Project 2: Realistic Camera Model

B03902124 黃信元

I. Basic Workflow

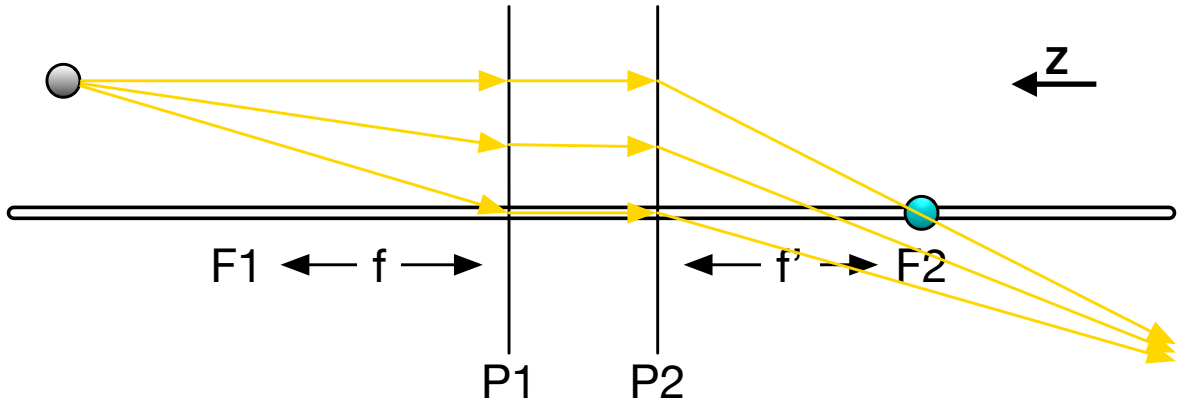
I have created a few variants of realistic camera model based on the paper, nevertheless they all follows a basic workflow consisting of the following steps:

A. Preprocess:
- Some simple reading and calculation from the spec file for the camera.
- Create the Exit Pupil: Using rear-most lens or a more accurate version from the paper.
- Create a transform **Raster2Camera**.

B. For each function call to **GenerateRay**:
- Transform a point in raster (film) space to a point in camera space via **Raster2Camera**.
- Sample a point on the Exit Pupil and create the initial ray **Incident**.
- Trace the ray through the lens system: Using full simulation or by thick lens approximation.
- Calculate the weighting for the initial ray **Incident**.

II. Thick Lens Approximation

Using thick lens is important when calculating exit pupil as stated in the paper and can also give rise to a fast approximation to the original lens system, so I will briefly introduce some basics of thick lens approximation and my implementations.



Basically it is the same as ideal lens, i.e. when the ray hits the center of the lens, the ray will not be refracted; when a parallel light enter, it will be refracted and pass through the focal point. These two rays will intersect at a point, which is the image of the grey point. All other ray will be refracted in such a way that it pass through the image point. The only difference for thick lens is that when hitting the lens at **P1**, the ray will be transported to **P2**, and the same behaviour apply.

I call the left space as real space and the right space as the image space, since the film is often placed to the right. Note that **f** and **f'** happens to be the same in our case, since the medium in real and image space are the same. One important issue is to calculate the transformation between real space and image space, which is named **TL_toImage** and **TL_toReal** in my implementation. From the refraction rule and some geometry, we can have the following equations. Some simple algebra will then give us the desired transformation for both image to real and real to image.

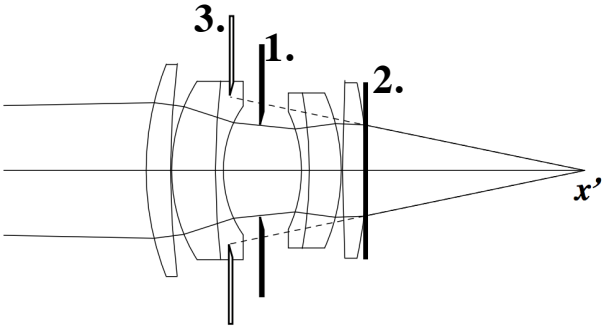
$$\frac{1}{z_r - P_1} + \frac{1}{P_2 - z_i} = \frac{1}{f} \qquad \frac{x_r}{z_r - P_1} = \frac{x_i}{z_i - P_2} \qquad \frac{y_r}{z_r - P_1} = \frac{y_i}{z_i - P_2}$$

