rptcam: An Implementation of a Realistic Camera Model

Rahul Kumar, Rohan Kumar, Lucy Meng, Michael Yu
UC Berkeley

ABSTRACT

In this paper, we present `rptcam`, an open-source library that integrates configurable camera models into `rpt`, a CPU-based raytracing implementation [5]. `rpt` is of similar complexity and functionality as the pathtracer we wrote in HW3 but additionally includes a configurable aperture diameter for their pinhole camera implementation. In fact, `rpt` was written for an assigment in one of MIT's computer graphics courses. `rptcam` is capable of simulating custom aperture shapes and a variety of lens systems, allowing users to render images as if they were taken by a real camera. The rendered images display artifacts that are characteristic of real cameras, such as bokeh, depth-of-field, spherical aberration, and chromatic aberration.

CCS CONCEPTS

• **Computing methodologies** → **Ray tracing**.

KEYWORDS

Ray tracing, Computer Graphics, Camera Model, Aperture, Lenses, Chromatic Aberration

ACM Reference Format:
Rahul Kumar, Rohan Kumar, Lucy Meng, Michael Yu. 2024. rptcam: An Implementation of a Realistic Camera Model. In *Proceedings of Computer Graphics 2024 (CS 184 '24).* ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

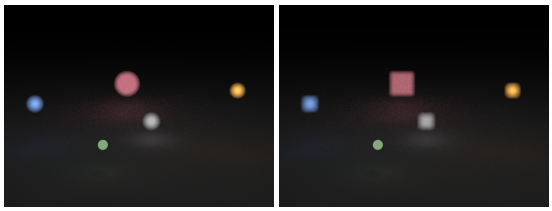
Our contribution is `rptcam`, an open-source library that provides mechanisms for simulating custom aperture configurations, arbitrary spherical lens configurations, and chromatic aberration. This paper will focus on the technical details of these simulation features as well as tools we developed for determining appropriate parameters for different lens systems. An interactive demonstration of our results can be found here.

2 CUSTOM APERTURE SHAPES

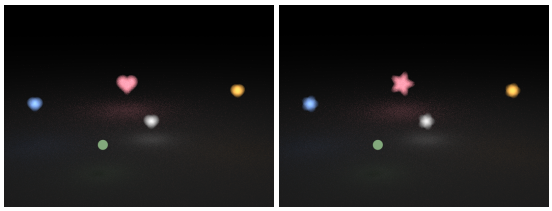
We allow users to configure the aperture shape and diameter for the camera in the scene. Beyond the circular aperture that was already implemented in `rpt`, we support apertures in the shape of an arbitrary polygon.

To implement custom aperture shapes, we adapted the ray sampler. Rather than picking a ray going straight through a particular camera pixel and pinhole, we uniformly sample a point within the camera's aperture shape. For arbitrary polygons, we keep uniformly sampling within the polygon's bounding box until we obtain a point that is contained within the polygon. In the case of a pinhole camera, this point is used to offset the origin of the ray being cast.

Figure 1 shows results that demonstrate our implementation. We set up a scene (based on example scenes provided by the 'rpt' authors) with several spheres having emissive materials. We rendered the scene twice: once with a circular camera aperture, and once with a heart-shaped camera aperture. When imaging an out-of-focus point light source with a camera with a non-zero aperture size, the point source will appear on the sensor plane as the aperture shape. The resulting images are shown in figure 1.



(a) Circular aperture **(b) Square-shaped aperture**



(c) Heart-shaped aperture **(d) Star-shaped aperture**

Figure 1: A scene demonstrating bokeh due to different aperture shapes.

Note that the camera's focal point is set to the green sphere, so it appears circular even when imaged with a heart-shaped aperture.

3 LENS SYSTEM SIMULATION

The existing pathtracing framework came with a simple pinhole camera implementation that had a configurable aperture size. While this enabled simulating bokeh and depth-of-field, the framework did not have the infrastructure to simulate artifacts of lenses such as spherical and chromatic aberration. A major component of our project was simulating custom lens configurations, allowing us to create renders that more closely parallel images taken by real cameras.

Custom lens configurations were implemented as in [2]. Table 1 gives an example lens configuration for a simple biconvex lens. This configuration is then used to determine how rays cast from the camera refract through the lens system before being cast into the scene.

