Realistic Camera Model

Shan-Yung Yang

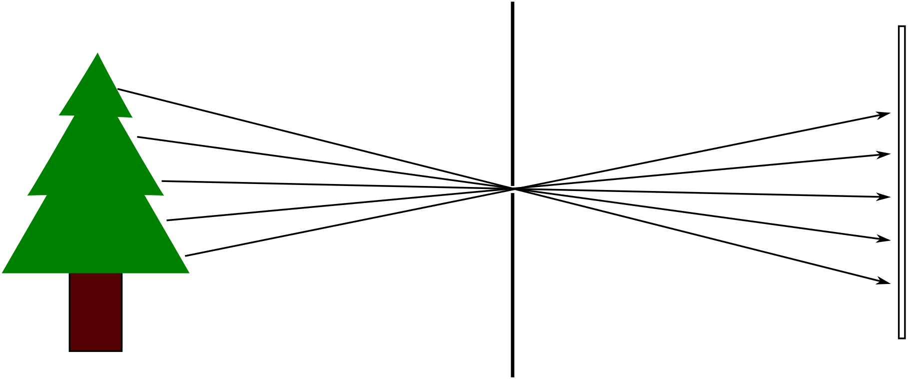
November 2, 2006

Outline

- Introduction
- Lens system
- Thick lens approximation
- Radiometry
- Sampling
- Assignment #2

Introduction

Until now we have only discussed the pinhole camera model, which is not phisically correct.



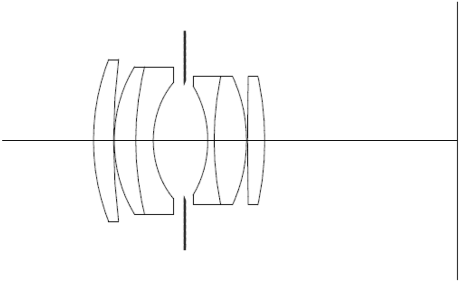
Why We Need Realistic Model

- Phisical correctness is our goal.
- Combining real images with synthetic ones is very common in digital visual effects.
- Machine vision and scientific applications need to simulate camera correctly.
- Users of 3D graphics system are familiar with cameras.

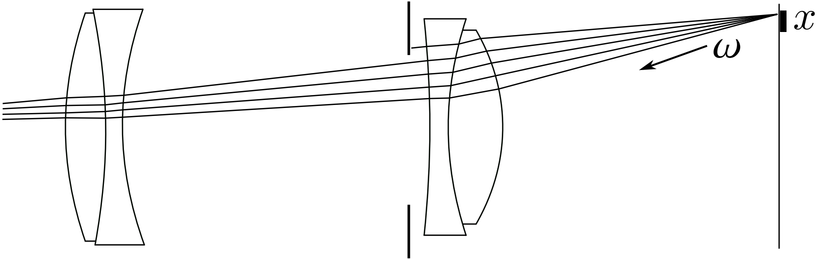
Lens Systems

Lens systems are typically constructed from a series of individual spherical lenses.

radius	thick	n_d	V-no	ap
58.950	7.520	1.670	47.1	50.4
169.660	0.240			50.4
38.550	8.050	1.670	47.1	46.0
81.540	6.550	1.699	30.1	46.0
25.500	11.410			36.0
	9.000			34.2
-28.990	2.360	1.603	38.0	34.0
81.540	12.130	1.658	57.3	40.0
-40.770	0.380			40.0
874.130	6.440	1.717	48.0	40.0
-79.460	72.228			40.0



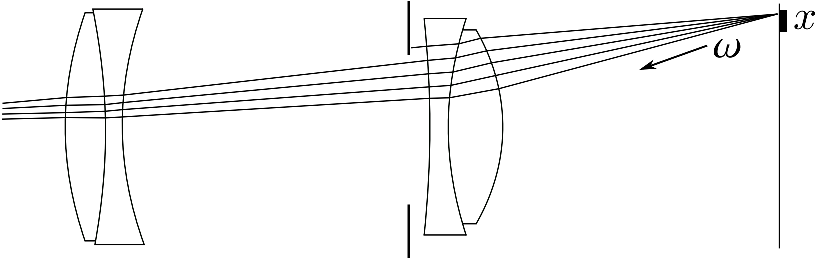
Measurement Equation



$$R = \int\int\int\int L(T(x,\omega,\lambda);\lambda)S(x,t)P(x,\lambda)\cos\theta \; dx \; d\omega \; dt \; d\lambda$$