The last thing we need is to construct ideal thick lens from a given lens system. This is done by tracing two rays parallel to the axis through the lens system (one from left, one from right). For

the incoming parallel ray going from left to right, there is an outgoing ray after tracing through the system (how to trace a ray through the lens system will be thoroughly discussed later). The intersection of the outgoing ray with the axis is the point F2, while the intersection with the original parallel ray is the point P2. This can be clearly seen from the figure. P1 and F1 can be determined from the ray going from right to left. Note that there are many parallel rays to choose from, any ray that will not be blocked is fine. To not be blocked, I choose one that is close to the center. Since it is only an approximation, using different ray will result in slightly different thick lens. Calculating the thick lens approximation is implemented in **ThickLens(int start, int end, float &P1, float &P2, float &F1, float &F2)**. The function can create thick lens approximation for any subsystem, this is needed when we want to calculate a more accurate exit pupil.

III. Exit Pupil

The only use of exit pupil is in **GenerateRay**. We will sample a point on the exit pupil and create a ray **Incident** from the given point on the film to the sampled point. In general, we should shoot ray from the given point to all directions. But since only some of them will pass through the lens system and enter the real world, we should only sample those that will. The exit pupil thus defines a set of direction for each point on the film. Some choices for exit pupil are as follows:

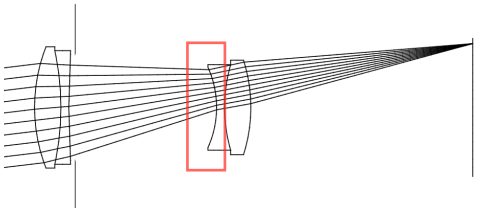


1. The aperture stop: This is actually a bad choice, as illustrated in the figure to the right (copied from the paper). There may be rays that will pass through the lens system but is not in the set of direction defined using the aperture stop.
2. The rearmost lens: The exit pupil is located at the last lens as shown in the figure. The radius of the pupil is the radius of the last lens, and the z position is at the boundary of the rearmost lens. This setting may include many rays that will not pass through the lens system. Also notice that there is a very small slit between exit pupil 2 and the lens boundary. If the rearmost lens has a very small radius or the film is very large compare to the lens, then points on the boundary of the film can shoot rays that will hit the rearmost lens (thus may pass through the system), but will not hit the exit pupil. This is the case for wide-angle camera. Thus exit pupil 2 has similar problem as 1.
3. The image of aperture stop: This is a more accurate exit pupil. We calculate the image of the aperture stop from the subsystem consisting of lens between aperture stop and the film. But the image is not easy to calculate, so we create a thick lens approximation of the subsystem. Then we use **TL_toImage** to find the image of the aperture circle (we only need to map one point due to azimuthal symmetry). The set of direction defined using this exit pupil is roughly the set of direction that will pass through the aperture stop. Since a ray that pass through the aperture stop is likely to pass through the lens system in a good camera design, most rays shooting toward exit pupil 3, unlike exit pupil 2, will not be wasted (blocked).

IV. Tracing Ray Through the Lens System

There are two ways to trace the ray through the lens system. The first one is by full simulation. It consist of a few steps for each refraction by the lens (the intersection with aperture stop is simple, so I will omit the discussion).

1. Intersection with the hemisphere: I solve the quadratic equation using **pbrt** implementation to find **t0** and **t1**. If **t0** is within the ray range, then **t0** is the intersection with the sphere, but we are now intersecting a hemisphere, so we have to check if it is on the correct side. E.g., on telephoto camera (shown to the left), when tracing left to



right (in thick lens approximation), there is a lens that has small radius so it will hit the wrong hemisphere. This problem can be handled by considering the sign of radius and the sign of the intersection point's **z** (in sphere space). If it doesn't hit the hemisphere, I will return 0.
2. Check if the intersection point is within the aperture: If not, I will simply return 0.
3. Refraction: With the intersection point P known, the normal can be calculated by P minus the sphere center. And I use Heckbert's method to calculate the refracted direction of the incident ray, since it uses the least **sqrt** and **division**.

Another way to trace through the lens system is by thick lens approximation. Since we only need to trace a ray starting from the film, we only need to focus on right to left (image space to real space). Consider a ray with origin **O** and direction **D**, we calculate the real point of **O**, say **O'**. Then we intersect the ray at z = P2 plane, and shift the z coordinate to P1, which becomes **O''**. As discussed before, the refracted ray will pass through **O'**. So the outgoing ray is simply **O''** to **O'**. This gives rise to a simple and efficient method to trace through the lens system.

