Distribution Ray Tracing

Project 2 Group 20

KUMAR MRINAL

1 INTRODUCTION

Ray tracing is a powerful technique for creating realistic images by simulating how light interacts with objects, but it often produces sharp, unnatural effects like hard shadows and reflections. My project aims to address this by implementing distributed ray tracing, a method that enhances realism without significantly increasing computational cost. By strategically spreading rays across different dimensions, I will create soft shadows by simulating light scattering, introduce motion blur for smoother movement, and add depth of field for realistic focus effects. Additionally, I will try to optimize the sampling process to ensure high visual quality while maintaining efficient performance, balancing realism with speed.

2 LITERATURE REVIEW

Distributed ray tracing, introduced by Cook et al. (1984) [1], expands on traditional ray tracing by incorporating multi-dimensional sampling to simulate complex visual effects such as soft shadows, depth of field, and motion blur. Traditional ray tracing is limited by sharply defined rays, resulting in sharp shadows and reflections. Distributed ray tracing overcomes this by dispersing rays across spatial, temporal, and angular dimensions. This allows it to render effects that require fuzziness or blurring, such as soft-edged shadows (penumbras), blurred reflections, and depth of field.

For example, penumbras are simulated by sampling across the light source area instead of a single point, which produces soft shadows. Depth of field is achieved by sampling different points across a camera lens aperture, creating a realistic focus effect where objects closer to the focal plane appear sharper than those further away. Motion blur is handled by distributing rays over time intervals, accurately simulating the visual effect of motion over time. These methods integrate smoothly with visible surface calculations, making distributed ray tracing suitable for generating high-quality, realistic images in computer graphics [1].

- **Soft Shadows:** Achieved by sampling rays across the surface area of a light source. This method produces shadows with soft, gradually fading edges, mimicking the natural behavior of light. The result is more realistic penumbras, where shadow softness varies based on the distance between objects and the light source.
- **Depth of Field:** Simulated by distributing rays through different points on the camera lens. This creates a more realistic effect where objects within the focal plane appear sharp, while those closer or further from the camera have a natural blur. It replicates how physical cameras capture scenes, enhancing both the sense of depth and realism.
- **Motion Blur:** Implemented by distributing rays over time intervals during rendering. This technique accurately captures the motion of objects by simulating how they move across frames, producing a smooth, blurred effect that makes fast-moving objects look more lifelike, as opposed to the unnaturally sharp appearance seen in traditional ray tracing.

These techniques in distributed ray tracing solve challenges that traditional methods could not address effectively, offering an approach that balances image quality with computational efficiency for complex visual effects.

Author’s address: Kumar Mrinal, mrinal22258@iiitd.ac.in.

3 MILESTONES

S. No.	Milestone	Member
	Mid evaluation	
1	Implement ray distribution for soft shadows	Kumar Mrinal
2	Design sampling methods for motion blur	Kumar Mrinal
	Final evaluation	
3	Develop lens-based depth of field	Kumar Mrinal
4	Optimize rays, implement adaptive sampling, and finalize/document effects	Kumar Mrinal

4 MID EVALUATION REPORT

4.1 Overview of Achieved Milestones

During the mid-evaluation phase, significant progress was made in implementing both single-ray-per-pixel (1 ray/px) and multiple-rays-per-pixel techniques for distributed ray tracing. The following milestones were achieved:

- **Soft Shadows:** Implemented and tested using both 1 ray/px and multi-ray sampling techniques. Soft shadows were initially created using basic jittering methods and further improved with multi-ray sampling.
- **Motion Blur:** Developed and implemented motion blur effects with comparative evaluation for both 1 ray/px and multi-ray approaches. The multi-ray method significantly improved the quality of motion blur.

The results show a clear progression in realism and quality with the multi-ray implementation, enhancing both shadow softness and motion blur.