L: radiance
T: image to object space transformation
S: shutter function
P: sensor response characteristics

Measurement Equation



$$R = \Delta t \cdot \int \int L(T(x,\omega)) \cos\theta \; dx \; d\omega$$

L: radiance T: image to object space transformation

Solving the Integral

Problem: given a function f and domain Ω, how to calculate

$$\int_{\Omega} f(x)dx$$

Solving the Integral

Problem: given a function f and domain Ω, how to calculate

$$\int_\Omega f(x)dx$$

Solution: Monte Carlo method:

$$\int_\Omega f(x)dx \approx [\frac{1}{N}\sum_{i=1}^{N} f(x_i)] \cdot \int_\Omega dx$$

where $x_1, x_2, \ldots, x_N$ are uniform distributed random samples in Ω.

Algorithm

1. For each pixel on the image, generate some random samples x_i and ω_i uniformly.

Algorithm

1. For each pixel on the image, generate some random samples x_i and ω_i uniformly.
2. For each x_i and ω_i, calculate $T(x_i, \omega_i)$.

Algorithm

1. For each pixel on the image, generate some random samples x_i and ω_i uniformly.
2. For each x_i and ω_i, calculate $T(x_i, \omega_i)$.
3. Shoot the ray according to the result of $T(x_i, \omega_i)$ into the scene, and calculate the radiance.

Algorithm

1. For each pixel on the image, generate some random samples x_i and ω_i uniformly.
2. For each x_i and ω_i, calculate $T(x_i, \omega_i)$.
3. Shoot the ray according to the result of $T(x_i, \omega_i)$ into the scene, and calculate the radiance.
4. Set the pixel value to the average of radiance.

Tracing Rays through Lens System

1. $R = Ray(x_i, \omega_i)$

Tracing Rays through Lens System

1. $R = Ray(x_i, \omega_i)$
2. Calculate the intersection point p for each lens element E_i from rear to front.

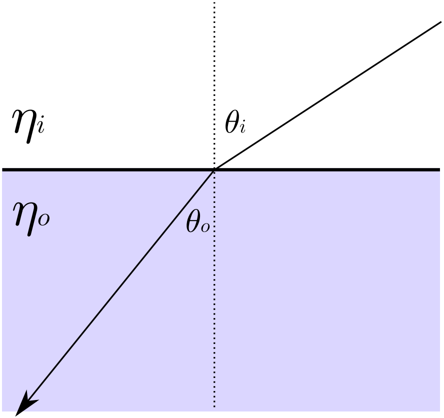
Tracing Rays through Lens System

1. $R = Ray(x_i, \omega_i)$
2. Calculate the intersection point p for each lens element E_i from rear to front.
    1. Return zero if p is outside the aperture of E_i.

Tracing Rays through Lens System

1. $R = Ray(x_i, \omega_i)$
2. Calculate the intersection point p for each lens element E_i from rear to front.
    1. Return zero if p is outside the aperture of E_i.
    2. Compute the new direction by Snell's law if the medium is different.

Snell's Law



$$\eta_i \sin\theta_i = \eta_o \sin\theta_o$$

Ideal Lens Approximation

- In some situations we need an ideal lens approximation.

Ideal Lens Approximation

- In some situations we need an ideal lens approximation.
  - Ideal lens: each point in object space is imaged onto a single point in the image space.

Ideal Lens Approximation

- In some situations we need an ideal lens approximation.
  - Ideal lens: each point in object space is imaged onto a single point in the image space.
  - All points on the **plane of focus** map onto the image plane.

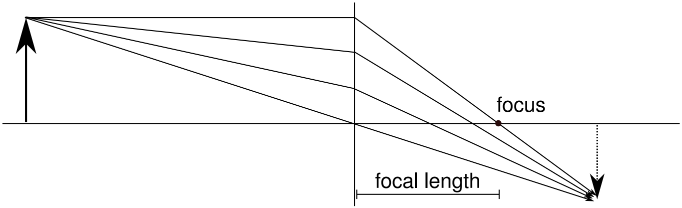
Ideal Lens Approximation

- In some situations we need an ideal lens approximation.
  - Ideal lens: each point in object space is imaged onto a single point in the image space.
  - All points on the **plane of focus** map onto the image plane.
- Thin lens approximation assumes that the thickness of lens is zero.