V. Miscellaneous

1. **Raster2Camera** is easy to calculate. One minor problem is that the image of the world on the film is 上下顛倒左右相反 (unlike perspective camera), so we have to flip the coordinate to form a more natural image (I combined this flipping in **Raster2Camera**).
2. I simply use **ConcentricSampleDisk**, implemented in **pbrt**, to sample a point on unit disk.
3. I follow the derivation in the paper to calculate irradiance, which equals the LHS of the below equation, where A_D is the area of the exit pupil. By uniform sampling on the disk, it becomes the RHS of the below equation. So the weighting for each sample is pi * r^2 * cos^4(t) / Z^2, where r is the radius of exit pupil, t is \theta' and Z is the distance between film and exit pupil.

$$\frac{1}{A_D}\int_{x''\in D}\frac{A_D}{Z^2}L(x'',x)\cos^4\theta' dA'' \approx \frac{1}{N}\sum_{i=1}^{N}\frac{A_D}{Z^2}\cos^4\theta' L(x'',x)$$

VI. My Variants for Realistic Camera Model

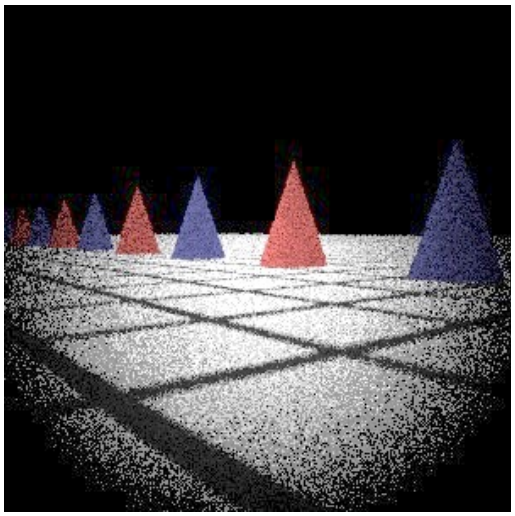