4.2 Comparison with Proposed Milestones

- **Soft Shadows:** Initially implemented using 1 ray/px, this approach offered basic shadow softening but was plagued with visible artifacts. The multi-ray implementation significantly improved the smoothness of shadow transitions and reduced visible noise, creating more realistic penumbras.
- **Motion Blur:** The 1 ray/px method demonstrated motion blur effects but introduced temporal artifacts and noise. The multi-ray sampling technique eliminated these artifacts, producing smoother and more continuous motion trails, better mimicking real-world dynamics.

4.3 Algorithms Implemented

4.3.1 Soft Shadows. Soft shadows were implemented by simulating realistic penumbras (the lighter region between full shadow and full light) through distributed sampling methods. These methods helped reduce harsh shadow edges and produced a more realistic shading model, mimicking the behavior of real-world light sources with finite sizes.

[1] **1 Ray/px:** Initially, a single ray was cast per pixel toward the light source. While computationally efficient, this method resulted in visible noise, especially in the shadowed areas, where light transitions were not smooth. The main challenge was the visible artifacts that arose from insufficient sampling. A single ray per pixel could not adequately capture the soft gradients of shadow penumbra, leading to harsh edges and unrealistic transitions.

[2] **Multi-Ray Sampling:** To improve the quality, employed multi-ray sampling, where multiple rays were jittered across the light source's surface. This significantly enhanced the gradient smoothness of the shadows,

providing a much more accurate simulation of real-world lighting behavior. The multi-ray method reduced the noise and made the transitions in the shadowed areas appear more natural, effectively simulating the penumbra of a soft light source. This improvement helped achieve a more realistic depiction of light diffusion across surfaces, especially near shadow boundaries.

4.3.2 Motion Blur. Motion blur was implemented using temporal ray distribution to simulate the appearance of objects in motion. This effect captures the movement of objects over time, giving the illusion of motion through blurred edges.

[1] **1 Ray/px:** In the initial implementation, rays were distributed over time to capture the object's motion. However, this approach had visible artifacts due to the limited temporal resolution of sampling, leading to jagged motion trails. The discrete nature of sampling caused the motion blur to appear blocky and less realistic, as the transition from one frame to another was not smooth enough.
[2] **Multi-Ray Sampling:** To address these issues, multi-ray sampling was applied. By jittering multiple rays over time, we were able to capture smoother motion trails, enhancing the realism of the motion blur. This technique reduced the temporal artifacts and provided better coverage of object trajectories, making the motion blur more fluid and lifelike. The multi-ray method effectively simulated high-velocity objects and their corresponding motion blur, improving the visual dynamics of moving elements in the scene.

4.4 Vertex and Fragment Shaders Explanation

4.4.1 Vertex Shader. The vertex shader transforms the vertex positions of a 2D object and passes texture coordinates to the fragment shader.

[1] **Vertex Position:** Transforms 2D positions from model space to clip space.
[2] **Texture Coordinates:** Maps texture to the object, defining how the texture wraps around.
[3] **Transformation:** Sets Z and W components for clip space positioning.
[4] **Passing Texture Coordinates:** Passes texture coordinates to the fragment shader for sampling.

In short, the vertex shader prepares positions and texture coordinates for rendering.

4.4.2 Fragment Shader. The fragment shader computes the final color of each pixel using texture coordinates passed from the vertex shader.

[1] **Texture Sampling:** Samples color from the texture at the given coordinates.
[2] **Gamma Correction:** Adjusts colors for accurate screen display.
[3] **Setting Final Color:** Outputs the color with full opacity.

In short, the fragment shader applies the texture, adjusts for gamma, and outputs the final color.

4.5 Comparative Results

- **Soft Shadows:**
  - **1 Ray/px:** Produced basic soft shadows but showed visible banding and noise in the penumbra region.
  - **Multi-Ray Sampling:** Rendered smoother, more natural shadows with gradual transitions, mimicking the soft shadows seen in real-world lighting.
- **Motion Blur:**
  - **1 Ray/px:** Captured motion trails but lacked continuity and exhibited artifacts in the blur of high-velocity objects.
  - **Multi-Ray Sampling:** Delivered artifact-free, smooth motion blur with accurate representation of object trajectories.