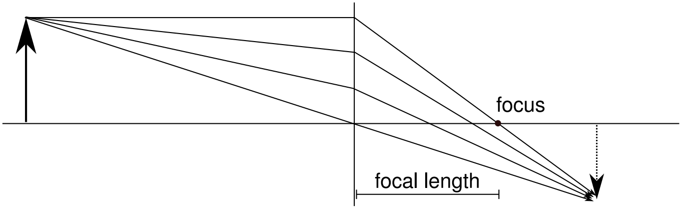
Ideal Lens Approximation

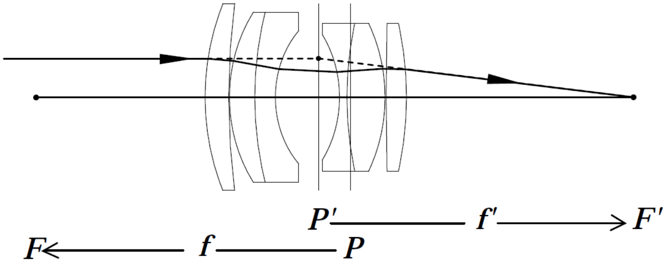
- In some situations we need an ideal lens approximation.
  - Ideal lens: each point in object space is imaged onto a single point in the image space.
  - All points on the **plane of focus** map onto the image plane.
- Thin lens approximation assumes that the thickness of lens is zero.
- Thick lens approximation has additional parameter of thickness.

Thin Lens and Thick Lens

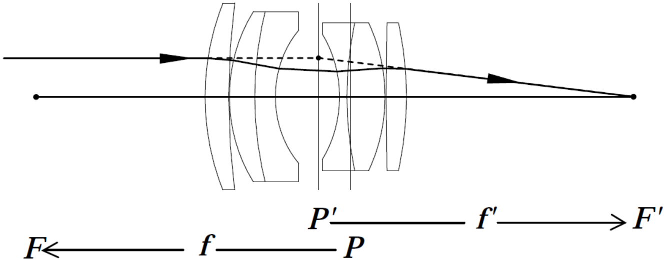


Thin Lens and Thick Lens



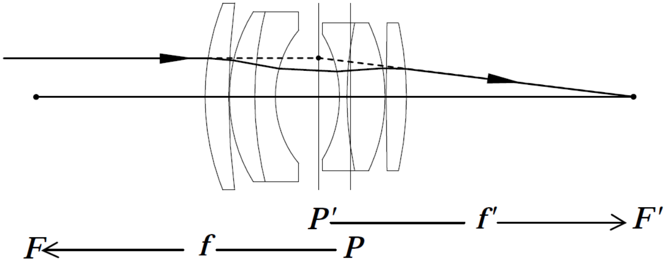


Finding Thick Lens Approximation



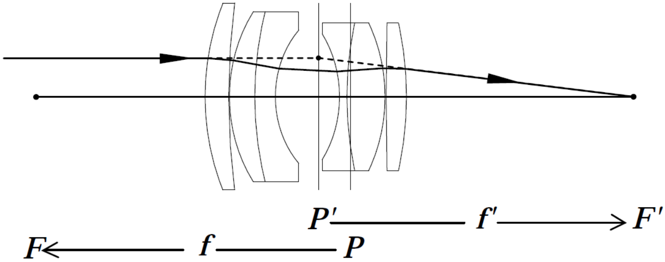
1. Shoot a ray parallel to the axis to find the focus.

Finding Thick Lens Approximation



1. Shoot a ray parallel to the axis to find the focus.
2. Find the principal plane by intersecting the refracted ray and parallel one.

Finding Thick Lens Approximation



1. Shoot a ray parallel to the axis to find the focus.
2. Find the principal plane by intersecting the refracted ray and parallel one.
3. Find the secondary principal plane by tracing from another side.

Application of Thick Lens Approximation

- A faster way to calculate $T(x_i, \omega_i)$.