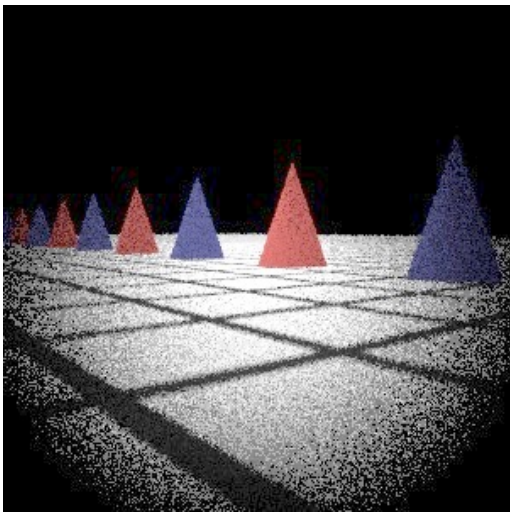
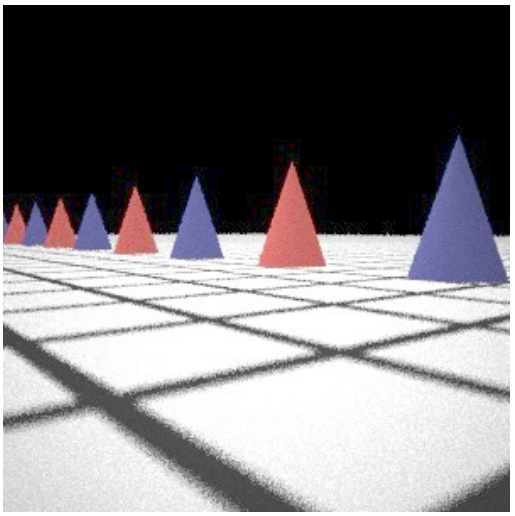
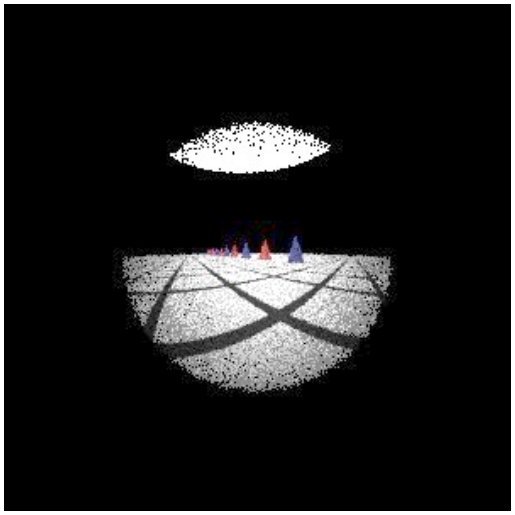
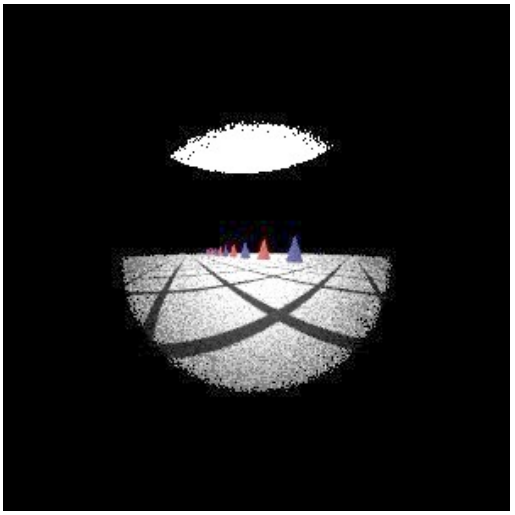
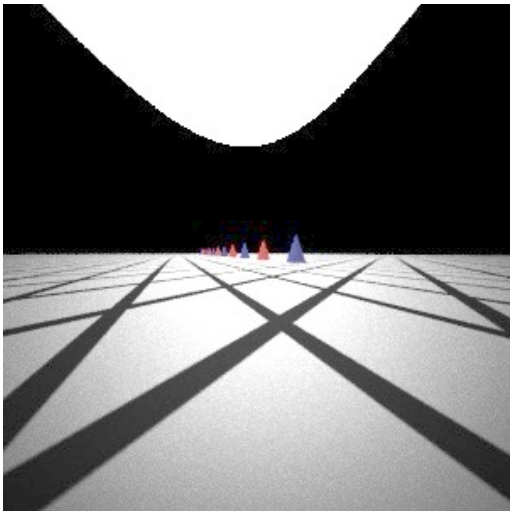
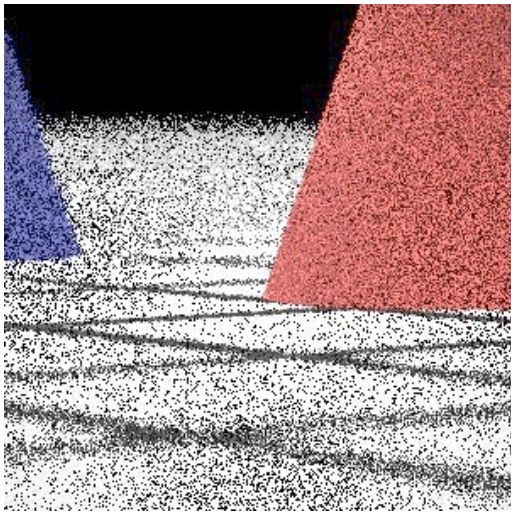
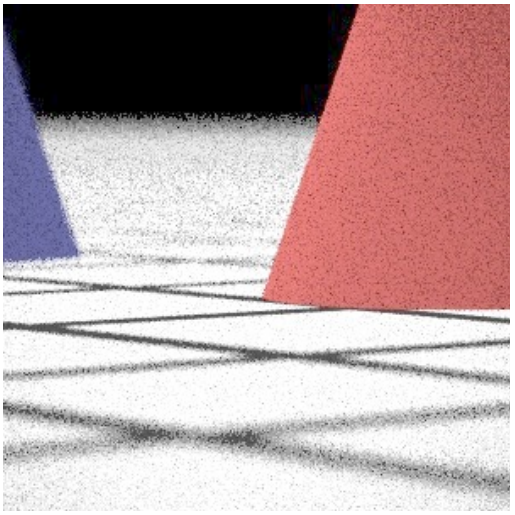
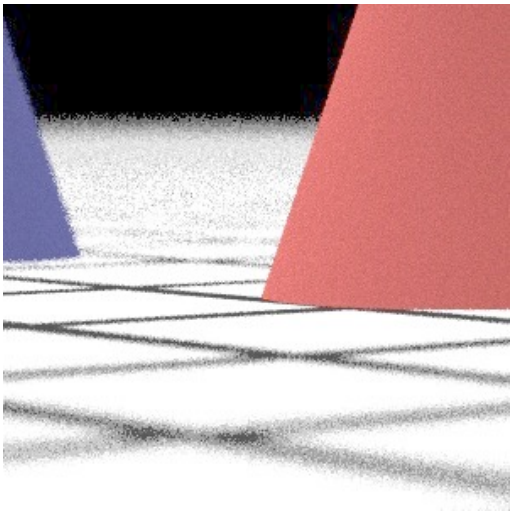
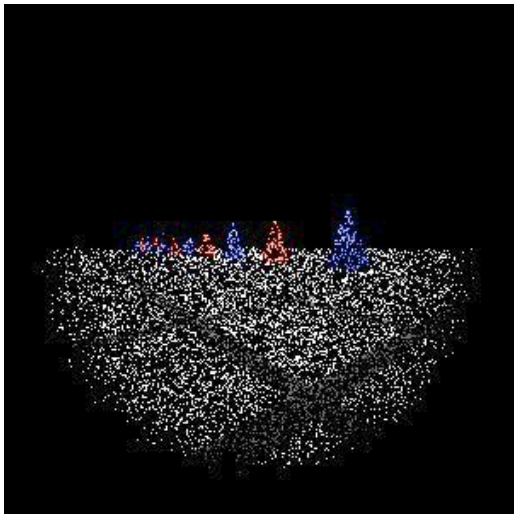
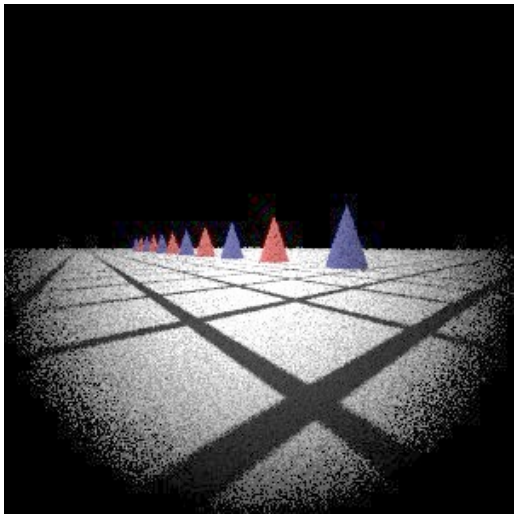
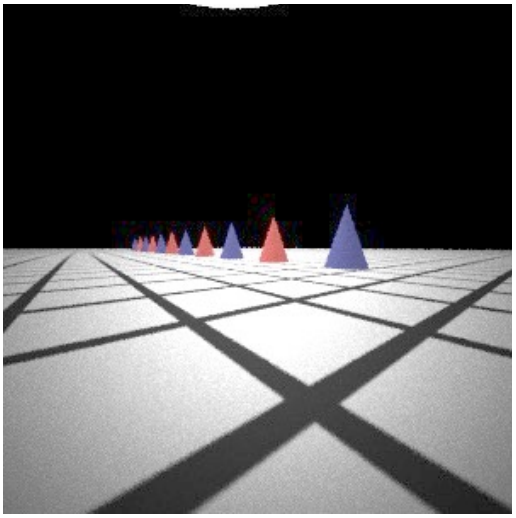
I have implemented three variants based on the above workflow.

1. **realistic.***: It uses the exit pupil 2 and trace the ray using full simulation.
2. **realistic_EP.***: It uses the exit pupil 3 and trace the ray using full simulation.
3. **realistic_TL.***: It uses the exit pupil 3 and trace the ray using thick lens approximation.

The first variant is the fastest, since it threw away many rays if it is blocked when tracing through the lens. Thus it contains many black dots in 4 samples per pixel. There are also more black region on the boundary (e.g. wide angle camera), due to the reason I have previously mentioned. The second variant is the slowest, because most rays can pass through the lens system, and full simulation is much slower than thick lens approximation (but tracing ray in lens system is not the bottleneck, so the time difference is not so big). The third variant is very clear and doesn't contain any black dots, because all rays will pass through the approximated lens system (Even at film boundary). In general, thick lens approximation is similar to full simulation, but for camera that has large distortion such as fisheye camera, thick lens approximation is not so accurate.

VII. My Results for All Variants

There are totally 48 pictures. The next page consists of all pictures using 4 samples per pixel. The page after the next page consists of all pictures using 512 samples per pixel.

Variant 1	Variant 2	Variant 3
		
		
		
		

Variant 1	Variant 2	Variant 3