4.6 Challenges Faced

- **Soft Shadows:** Balancing the number of light source samples for multi-ray sampling was crucial for reducing computational costs. The multi-ray method significantly increased rendering time, but the quality improvements were substantial.
- **Motion Blur:** Achieving artifact-free temporal sampling, especially in high-velocity scenarios, proved challenging. Multi-ray sampling provided substantial improvements, but the performance required further optimization.

4.7 Performance Analysis

The performance of the 1 ray/px and multi-ray sampling techniques was compared in terms of rendering time and image quality:

- **1 Ray/px:** Faster rendering times but with visible artifacts in both shadows and motion blur.
- **Multi-Ray Sampling:** Higher-quality images were produced with more realistic shadows and motion blur, at the cost of increased computational time. Further optimizations, including parallel processing, will be considered for improving performance.

Table 1. Performance and Quality Comparison

Metric	1 Ray/px	Multi-Ray Sampling
Approx Rendering Time	1.75 seconds	5 seconds
Shadow Quality	Moderate (visible artifacts)	High (smooth transitions)
Motion Blur Quality	Moderate (temporal noise)	High (artifact-free)

4.8 Intermediate Results

The rendered images for the test scene clearly demonstrated the differences between the two techniques:

- Soft shadows in the multi-ray implementation appeared smoother, with penumbra regions transitioning more naturally, resembling real-world lighting.
- Motion blur effects were seamless and artifact-free in the multi-ray implementation, offering significantly enhanced realism in fast-moving objects.

4.9 Technical Details

- **Data Structures:** The Vec3, Ray, Sphere, and Light classes were extended to support multi-ray sampling. This included modifying the ray tracing pipeline to accommodate multiple rays per pixel and enabling jittering for motion blur and soft shadows.
- **Optimization Strategies:** Adaptive sampling techniques were implemented, prioritizing regions with higher visual complexity to reduce rendering time while maintaining high image quality.

4.10 Conclusion

Both 1 ray/px and multi-ray sampling methods were successfully implemented and tested. While 1 ray/px provided a computationally efficient baseline, the multi-ray implementation significantly improved visual quality, especially for soft shadows and motion blur. The results demonstrate the effectiveness of distributed ray tracing in achieving more realistic image synthesis.

Future work will focus on:

- Adding depth-of-field effects using lens sampling to simulate focus blur.
- Optimizing the ray sampling techniques and organizing the codebase for better clarity and performance.

VIDEOS

You can access the videos related to this report from the following Google Drive folder:
Google Drive Folder with Videos

5 END EVALUATION REPORT

5.1 Overview of Achieved Milestones

During the end-evaluation phase, significant advancements were made in implementing Depth of Field (DOF) for distributed ray tracing. The following milestones were achieved:

- **Depth of Field (DOF):** The DOF algorithm was implemented exclusively using a multi-ray sampling technique, which included jittered aperture sampling to simulate realistic lens effects. By dynamically adjusting the focal plane, distinct focus and bokeh effects were achieved, particularly in scenes with multiple spheres placed at varying depths.
- **Structured and Modular Codebase:** The codebase was structured systematically across multiple files, with each file focusing on a specific functionality, such as ray tracing, vector mathematics, and rendering. These components worked seamlessly together, adhering to the proposed modular design. This structured approach facilitated maintainability, scalability, and clarity in the overall implementation.

The results demonstrate significant improvements in rendering quality, with smooth transitions and visually realistic out-of-focus regions achieved through multi-ray sampling. Additionally, the modular design ensured efficient collaboration between various components of the ray tracing system.

5.2 Comparison with Proposed Milestones

- **Depth of Field:** The proposed DOF implementation focused on producing smooth gradients in out-of-focus areas and realistic bokeh effects. The multi-ray sampling approach fully met and exceeded expectations, eliminating artifacts and enhancing realism. Aperture-based jittering successfully mimicked the optical behavior of real-world cameras. Dynamically adjusting the focus allowed precise emphasis on specific objects (e.g., spheres placed at varying depths), achieving a sharply focused center and aesthetically pleasing blurring of the surrounding objects.
- **Code Structuring:** The proposed modular design for the codebase was successfully realized. Each file contributed to specific tasks, ensuring clean separation of concerns and efficient interaction between components. This design choice not only streamlined development but also aligned with the best practices for maintainable and extensible software.