CS 184 '24, April 16, 2024, Berkeley, CA

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

Radius	Thickness	n	d	Aperture Shape
4	0.01	1.6	0.02	Circle
-4	3.995		0.02	Circle

Table 1: A sample configuration of a simple biconvex lens. Surfaces are listed from object-facing to image-facing. The first column gives the radius of curvature and is positive for lenses that curve away from object-space. The thickness gives the distance to the next surface along the camera's axis. n is the index of refraction up to the next surface and d is the aperture diameter. In this case, the specified $n = 1.6$ lens is centered 4 from the image sensor and has two convex surfaces with radius of curvature 4.

Our algorithm for casting rays through a custom lens system is detailed below. We define **d** as the normalized direction of the camera, **r** as the normalized right vector of the camera, and **u** as the normalized up vector of the camera. **d**, **r**, and **u** are mutually orthogonal. We additionally define **o** as the position of the camera in the world frame.

(1) Based on the pixel being sampled, compute the corresponding location on the camera sensor. If the sensor width and height are w and h, respectively, and the x and y coordinates of the pixel normalized to the standard [-1, 1] box are x and y, respectively, the pixel location in world space is

$$\mathbf{p} = \mathbf{o} + \frac{wx}{2}\mathbf{r} + \frac{hy}{2}\mathbf{u}.$$

(2) Randomly choose a point within the aperture of the lens surface closest to the sensor, and cast a ray starting from the computed sensor location to this point on the lens surface. Given x and y coordinates sampled from the 2D aperture in the standard [-1, 1] box and the aperture scaling s, we can first compute the z coordinate of this (x, y) location when projected on the spherical lens surface in relation to the center of the sphere:

$$z = \sqrt{R^2 - (sx)^2 - (sy)^2}.$$

Given the distance of the lens surface to the sensor t, the location of this projected point in world space can then be computed as

$$\mathbf{p'} = \mathbf{o} + (t - R + z \cdot \text{sign}(R))\mathbf{d} + sx\mathbf{r} + sy\mathbf{u}.$$

We now take the ray going from **p** to **p′** and cast it into the remaining lens surfaces.

(3) For each lens surface, starting from the one closest to the sensor, compute where the ray intersects the lens surface. Let **c** be the center of the spherical lens surface (which can be computed using some simple vector arithmetic), **p** be the origin of the current ray, and **v** be the direction of the current ray. The sphere intersection formula is given by

$$(\mathbf{p} + t\mathbf{v} - \mathbf{c})^2 - R^2 = 0.$$

We then solve for t and take the solution that corresponds to the appropriate surface of the lens (closer to the camera for surfaces that curve away from the camera, vice versa for surfaces that curve toward it). If the ray intersects the lens surface outside of its aperture, restart from step 2. To check this, we first need to project the intersection point **i** onto the standard [-1, 1] box:

$$\mathbf{i}_c = \mathbf{i} - \mathbf{o}$$
$$x = \frac{1}{s}\text{proj}_{\mathbf{r}}\mathbf{i}_c$$
$$y = \frac{1}{s}\text{proj}_{\mathbf{u}}\mathbf{i}_c$$

We then check if this (x, y) is contained within the aperture specified in the lens configuration. If the ray lands inside the aperture, use Snell's law to compute the new direction of the ray. The normal vector of the lens is given by

$$\mathbf{n} = \mathbf{i} - \mathbf{c}.$$

We can then compute the required trigonometric functions:

$$\sin\theta_1 = ||\mathbf{n} \times \mathbf{v}||$$
$$\sin\theta_2 = \frac{n_1}{n_2}\sin\theta_1$$

We then compute the new ray direction by decomposing the ray direction into normal and tangential components, then scaling the tangential component to achieve the desired θ_2:

$$\mathbf{v}_\perp = \text{proj}_{\mathbf{n}}\mathbf{v}$$
$$\mathbf{v}_\parallel = \mathbf{v} - \mathbf{v}_\perp$$
$$\mathbf{v}'_\parallel = \frac{\sin\theta_2}{\sin\theta_1}\mathbf{v}_\parallel$$
$$\mathbf{v}' = \frac{\mathbf{v}_\perp + \mathbf{v}'_\parallel}{||\mathbf{v}_\perp + \mathbf{v}'_\parallel||}$$

Update the ray to originate from the intersection point with the surface and travel in the computed direction.

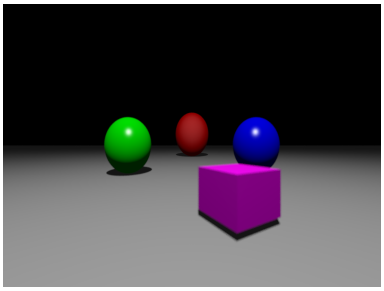
(4) Cast the ray originating from the final lens surface into the scene and continue path tracing as normal.