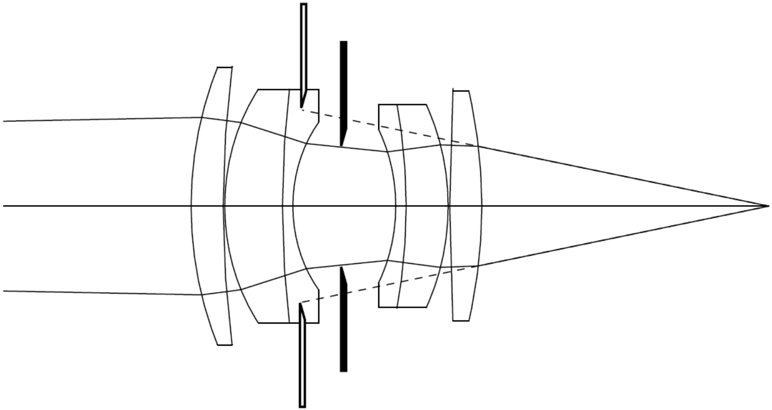
Application of Thick Lens Approximation

- A faster way to calculate $T(x_i, \omega_i)$.
- Autofocus.

Application of Thick Lens Approximation

- A faster way to calculate $T(x_i, \omega_i)$.
- Autofocus.
- Calculate the **exit pupil**.

Exit Pupil (1/2)



The exit pupil is the effective aperture stop in the image space which allows ray incidence.

Exit Pupil (2/2)

- Finding the exit pupil:

Exit Pupil (2/2)

- Finding the exit pupil:
  1. For each aperture stop, calculate its image by thick lens approximation.

Exit Pupil (2/2)

- Finding the exit pupil:
  1. For each aperture stop, calculate its image by thick lens approximation.
  2. Find the aperture stop whose image subtends the smallest solid angle.

Exit Pupil (2/2)

- Finding the exit pupil:
  1. For each aperture stop, calculate its image by thick lens approximation.
  2. Find the aperture stop whose image subtends the smallest solid angle.
- You may also use the aperture of the nearest lens as the exit pupil.

Exposure

- Assume that the irradiance is constant over the exposure period:

$$R = \Delta t \cdot \int \int L(T(x', \omega)) \cos \theta' \; dx' \; d\omega$$

Exposure

- Assume that the irradiance is constant over the exposure period:

$$R = \Delta t \cdot \int \int L(T(x', \omega)) \cos\theta' \; dx' \; d\omega$$

- In practice, we only need to integrate over the **exit pupil** instead of the whole semisphere.

Exposure

- Assume that the irradiance is constant over the exposure period:

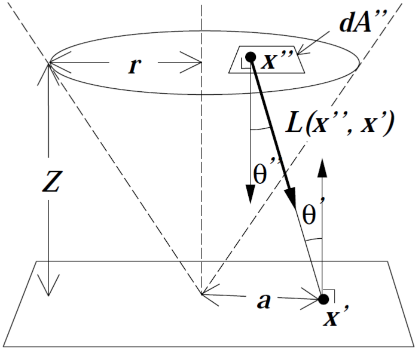
$$R = \Delta t \cdot \int \int L(T(x', \omega)) \cos\theta' \; dx' \; d\omega$$

- In practice, we only need to integrate over the **exit pupil** instead of the whole semisphere.
- Let

$$E(x') = \int L(T(x', \omega)) \cos\theta' \; d\omega$$

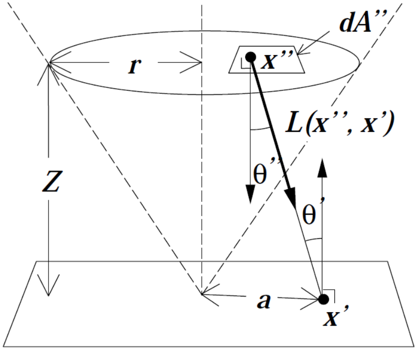
$$R = \Delta t \cdot \int E(x') \; dx'$$