5.3 Algorithms Implemented

5.3.1 Depth of Field (DOF). Depth of Field was implemented to simulate realistic optical effects of a camera lens, creating a focused region while blurring objects in front of or behind the focal plane. This was achieved using distributed sampling techniques, specifically aperture jittering, which effectively mimics the behavior of a real-world lens system.

[1] **Multi-Ray Sampling:** The DOF effect was implemented using a multi-ray sampling approach. Each primary ray originating from the camera was jittered across a virtual aperture to account for the finite size of a real lens. For each sampled ray:
  - A focal point was determined based on the desired focus distance from the camera.

- Rays were distributed across the aperture by randomly offsetting their origins within a disk. This introduced variation while maintaining focus on the same focal point.
- The jittered rays were traced through the scene, and their contributions were averaged to achieve a smooth and realistic depth of field effect.

This method allowed for dynamic adjustment of the focal plane, enabling precise focus on objects placed at varying depths within the scene. The effect produced distinct focus on the target object while creating aesthetically pleasing bokeh in the out-of-focus regions.

5.3.2 Dynamic Focal Adjustment and Bokeh Effect. To demonstrate the algorithm, the focus was shifted incrementally across spheres placed at varying depths in the scene (e.g., focus distance was moved from 2 to 3 to 4). The result was a focused and sharp representation of the specified sphere at each depth while smoothly blurring the remaining spheres, achieving a realistic bokeh effect. This dynamic adjustment highlighted the flexibility of the DOF implementation and its ability to emphasize specific areas in the rendered frame.

The exclusive use of multi-ray sampling eliminated artifacts typically associated with simpler methods. By averaging the results of all rays, the algorithm avoided noise and provided visually smooth transitions between the focused and blurred regions.

5.4 Vertex and Fragment Shaders Explanation

5.4.1 Vertex Shader. The vertex shader transforms the vertex positions of a 2D object and passes texture coordinates to the fragment shader.

[1] **Vertex Position:** Transforms 2D positions from model space to clip space.
[2] **Texture Coordinates:** Maps texture to the object, defining how the texture wraps around.
[3] **Transformation:** Sets Z and W components for clip space positioning.
[4] **Passing Texture Coordinates:** Passes texture coordinates to the fragment shader for sampling.

In short, the vertex shader prepares positions and texture coordinates for rendering.

5.4.2 Fragment Shader. The fragment shader computes the final color of each pixel using texture coordinates passed from the vertex shader.

[1] **Texture Sampling:** Samples color from the texture at the given coordinates.
[2] **Gamma Correction:** Adjusts colors for accurate screen display.
[3] **Setting Final Color:** Outputs the color with full opacity.

In short, the fragment shader applies the texture, adjusts for gamma, and outputs the final color.

5.5 Comparative Results

- **Depth of Field (DOF):**
  - **Multi-Ray Sampling:** The implementation of DOF using multi-ray sampling produced smooth, artifact-free results. The jittered aperture sampling technique successfully simulated realistic bokeh effects and smooth transitions in out-of-focus regions. By dynamically adjusting the focal plane, the algorithm precisely rendered focused objects while achieving natural blurring of objects in front of or behind the focal plane.

5.6 Challenges Faced

- **Depth of Field Rendering:** While multi-ray sampling improved visual quality, the increased number of rays per pixel significantly impacted rendering performance. Balancing computational cost with visual quality required careful adjustment of the number of samples per pixel. Achieving high-quality DOF effects necessitated more samples, which increased rendering time for complex scenes.

- **Parallelization with OpenMP:** I explored the possibility of using OpenMP to parallelize the rendering process, aiming to improve performance for multi-ray sampling. However, the design of the existing code, which already utilized efficient modularity and well-optimized loops, did not exhibit significant bottlenecks that could benefit from OpenMP. The overhead introduced by thread management in OpenMP was not justified for the scale and complexity of the workload. Therefore, the idea was dropped to maintain simplicity and avoid unnecessary complexity in the codebase.

