Hw2 - Realistic camera model

1. Implementation:

1. Len data structure:

首先第一步，要先決定好len的資料結構是什麼，於是很簡單的，我就把dat檔給打開來，發現裡面有四欄分別是曲率半徑（radius）、鏡片厚度（axpos）、折射率（N）跟光圈大小（aperture），就把他們都存起來。後來又發現他折射率如果是零代表這邊光圈，只要檢查光線有沒有在光圈裡面就好，不需要去運算折射，所以我多了一個bool叫isStop去存他現在是不是光圈，不能直接看N是因為我把N轉成1了，空氣折射率本來就是1，這樣計算比較保險。

後來又發現鏡片厚度對我計算上來說似乎沒什麼用處，所以我在存資料時改成存每個鏡片的z座標，然後加上厚度就又可以得到下一片鏡子的z座標了。最後再算折射時，因為一直有用到曲率中心，所以我最後也把它給加入資料結構裡面了。總結以下為我的資料結構：

```
struct Len {
    // the z of len
    float z;
    // the radius of len
    float radius;
    // the n of len
    float n;
    // the aperture value of len
    float aperture;
    // true if it's a stop
    bool isStop;
    // the radius center
    Point center;
};
```

2. Constructor:

再來就是讀檔，這邊問題不大，我就把dat讀進來一個個存進去我的資料結構裡，所以現在我有一堆Lens，而每個Lens都有上述那六樣東西。

做完之後，就是要想辦法把Raster space轉成Camera space，做完這部之後就construct完成了。

3. Generate the Ray and Trace it:

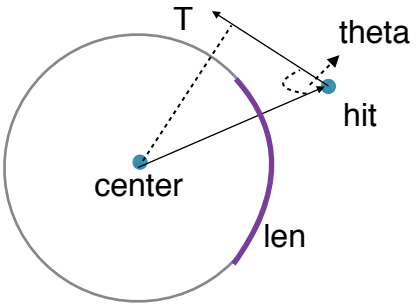
接下來就是實作GenerateRay這個method了，我這邊的想法是我從底片發出一個ray到最後一個透鏡，然後做做看折射，如果可以就繼續折射下去，如果中間有折射後無法到達下一個鏡片的情況就代表這個sample出來的ray是沒有用的，return 0再繼續下去，那至於怎麼產生這條ray，其實要先在最後一個鏡片上sample一個點出來，然後再從底片上sample的那個點連過來，就產生的第一條ray，比較麻煩的當然是在鏡片上sample一個點出來，我這邊用的是投影片的方法去產生：

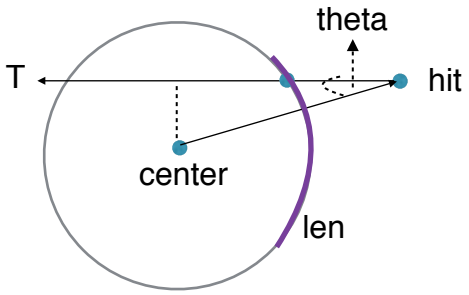
$$r = \sqrt{\xi_1},\ \theta = 2\pi\xi_2$$

然後r再乘上鏡片的半徑，這樣就確保了點會均勻的在盤狀裡面，比較麻煩的事因為鏡片是立體的，所以還需要考慮鏡片的厚度，所以有了sample point的x和y座標之後還要記得用半徑跟xy去算出z座標。

終於找到最後一面鏡子上的sample point之後，也有ray了，就可以去算折射線了，這邊花了我很多時間才想到完全正確，首先我應該要先判斷我現在遇到的是不是光圈，光圈他

就是空的左右兩邊都是空氣，所以根本不用算折射就直直前進，唯一需要注意的就是他有沒有超出光圈的大小範圍，也是寫到這邊才想到資料結構應該要記錄一下現在是不是光圈的，要不然要連續比兩個N都是不是1有點麻煩。

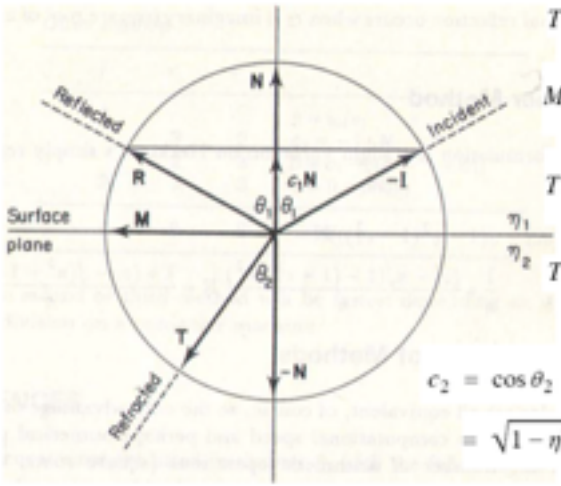




如上圖，hit是我們的出發點其實也就是上一個鏡片的交點，T向量是光線的行徑方向其實也就是上一個鏡片出來的折射線，所以我們需要判定到底這條光線會不會打到這顆球，如果有打到這顆球還要繼續判定到底有沒有打在鏡片上，而第一個hit一定會在鏡片上，因為是我們剛剛辛苦sample出來的，所以第一次做完之後hit不會往前移動但還是會有折射線，因此第二次開始就變成上面這種正常的判斷了。