Using a simple biconvex lens, we are able to observe realistic artifacts such as depth-of-field changing with aperture size, spherical aberration, and an inverted image. A couple images rendered using this lens configuration are shown below. Figure 2 demonstrates the effect of aperture size on the depth of field of the image. Figure 3 demonstrates the effect of the index of refraction of the lens on the focus point of the camera.

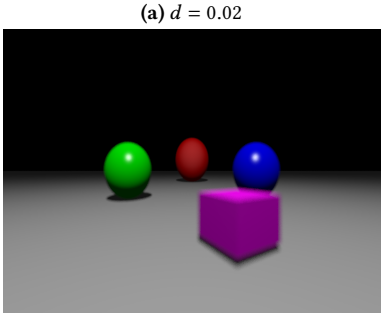
As expected, increasing the aperture size decreases the depth of field as the circle of confusion grows more quickly. Similarly, increasing the index of refraction brings the focus point closer to the camera as the lens refracts light more sharply. It can also be seen that off-axis geometry tends to be more blurry and stretched even when the object is in focus, which demonstrates spherical aberration.

4 CHROMATIC ABERRATION SIMULATION

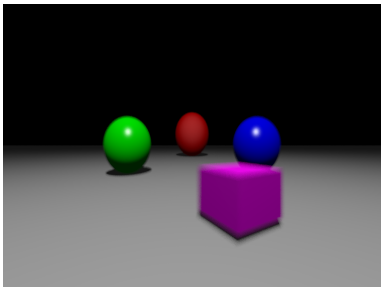
Chromatic aberration is an optical nonideality arising from the fact that many materials (glass in particular) have a wavelength-dependent index of refraction. In many types of glass, the index of refraction decreases with increasing wavelength. In an imaging system, this results in different colors of light coming to focus at different positions. Chromatic aberration commonly manifests as



(a) $d = 0.02$

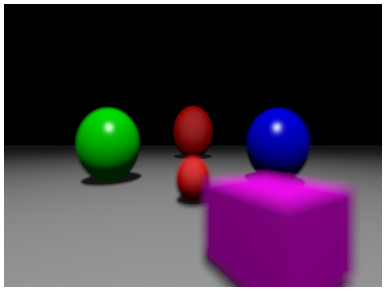


(b) $d = 0.05$

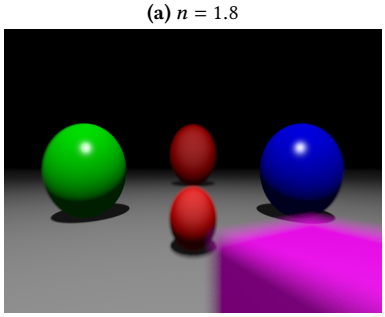


(c) $d = 0.07$

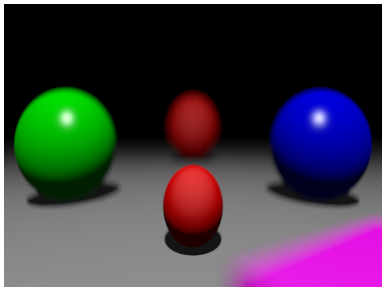
Figure 2: Images rendered using a biconvex lens configuration with increasing aperture diameter d.



(a) $n = 1.8$



(b) $n = 2.3$



(c) $n = 3.5$

Figure 3: Images rendered using a biconvex lens configuration with increasing index of refraction n.

rainbow "smearing", especially near high-contrast, off-axis locations in the image.

Many rendering tools use heuristic approaches to model chromatic aberration. For example, one common approach is to apply scaling and/or offset to the R, G, and B channels of a rendered image. For this project, however, we wanted to implement a physically-based approach to simulating chromatic aberration.