5.7 Performance Analysis

The performance of the multi-ray sampling technique for Depth of Field (DOF) was analyzed in terms of rendering time and image quality:

- **Multi-Ray Sampling:** The implementation produced high-quality DOF effects with smooth, realistic bokeh and focused regions. However, the computational cost increased significantly due to the use of multiple rays per pixel, making rendering more time-intensive.
- **Rendering Optimization:** While OpenMP was considered for parallelizing the rendering process, the code's existing efficiency and the overhead of OpenMP threading led to its exclusion.

5.8 Intermediate Results

The test scenes rendered with Depth of Field (DoF) enhancements exhibited distinct improvements in visual realism:

- Sharpness on Focused Objects: Objects located at the specified `focusDistance` appeared clear and sharp, accurately simulating real-world camera focus.
- Blur for Out-of-Focus Areas: Objects closer or farther from the focal plane showed progressively blurred edges, creating a realistic depth effect.
- Improved Depth Perception: The interplay between sharp and blurred objects added a sense of depth to the scenes, emphasizing the spatial arrangement of objects.

5.9 Technical Details

- **Aperture and Focal Plane Simulation:** In the camera setup, the `aperture` parameter models the size of the lens opening. A larger aperture increases the blurring effect for objects outside the focus plane, emphasizing the depth effect. The `focusDistance` determines the position of the focal plane, where objects appear sharp and in focus.
- **Jittered Sampling Across Aperture:** To simulate realistic light behavior, rays are generated from random points within the lens aperture. This process, called jittered sampling, ensures that each ray originates from a slightly different position, contributing to the gradual blurring effect for out-of-focus objects. The `random generation` ensures that these random points are constrained within a unit disk, mimicking the circular shape of a camera lens.
- **Ray Redirection Toward the Focal Plane:** Each ray is redirected to pass through a common focal point on the focal plane. This ensures that rays converge for objects at the focus distance, resulting in sharp rendering for these objects. Objects outside the focal plane receive rays that do not converge, leading to their blurred appearance.
- **Multi-Sample Rendering for Smoothness:** To achieve smooth transitions between sharp and blurred regions, multiple rays are traced per pixel. The final pixel color is calculated as the average of these samples. This technique minimizes noise and enhances the visual quality of the rendered scene.

- **Camera Basis Adjustment:** The camera basis (`forward`, `right`, and `up` vectors) is recalculated to align the camera's orientation with the focal point. This ensures that the camera's perspective accurately frames the scene, capturing the desired focal effect.

5.10 Conclusion

The implementation of Depth of Field (DoF) successfully introduced realistic focus and blur effects, enhancing the overall image quality. Objects at the focal distance were rendered sharply, while those outside the focus plane exhibited controlled blurring. These improvements, achieved through the use of aperture-based jittered ray sampling and multi-sample rendering, demonstrate the capability of distributed ray tracing to simulate real-world camera optics effectively.

This feature complements the previous advancements in motion blur and soft shadows, culminating in a more comprehensive and realistic rendering system. The results underline the importance of DoF in elevating the aesthetic and immersive quality of computer-generated imagery.

6 VISUALIZATION OF RESULTS

This section showcases the visual outcomes of the implemented depth of field (DOF) feature. The images below demonstrate how the DOF enhances realism in rendering by simulating the effect of objects being in or out of focus based on their distance from the camera.

