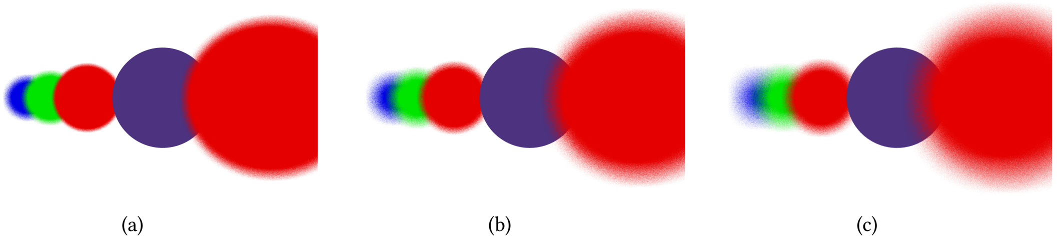


Fig. 2. The figures having larger apertures experience large distortions. Keeping the focal length constant at 20 and changing the apertures, (a) represents aperture of 1 (b) aperture of 2 and (c) aperture of 3.

5 ALGORITHM ANALYSIS

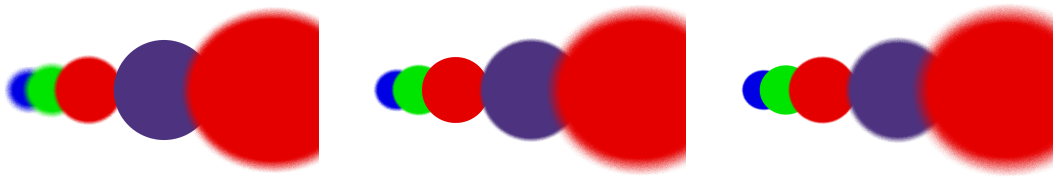
A ray tracer's complexity depends upon various factors like the shading used, shadow rays etc. Hence let $O(r)$ be the complexity of a ray tracer where $r = h * w$ for a pathtracer, h being the height, w the width.

A naive implementation of antialiasing would be to divide each pixel into $\vec{n} \times \vec{n}$ subpixels and emanate a ray from each of the subpixels. This is $O(r * n^2)$. We can improve the performance by checking if a pixel is an edge. I have implemented it by checking if the euclidean distance between two pixels is greater than a threshold. In a column-order rendering of pixels, colour dissimilarity with the left and top pixels can be checked. The worst-case complexity remains the same but practically, the performance improves significantly as shown in the results. We can also memoize the left and top pixel values which leads to a space complexity of $O(h * w)$ but speeds up the code further.

The time complexity of depth of field is $O(r * n)$. Combining the two can lead to a worst case of $O(r * n^3)$.

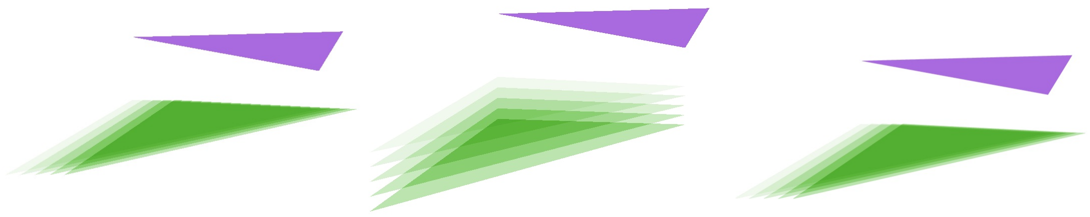
6 RESULTS

As shown, the antialiased image looks much smoother than a simple ray traced image. Antialiasing on all pixels takes 7x the time required to antialias on the edges. The results for depth of field are also shown above. In the images, spheres of the same radius are placed at varying points on the z axis; the ones near to the focal plane are clearer than those away from it. The results for soft-shadow and blur due to motion are as shown.



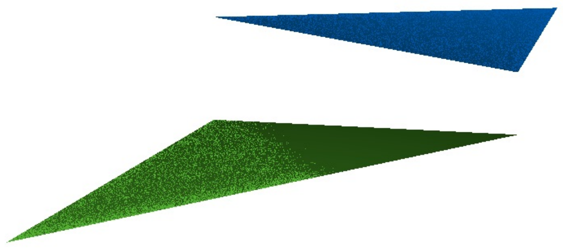
(a) (b) (c)

Fig. 3. As the focal length increases, the spheres farther to the image plane becomes focused (a) represents focal length of 20 (b) focal length of 30 and (c) focal length 40.



(a) (b) (c)

Fig. 4. The lower triangle is in motion while the upper triangle is stationary. Fig. (a) shows motion blur as the object moves in the x direction and (b) as the object moves in y direction (c) both blurring and antialiasing



(a)

Fig. 5. Soft shadow of the above triangle falling on the lower one.

REFERENCES

[1] Solomon Boulos, Dave Edwards, J Dylan Lacewell, Joe Kniss, Jan Kautz, Peter Shirley, and Ingo Wald. 2007. Packet-based whitted and distribution ray tracing. In *Proceedings of Graphics Interface 2007*. 177–184.

[2] Robert L Cook, Thomas Porter, and Loren Carpenter. 1984. Distributed ray tracing. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. 137–145.

[3] Eagle2765. 2016. https://github.com/eagle2765/Distribution-Ray-Tracing

[4] Alexander Keller and Wolfgang Heidrich. 2001. Interleaved sampling. In *Eurographics Workshop on Rendering Techniques*. Springer, 269–276.

[5] Mark E Lee, Richard A Redner, and Samuel P Uselton. 1985. Statistically optimized sampling for distributed ray tracing. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*. 61–68.

[6] Eric Lopez. 2018. https://medium.com/@elope139/depth-of-field-in-path-tracing-e61180417027

[7] Peter Shirley, Michael Ashikhmin, and Steve Marschner. 2009. *Fundamentals of computer graphics*. AK Peters/CRC Press.

[8] Ingo Wald, Philipp Slusallek, and Carsten Benthin. 2001. Interactive distributed ray tracing of highly complex models. In *Eurographics Workshop on Rendering Techniques*. Springer, 277–288.

[9] Changyaw Wang. 1992. Physically correct direct lighting for distribution ray tracing. In *Graphics Gems III (IBM Version)*. Elsevier, 307–313.