Technical University of Denmark

02562 Rendering

# Final Project: Depth of Field

## Stochastic Sampling  Physically Based Materials

**Student:**
Bashar Bdewi (s183356)

**Links:**
Lab journal: https://basharbd.github.io/02562-rendering/
Project implementation: https://github.com/basharbd/02562-rendering

**Date:**
December 19, 2025

# Contents

# Abstract

This project extends a standard WebGPU path tracer by implementing a physically based **Thin Lens Camera Model** to simulate depth of field (DoF). Unlike the traditional pinhole camera model, which renders all objects in infinite focus, the thin lens model introduces a finite aperture and a specific focal plane. This allows for realistic optical effects where a subject can be isolated in sharp focus while foreground and background elements are rendered with physically accurate blur (bokeh).

The renderer is built upon a robust path tracing engine featuring **Next Event Estimation** (Direct Illumination), Russian Roulette termination, and a **BSP-Tree** acceleration structure for efficient geometry traversal. The material system supports Lambertian, Glossy, and Dielectric (Glass) BSDFs, including Fresnel reflectance and volumetric absorption (Bouguer's Law). The final application runs in real-time in a web browser, allowing users to interactively adjust the aperture radius and focal distance to observe the optical changes dynamically.


# 1 Introduction

In computer graphics, the default method for generating images is the "pinhole camera" model. This model assumes an infinitesimally small aperture, resulting in an image where every object, regardless of its distance from the camera, appears perfectly sharp. While computationally simple, this lacks the cinematic and realistic quality of physical photography .

Real cameras use systems of lenses with finite apertures. Light rays originating from a single point on an object pass through different parts of the lens and converge at a specific distance behind it (the focal plane). Objects located at this specific distance appear sharp, while objects closer or farther away create a "Circle of Confusion" on the sensor, resulting in blur

[Image of depth of field diagram] .

The goal of this project is to implement this **Thin Lens Model** within a stochastic path tracing framework. By replacing the singular ray origin with a sampled disk, we can simulate realistic depth of field, significantly enhancing the photorealism of the rendered scene.

# 2 Method

## 2.1 Thin Lens Camera Model

The core of the implementation replaces the deterministic ray generation of the pinhole model with a stochastic process.

1. **Focal Plane Definition:** We define a focal distance $f_{dist}$. A ray is first cast from the center of the lens (the eye position $E$) through the pixel coordinates on the image plane to find a point of perfect focus, $P_{focus}$.

2. **Aperture Sampling:** We model the lens as a disk with radius $R$ (controlled by the aperture parameter). We sample a random point $L_{sample}$ on this disk using concentric disk sampling to ensure a uniform distribution.

3. **Ray Perturbation:** The new primary ray originates from $L_{sample}$ and is directed towards $P_{focus}$.

Mathematically, this perturbation ensures that for any point on the focal plane, all rays from the lens will converge at that exact point (resulting in a sharp image). For points outside this plane, the rays will arrive at slightly different angles, creating the desired blur.

### 2.2 Path Tracing Integration

The depth of field effect relies on the Monte Carlo nature of the path tracer. A single sample per pixel produces a noisy, jittered image (resembling a broken lens). However, as the renderer accumulates hundreds of samples per pixel over time, the random positions on the lens average out, converging to a smooth, high-quality soft focus.

## 3 Implementation Details

The project is implemented in **WGSL**. The rendering pipeline is split into a vertex shader (handling the quad) and a fragment shader (handling the path tracing logic).

### 3.1 Ray Generation Logic

The function `get_camera_ray` implements the thin lens mathematics. It shifts the ray origin based on the sampled disk position and recalculates the direction to intersect the focal point.

```
fn get_camera_ray(ipcoords: vec2f, seed: ptr<function, u32>) -> Ray {
    // 1. Calculate standard pinhole ray vectors
    let v = normalize(p - e);
    let b1 = normalize(cross(v, u));
    let b2 = cross(b1, v);

    // 2. Calculate the focal point
    let focus_plane_dist = length(p - e) * uniforms_f.focus_dist;
    let focus_point = e + v * focus_plane_dist +
                      (b1 * ipcoords.x + b2 * ipcoords.y) * (focus_plane_dist / d);

    // 3. Sample the lens (Aperture)
    let lens_radius = uniforms_f.aperture * 15.0;
    let lens_uv = sample_disk(seed) * lens_radius;
    let lens_offset = lens_uv.x * b1 + lens_uv.y * b2;

    // 4. Construct new ray from lens sample to focus point
    let new_origin = e + lens_offset;
    let new_direction = normalize(focus_point - new_origin);

    return Ray(new_origin, new_direction, 1e-9, 1e9);
}
```