Our model of optical dispersion accepts a V number (also known as Abbe's number) for each optical element in the lens system. The V number is defined as

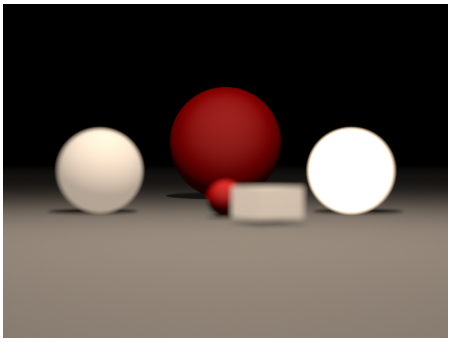
$$v = \frac{n_D - 1}{n_F - n_C}, \tag{1}$$

where n_D, n_F, and n_C are the indices of refraction at the yellow sodium D line (589.3 nm), the blue hydrogen F line (486.1 nm), and the red hydrogen C line (656.3 nm), respectively [3]. We assume that the index of refraction decreases linearly in the visible spectrum, so that

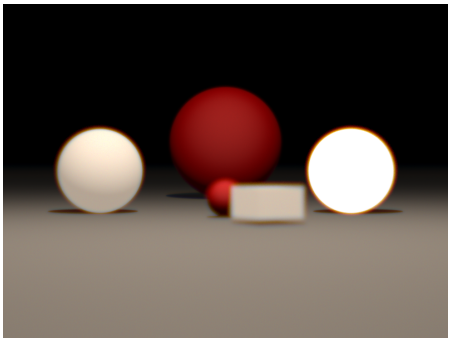
$$n(\lambda) = n_D - \frac{(n_D - 1)(\lambda - \lambda_D)}{v(\lambda_C - \lambda_F)}, \tag{2}$$

where λ_D, λ_F, and λ_C are the wavelengths corresponding to n_D, n_F, and n_C, respectively.

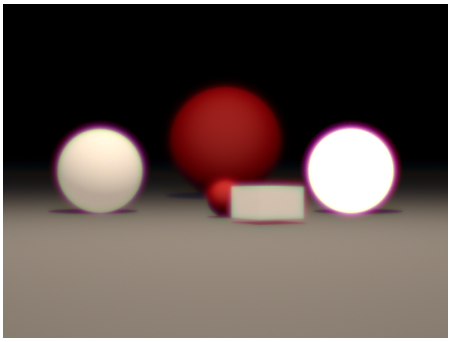
To account for dispersion during rendering, we add an additional Monte Carlo dimension to the lens system simulator described earlier. We assign a wavelength, chosen uniformly randomly between 400nm and 700nm, to each sampled ray. We use equation 2 to calculate the index of refraction of each element in the camera system, and use these indices to calculate the exit position and direction of the sampled ray. We then perform regular ray tracing to estimate the color of the target pixel, and multiply the estimated color with the color assigned to the ray. We note that the multiplication of the sampled color with the ray color is not quite physically accurate; a more accurate calculation would consider the spectral response of the R, G, and B pixels of the camera sensor. Images produced by our simulator are shown in figure 4.



(a) $v = 200$



(b) $v = 10$



(c) $v = 3$

Figure 4: A scene demonstrating chromatic aberration in a single lens camera with varying v number. The images are intentionally out of focus to make the chromatic aberration more visually apparent.

5 ANALYSIS OF LENS SYSTEMS

This section highlights our analysis of various lens systems and how this analysis was used to implement distance-based autofocus and chromatic aberration correction.

5.1 Single Lens

One lens system we implemented is a single lens with configurable radius of curvature, refractive index, and V number. To make it easier to focus the camera at an object a known distance away using this lens, we derived an analytical expression to compute the placement of the lens relative to the sensor given a certain sensor-to-object distance.

The focal length of a single lens is given by the lens maker's equation:

$$\frac{1}{f} = (n-1)\left(\frac{1}{R_1} + \frac{1}{R_2} - \frac{(n-1)t}{nR_1R_2}\right) \tag{3}$$

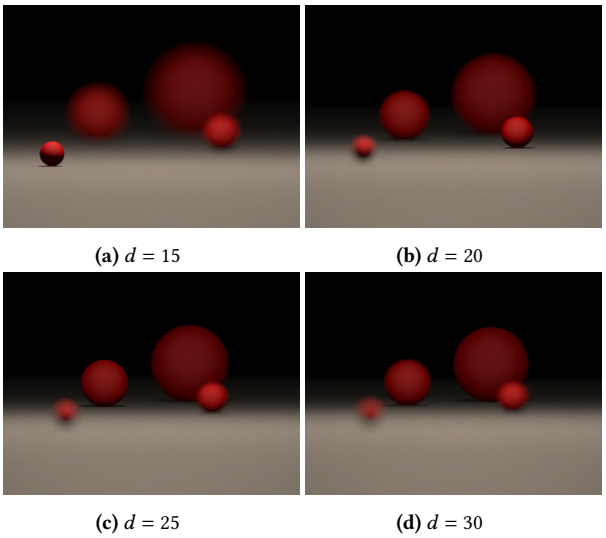
Using this focal length, we can solve for the image distance d_i given the distance between the object and sensor d using the following equation:

$$\frac{1}{f} = \frac{1}{d_i} + \frac{1}{d - d_i} \tag{4}$$

Solving for d_i gives the following equation:

$$d_i = \frac{d \pm \sqrt{d^2 - 4fd}}{2}. \tag{5}$$

We take the smaller of the two solutions that is valid (i.e. the lens does not intersect or go behind the sensor) to allow for larger field of view.