Integral over the Exit Pupil



$$E(x') = \int_{x'' \in D} L(x'', x') \cos\theta' \; d\omega$$

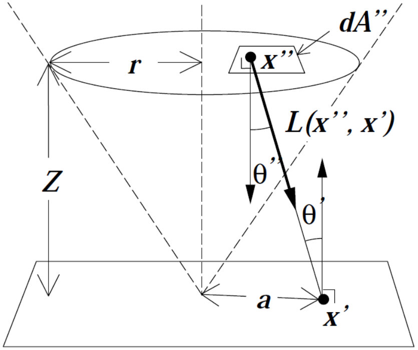
Integral over the Exit Pupil



$$E(x') = \int_{x'' \in D} L(x'', x') \cos\theta' \; d\omega$$

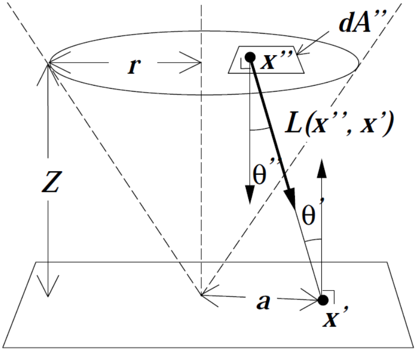
$$d\omega = \frac{\cos\theta''}{r^2} dA''$$

Integral over the Exit Pupil



$$E(x') = \int_{x'' \in D} L(x'', x') \frac{\cos\theta' \cos\theta''}{||x'' - x'||^2} \, dA''$$

Integral over the Exit Pupil



$$E(x') = \frac{1}{Z^2} \int_{x'' \in D} L(x'', x') \cos^4 \theta' \, dA''$$

Sampling a Disk Uniformly

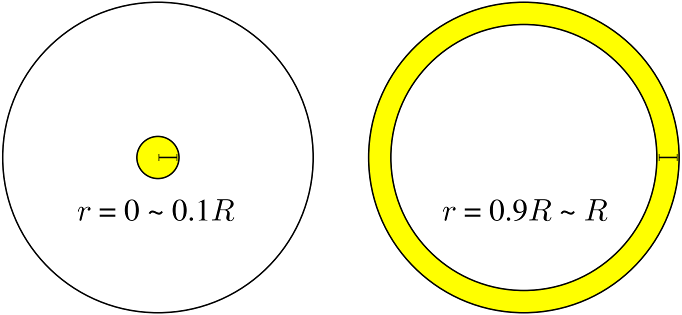
- Now we need to obtain random samples on a disk uniformly.

Sampling a Disk Uniformly

- Now we need to obtain random samples on a disk uniformly.
- How about uniformly sample r in $[0, R]$ and θ in $[0, 2\pi]$ and let $x = r\cos\theta, y = r\sin\theta$?

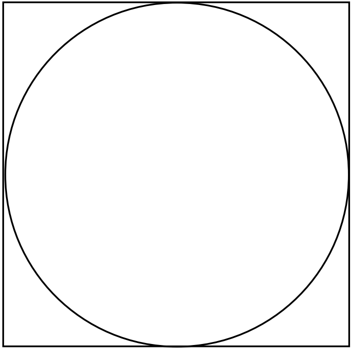
Sampling a Disk Uniformly

- Now we need to obtain random samples on a disk uniformly.
- How about uniformly sample r in $[0, R]$ and θ in $[0, 2\pi]$ and let $x = r\cos\theta, y = r\sin\theta$?
  - The result is not uniform due to coordinate transformation.



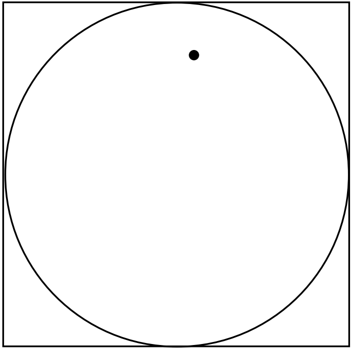
A Simple Method: Rejection

1. Uniformly sample a point in the bounding square of the disk.
2. If the sample lies outside the disk, reject it and sample another one.



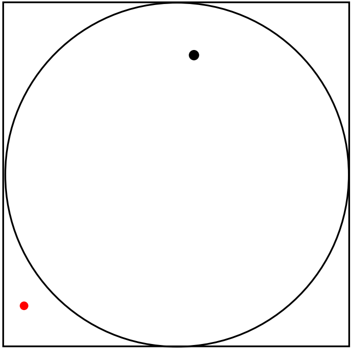
A Simple Method: Rejection

1. Uniformly sample a point in the bounding square of the disk.
2. If the sample lies outside the disk, reject it and sample another one.