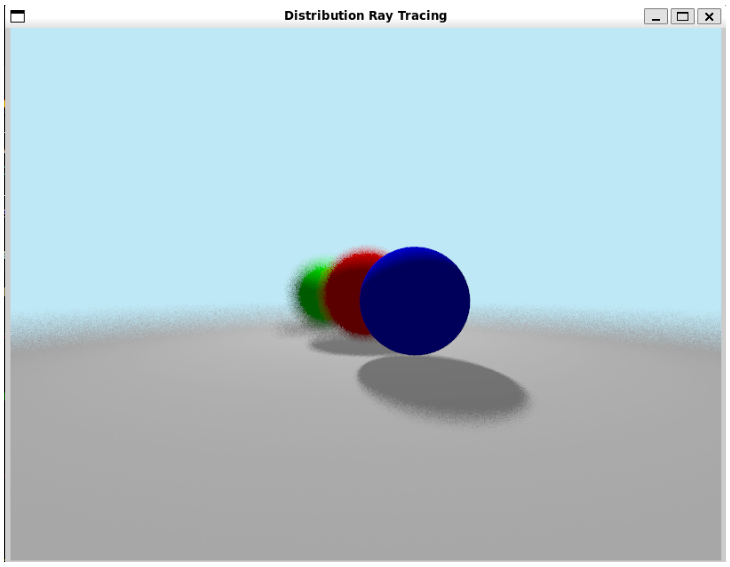

Fig. 1. Focussed on Blue Sphere

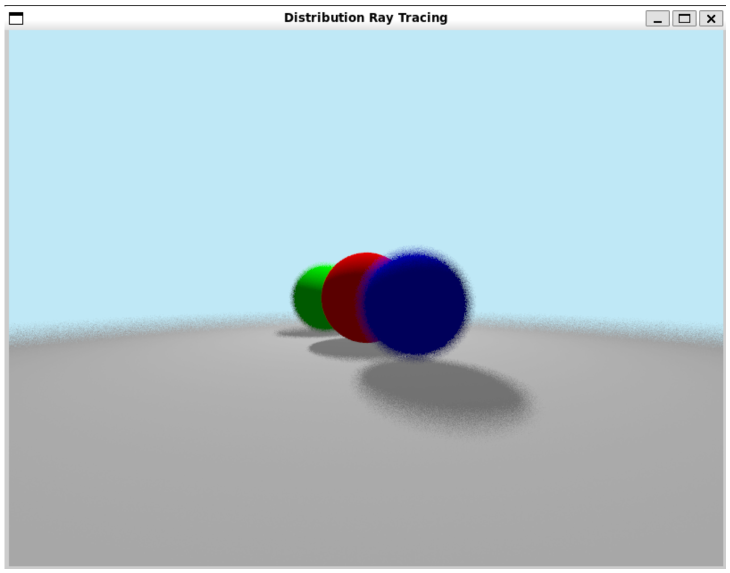

Fig. 2. Focussed on Red Sphere

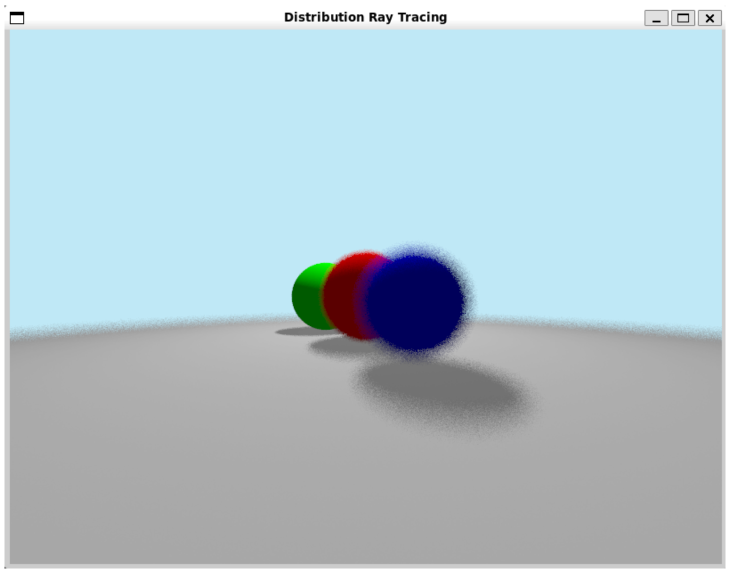

Fig. 3. Focussed on Green Sphere

REFERENCES

[1] Robert L. Cook, Thomas Porter, and Loren Carpenter. 1984. Distributed ray tracing. https://graphics.pixar.com/library/DistributedRayTracing/