(a) $d = 15$ **(b)** $d = 20$

(c) $d = 25$ **(d)** $d = 30$

Figure 5: A scene demonstrating focusing on objects at various distances from the camera. The camera is placed at $z = 15$, while the spheres are placed at $z = 0$, $z = -5$, and $z = -10$, and $z = -15$. By varying the focus object distance d, we can focus on each sphere sequentially.

5.2 Achromatic Doublet

An ideal achromatic lens would correctly focus all wavelengths of light. An achromatic doublet is a 2 lens system designed such that two specific wavelengths of light (usually red and blue) are focused correctly. We model an achromatic doublet containing two lenses: a biconvex lens with lower dispersion and higher index of refraction, and a biconcave lens with higher dispersion and lower index of refraction as shown in 6.

We implemented a procedure for determining the geometric parameters (in particular, the radii of curvature of the lenses) of an achromatic doublet given the optical parameters of two materials

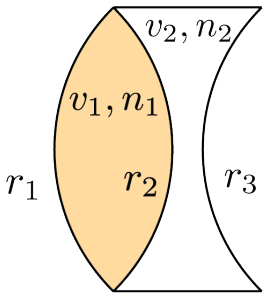


Figure 6: Setup for analyzing the achromatic doublet lens.

and the target focal length of the system. We use the thin lens approximation for simplicity. Our analysis is based on [1], [3], and [4]. We assume that we have two materials with indices of refraction n_1 and n_2 (measured at the yellow D line) and v numbers v_1 and v_2, with $n_1 < n_2$ and $v_1 > v_2$. We assume a lens system composed of a biconvex lens followed immediately by a biconcave lens. This lens system has three surfaces: the boundary between air and the first lens, between the first lens and the second, and between the second lens and air. Denote the radii of curvature of each surface as r_1, r_2, and r_3, respectively, as shown in figure 6. Using the thin lens approximation, the optical power (inverse of focal length) of each lens is given by

$$P_1(\lambda) = \frac{1}{f_1(\lambda)} = (n_1(\lambda) - 1)\left(\frac{1}{r_1} + \frac{1}{r_2}\right) \tag{6}$$

$$P_2(\lambda) = \frac{1}{f_2(\lambda)} = -(n_2(\lambda) - 1)\left(\frac{1}{r_2} + \frac{1}{r_3}\right) \tag{7}$$

The total power of the system (evaluated at the D line) is

$$P(D) = \frac{1}{f_{eq}(D)} = \frac{1}{f_1(D)} + \frac{1}{f_2(D)}. \tag{8}$$

Equating the total power at the C line wavelength and the F line wavelength and simplifying gives

$$P_1(C) + P_2(C) = P_1(F) + P_2(F) \tag{9}$$

$$(n_1(C) - n_1(F))\left(\frac{1}{r_1} + \frac{1}{r_2}\right) = (n_2(C) - n_2(F))\left(\frac{1}{r_2} + \frac{1}{r_3}\right) \tag{10}$$

$$\frac{n_1(D) - 1}{v_1}\left(\frac{1}{r_1} + \frac{1}{r_2}\right) = \frac{n_2(D) - 1}{v_2}\left(\frac{1}{r_2} + \frac{1}{r_3}\right) \tag{11}$$

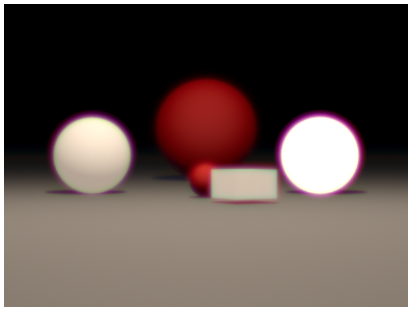
$$\frac{1}{f_1(D)v_1} = \frac{1}{f_2(D)v_2} \tag{12}$$

Our implementation takes in $f_{eq}(D)$ and solves the system of equations given by equations 8 and 12, yielding the geometric parameters for an achromatic doublet with a desired overall focal length. We use the overall focal length to determine where to position the lens, using the approach described in the previous section.