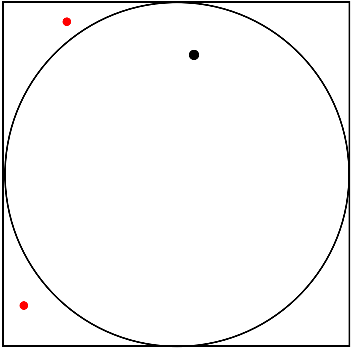
A Simple Method: Rejection

1. Uniformly sample a point in the bounding square of the disk.
2. If the sample lies outside the disk, reject it and sample another one.



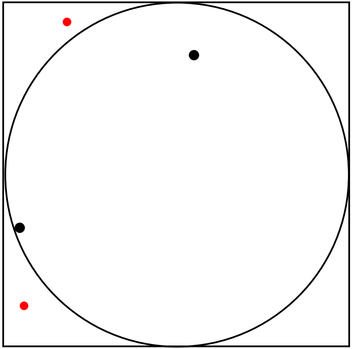
A Simple Method: Rejection

1. Uniformly sample a point in the bounding square of the disk.
2. If the sample lies outside the disk, reject it and sample another one.



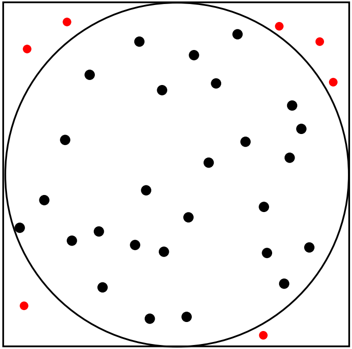
A Simple Method: Rejection

1. Uniformly sample a point in the bounding square of the disk.
2. If the sample lies outside the disk, reject it and sample another one.



A Simple Method: Rejection

1. Uniformly sample a point in the bounding square of the disk.
2. If the sample lies outside the disk, reject it and sample another one.



Another Method

- Sample r and θ in a specific way so that the result is uniform after coordinate transformation.

Another Method

- Sample r and θ in a specific way so that the result is uniform after coordinate transformation.
- Let

$$r = \sqrt{\xi_1},\ \theta = 2\pi\xi_2$$

where ξ_1 and ξ_2 are random samples distributed in $[0, 1]$ uniforml uniformly.

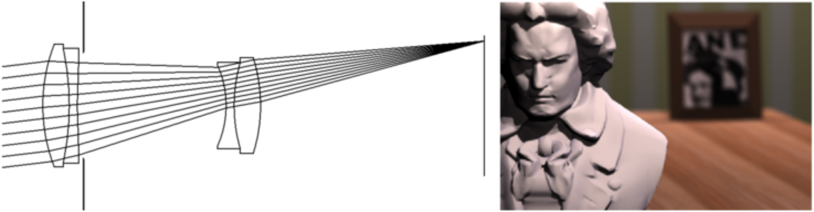
Another Method

- Sample r and θ in a specific way so that the result is uniform after coordinate transformation.
- Let

$$r = \sqrt{\xi_1},\ \theta = 2\pi\xi_2$$

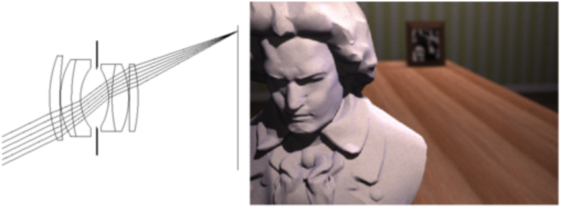
  where ξ_1 and ξ_2 are random samples distributed in $[0, 1]$ uniforml uniformly.
- This produce uniform samples on a disk after coordinate transformation. We will prove it later in chapter 14 “Monte Carlo integration”.

Results



200mm Telescope

Results



50mm General

Results

35mm wide-angle

Results

16mm Fisheye

Assignment #2

- Write the "realistic" camera plugin for PBRT which implements the realistic camera model.

Assignment #2

- Write the “realistic” camera plugin for PBRT which implements the realistic camera model.
- The description of lens system will be provided.

Assignment #2

- Write the “realistic” camera plugin for PBRT which implements the realistic camera model.
- The description of lens system will be provided.
- `GenerateRay(const Sample &sample, Ray *ray)`
  - PBRT generate rays by calling `GenerateRay()`, which is a virtual function of `Camera`.