然後我在程式碼裡面所建的三角形，c2hit就是hit和center之間的向量，projection_distance就是c2hit往T向量去投影出來的，最後normal_distance就是圖上的虛線。有這些之後就可以算hit要移動的距離，才可以到達鏡片的表面。

做完以上判定也移動完hit之後，我們接下來要算折射線了，我用了投影片上的方法：Heckber's method，去算折射線T，圖如以下：



$$T = \sin\theta_2 M - \cos\theta_2 N$$

$$M = \frac{I_{perp}}{|I_{perp}|} = \frac{I + c_1 N}{\sin\theta_1}$$

$$T = \frac{\sin\theta_2}{\sin\theta_1}(I + c_1 N) - \cos\theta_2 N$$

$$T = \eta I + (\eta c_1 - c_2)N$$

$$c_2 = \cos\theta_2 = \sqrt{1 - \sin^2\theta_2}$$

$$= \sqrt{1 - \eta^2 \sin^2\theta_1} = \sqrt{1 - \eta^2(1 - c_1^2)}$$

最後我們就更新好光的行徑方向了，剛剛也更新了hit點，因此就用這兩個東西繼續跑下一個鏡片，直到跑完所有鏡片！

4. Set exposure weight:

本來做完上述之後，GenerateRay要回傳一個weight，我參考其他的檔案如orthographic他們都是回傳1，於是我就照做結果跑出來好像也沒錯但就很亮，然後就看到作業有額外的要求去設定weight，於是我就回去看paper，我發現這篇paper下面有附公式（後來看投影片好像也有），更好的是還有一張圖如下：

$$E(x') = L\frac{A}{Z^2}\cos^4\theta'$$

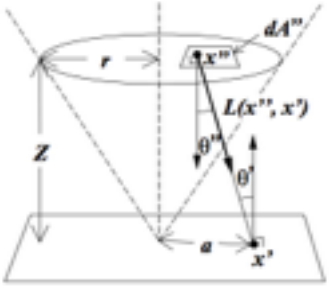
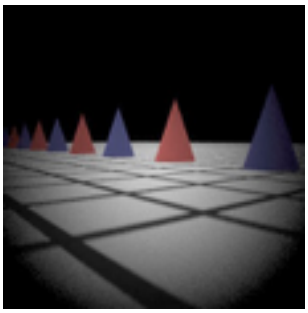


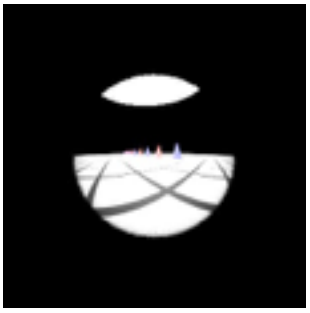
Figure 6: Geometry for computing the irradiance at a point on the film plane and the exact form factor.

於是我就照著這張圖，把他要的東西填進去，A就是圓形面積，而我也有最後一個透鏡的半徑了（aperture），Z就是最後一個透鏡到相機的距離，就拿最後一次hit跟相機位子相減就可以得到了，最後L就是最後出來的折射光，他的cos就是跟法向量做內積，我把它Normalize後再去四次方就沒有長度問題了，於是我就把exposure weight弄好了，去跑出來結果確實有比較暗，只是有點不確定是不是就是這樣做？（希望是）

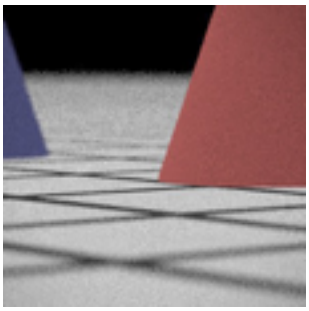
2. Results:



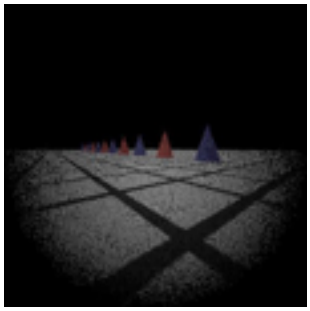
dgauss - 32 samples
Speed: 3.1s



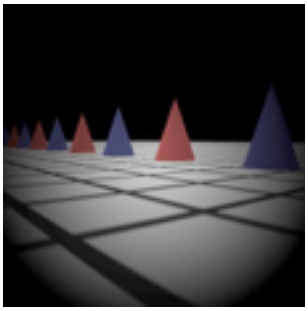
fisheye - 32 samples
Speed: 2.2s



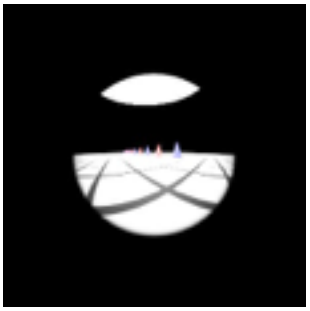
telephoto - 32 samples
Speed: 2.7s



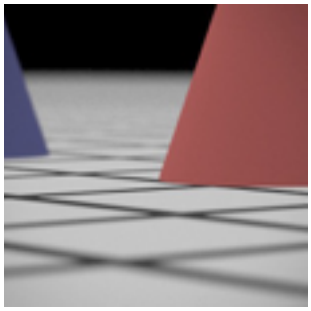
wide - 32 samples
Speed: 1.5s



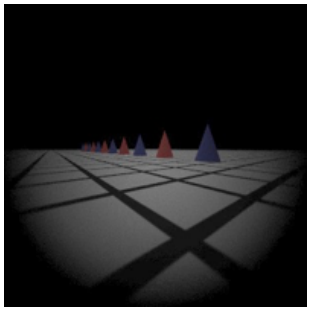
dgauss - 512 samples
Speed: 49.4s



fisheye - 512 samples
Speed: 36.3s

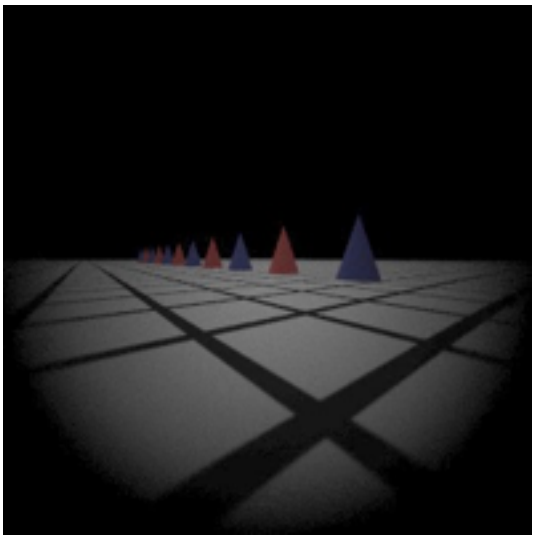


telephoto - 512 samples
Speed: 49.4s

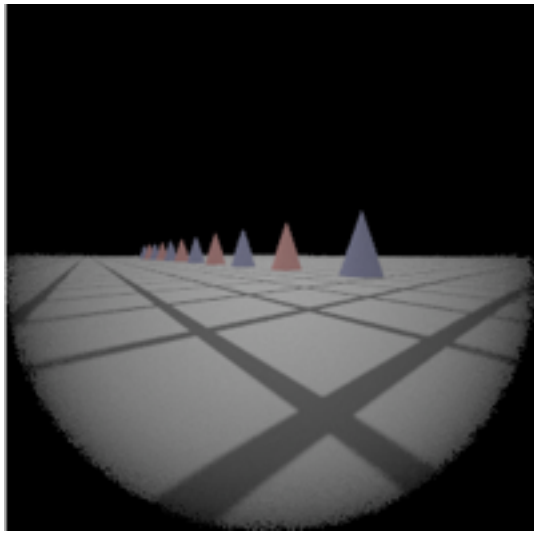


wide - 512 samples
Speed: 27.1s

助教我發現我轉成jpeg之後都會變比較暗耶，這是正常嗎？
例如我把wide - 512 samples拿來截圖比較一下！



jpeg file



exr file

3. Environment:

OS: MacOS
CPU: 2.6GHz Intel Core i5
Memory: 8GB 1600 MHz DDR3
Core number: [default] 4 cores

4. Others:

這次作業我好像沒有做什麼加速，也暫時沒有想到哪邊可以加速，就快到死線了只好就這樣交出去了。

這次做花時間的就是想那個幾何圖形了，所以過程中紙上畫了一堆圓圈跟向量，寫程式也遇到不少問題但還是上次作業比較慘，這次還好剛好都可以問到別人不會的也可以幫別人解答剛好我會的，就還蠻好的，雖然還是頗累，主要參考資料為投影片camera那章跟那個paper，尤其是paper他都可以補足有些我投影片看完不太確定的地方，包含解析dat的內容、看光圈的意義、一堆座標正負方向的問題等等，但有些數學其實還是沒有完全懂原理，例如exposure weight那邊，就只是把公式要的東西帶進去，然後網路資料也幫助我不少，但大多就是C++語法跟一些演算法的原理（Heckber's method）。

最後因為我覺得從exr轉成jpeg之後實在變得太暗了，所以我決定把所有exr都打包一起丟上去，如果助教覺得我上面的圖有點問題的話可以看看我的exr結果！