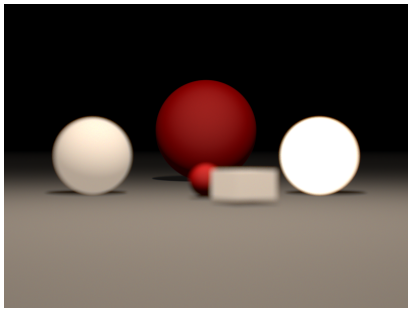
A visual comparison of the chromatic aberration improvement when using an achromatic doublet is shown in figure 7.

6 CONTRAST DETECTION AUTOFOCUS

In the real world, many cameras are able to automatically focus an image using contrast detection autofocus, a strategy that is based upon the idea that sharper images have higher contrast.



(a) Image taken by a single lens camera with $v_1 = 3$.



(b) Image taken with an achromatic doublet with $v_1 = 3$ and $v_2 = 1$.

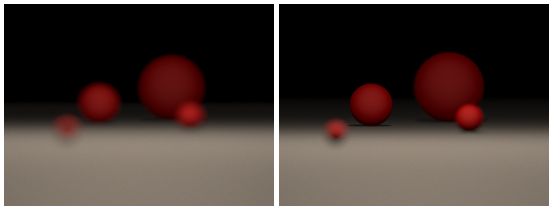
Figure 7: A scene demonstrating the performance of a simulated achromatic doublet. Note that the each lens in the doublet has more or equal dispersion compared to the single lens camera. Observe the significant reduction in aberration when using the achromatic lens.

We decided to simulate this in our project. The idea behind our algorithm is to start with an initial image distance, and iteratively adjust that image distance towards images of higher contrast. We determine the contrast of each image by computing the variance of each pixel's brightness. Contrast is the measure of the difference between the highest and lowest brightness levels in the scene, and variance proved to be a reliable measurement for the brightness's spread. Our algorithm is computed as follows, and the results can be seen in Figure 8:

(1) Given a scene and a camera, begin with an initial image distance $d = 50$ and step size of $s = 5$.
(2) Render an image with the parameters above and measure the variance of the pixel brightness, which represent contrast. To determine brightness, we by convert sRGB into relative luminance using the following formula.

$$Y = (0.2126 * R + 0.7152 * G + 0.0722 * B)$$

(3) Add the step size to the initial distance and recompute the brightness variance of the new image.
   – If the new variance is less than the initial, divide the step size in half and invert it.
   – If the new variance is greater than the initial, continue stepping the current direction.
(4) Repeat steps 2-3 until the brightness variance converges.



(a) Unfocused image **(b) Auto-focused image**

Figure 8: A scene demonstrating autofocus. Our autofocus algorithm starts at distance d=50, where nothing is in focus. It converges on a distance of d=25, which focuses on the third sphere.

This autofocus procedure to be used with any lens system implemented in `rptcam`, provided it implements a simple interface that allows us to vary its focal length.

7 CONCLUSION

Our project adds many camera modelling features compatible with basic raytracing algorithms. We completed most of the goals outlined on our proposal, with the exception of comparing our renders to those taken on real cameras. Real cameras have very complex and carefully engineered lens systems, and we were unable to design/replicate them in our simulator.

ACKNOWLEDGMENTS

We'd like to thank the CS 184 course staff for providing the idea for this project and feedback on the project's goals. We also thank the authors of `rpt` for creating the base path tracer implementation upon which we built our project.

REFERENCES

[1] Victor Argueta. 2024. Achromatic Doublet Design and Optimization. https://www.opticsforhire.com/blog/achromatic-doublet-with-flint-crown-glass/

[2] Craig Kolb, Don Mitchell, and Pat Hanrahan. 1995. A realistic camera model for computer graphics. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. Association for Computing Machinery, New York, NY, USA, 317–324. https://doi.org/10.1145/218380.218463

[3] Carl Rod Nave. [n. d.]. Dispersion. http://hyperphysics.phy-astr.gsu.edu/hbase/geoopt/dispersion.html

[4] Jeremy Tatum. 2022. 2.10: Designing an achromatic doublet. https://phys.libretexts.org/Bookshelves/Optics/Geometric_Optics_(Tatum)/02%3A_Lens_and_Mirror_Calculations/2.10%3A_Designing_an_Achromatic_Doublet

[5] Eric Zhang and Alexander Morozov. 2021. Ekzhang/RPT: A physically-based path tracer. https://github.com/ekzhang/rpt

Received 16 April 2024