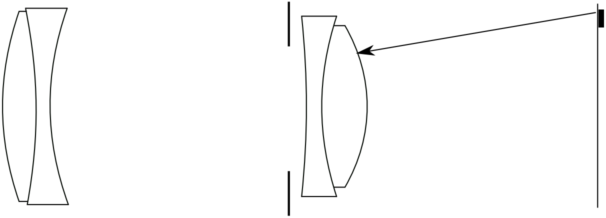
Assignment #2

- Write the "realistic" camera plugin for PBRT which implements the realistic camera model.
- The description of lens system will be provided.
- `GenerateRay(const Sample &sample, Ray *ray)`
    - PBRT generate rays by calling `GenerateRay()`, which is a virtual function of `Camera`.
    - PBRT will give you pixel location in `sample`.

Assignment #2

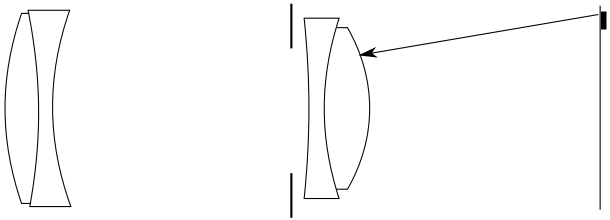
- Write the "realistic" camera plugin for PBRT which implements the realistic camera model.
- The description of lens system will be provided.
- `GenerateRay(const Sample &sample, Ray *ray)`
  - PBRT generate rays by calling `GenerateRay()`, which is a virtual function of `Camera`.
  - PBRT will give you pixel location in `sample`.
  - You need to fill the content of `ray` and return a value for its weight.

Inside GenerateRay()



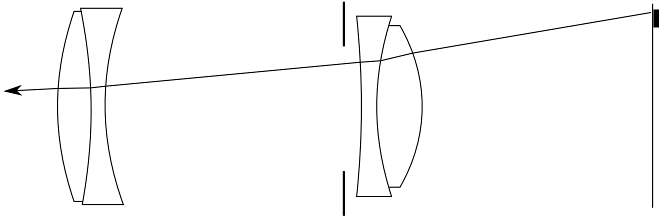
1. Sample a point on the exit pupil uniformly.

Inside `GenerateRay()`



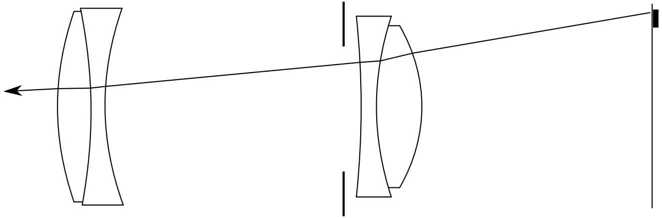
1. Sample a point on the exit pupil uniformly.
   - Hint: `sample.lensU` and `sample.lensV` are two random samples distributed in $[0, 1]$ uniformly.

Inside `GenerateRay()`



1. Sample a point on the exit pupil uniformly.
   - Hint: `sample.lensU` and `sample.lensV` are two random samples distributed in $[0, 1]$ uniformly.
2. Trace this ray through the lens system. You can return zero if this ray is blocked by an aperture stop.

Inside `GenerateRay()`



1. Sample a point on the exit pupil uniformly.
   - Hint: `sample.lensU` and `sample.lensV` are two random samples distributed in $[0, 1]$ uniformly.
2. Trace this ray through the lens system. You can return zero if this ray is blocked by an aperture stop.
3. Fill `ray` with the result and return $\frac{\cos^4 \theta'}{Z^2}$ as its weight.

Submission

- Your source code.
- A brief report describing your implementation and(optional) extensions(ex. importance sampling).
- The rendered result.
- Due date: 11/23

Submission

- Your source code.
- A brief report describing your implementation and(optional) extensions(ex. importance sampling).
- The rendered result.
- Due date: 11/23
  - You have three weeks for this project. However, assignment #3 will be announced two weeks later and due in another three weeks.
  - That is, the earlier you complete #2, the more time you have for #3.

Submission

- Your source code.
- A brief report describing your implementation and(optional) extensions(ex. importance sampling).
- The rendered result.
- Due date: 11/23
  - You have three weeks for this project. However, assignment #3 will be announced two weeks later and due in another three weeks.
  - That is, the earlier you complete #2, the more time you have for #3.
- My email address:
  littleshan@cmlab.csie.ntu.edu.tw
  littleshan@gmail.com