Listing 1: Thin Lens Implementation in WGSL

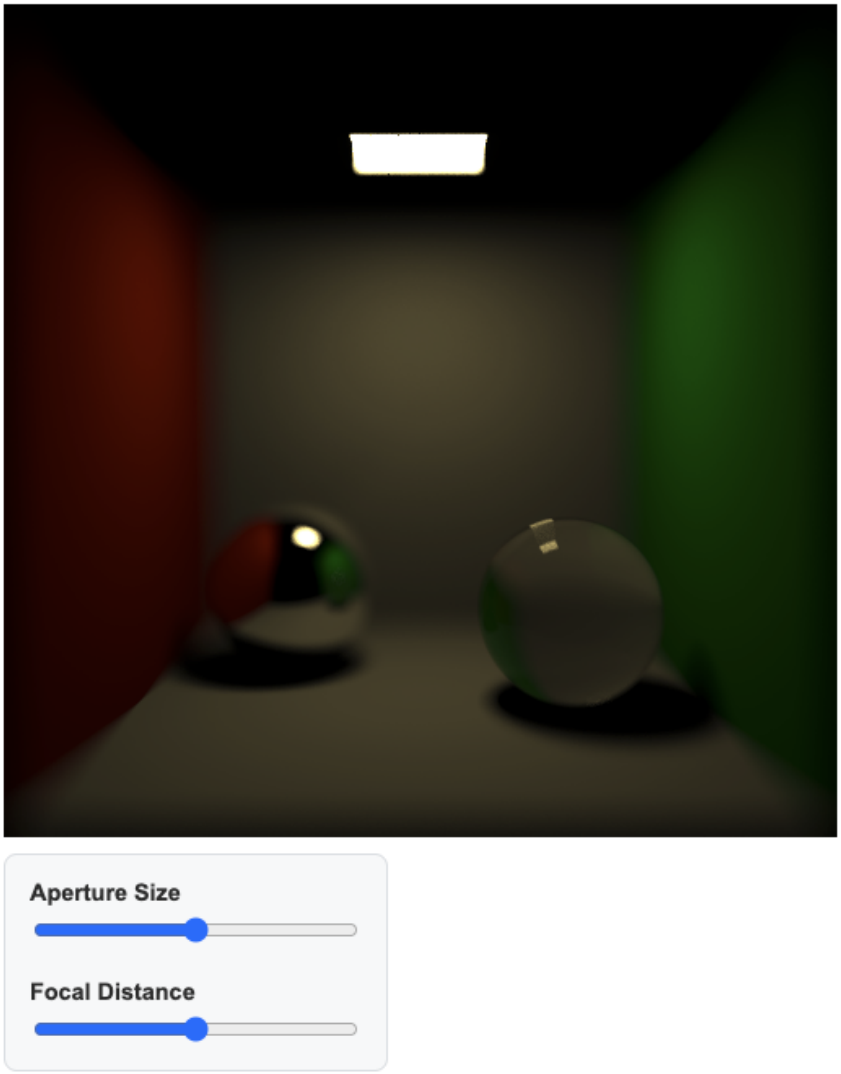


Figure 1: Interface: The user can control the Aperture Size and Focal Distance via sliders. The viewport updates in real-time, showing the shift in focus.

### 3.2 Material System (Dielectric BSDF)

To demonstrate the robustness of the renderer, transparent glass materials were implemented. This required:

- **Fresnel Reflectance:** Using the Fresnel equations to probabilistically determine if a photon reflects off the glass surface or refracts into it.
- **Snell's Law:** Calculating the new direction of the ray as it passes between media of different refractive indices ($\eta \approx 1.5$ for glass).
- **Bouguer's Law (Absorption):** Attenuating the light intensity as it travels through the medium based on distance $d$ and extinction coefficient $\sigma_t$:

$$T_r = e^{-\sigma_t \cdot d} \tag{1}$$

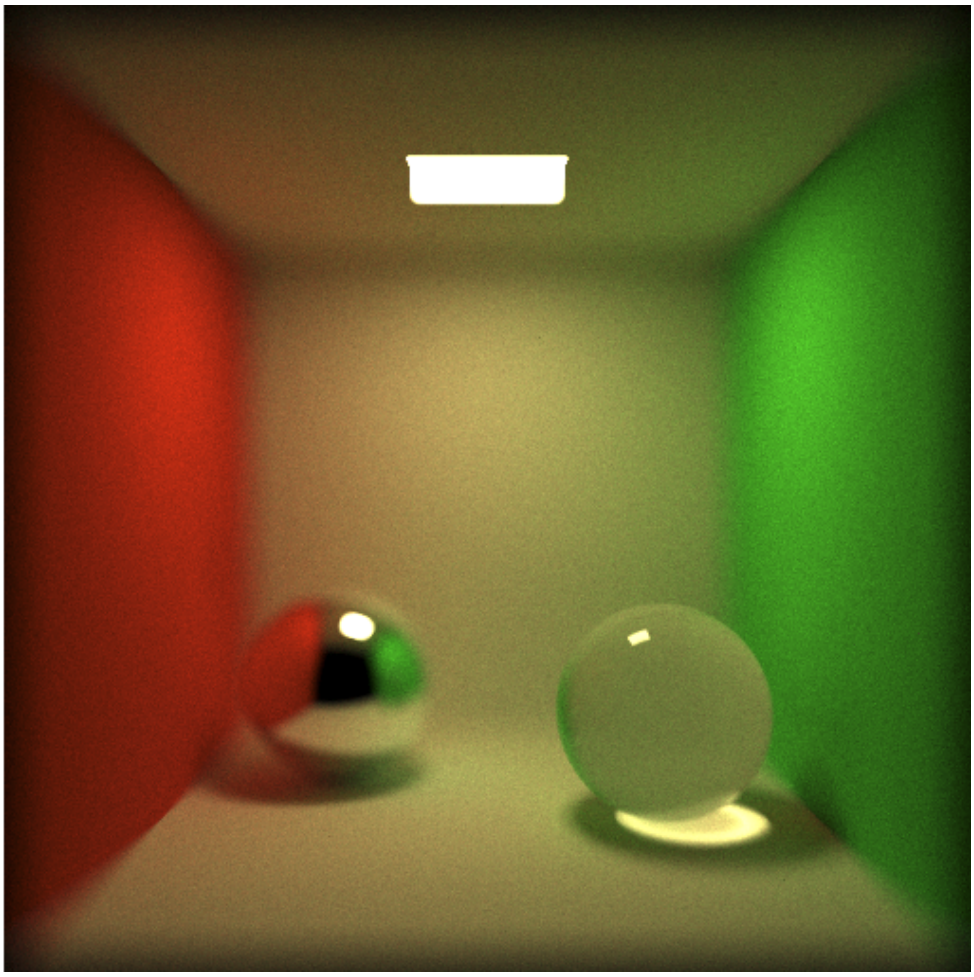

Figure 2: Final Render: The scene demonstrates complex global illumination. Note the soft focus on the back wall (DoF), the caustic highlights on the floor (from the glass sphere), and the color bleeding from the red/green walls.

## 4 Results & Discussion

The final application successfully renders the Cornell Box scene with plausible depth of field at interactive framerates.

**Visual Analysis (Fig. 2):**

- **Depth of Field:** The spheres are kept relatively sharp, while the back wall and corners are visibly softened. This confirms the correct functioning of the aperture sampling.
- **Caustics:** The glass sphere (right) casts a concentrated spot of light on the floor. This is a result of the path tracer successfully finding paths from the light source, through the glass, to the floor, and back to the eye.
- **Color Bleeding:** The left side of the white box and the floor show a reddish tint, while the right side shows green. This confirms that the indirect illumination (Global Illumination) step is correctly gathering scattered light from the colored walls.

**Performance:** The renderer utilizes a **BSP Tree** acceleration structure. This allows the ray-scene intersection to scale logically ($O(\log N)$) rather than linearly, making it possible to render the scene in a browser environment without stalling the GPU.

**Limitations:** The current sampling method (uniform random disk) is efficient but can produce clumping noise at low sample counts. A low-discrepancy sequence (like Halton or Sobol) would converge faster. Additionally, "fireflies" (bright noise pixels) are occasionally visible due to the high variance of sampling small, bright light sources through specular surfaces.

# Bibliography

## References

[1] Pharr, M., Jakob, W., and Humphreys, G. *Physically Based Rendering: From Theory to Implementation*. 4th ed., MIT Press, 2023. (Ch. 6.4: Projective Camera Models - The Thin Lens Model)

[2] Shirley, P. *Ray Tracing in One Weekend*. Series 1, v3.2.3, 2020. (Defocus Blur implementation details)

[3] Frisvad, J. R. *02562 Rendering - Course Notes*. Technical University of Denmark (DTU), 2024. (Week 7: Path Tracing, Week 8: Fresnel & Absorption)

[4] Khronos Group / W3C. WebGPU Specification. https://www.w3.org/TR/webgpu/

## Use of Generative AI

I utilized Gemini (Google) to help structure this report and refine the technical explanations of the Thin Lens Model and Bouguer's Law. I also used it to debug specific WGSL syntax errors related to the random number generator implementation. All code logic, parameter tuning, and final text were verified and finalized by me.

## Contribution

This project was designed and implemented individually.