Homework 2 – Realistic Camera

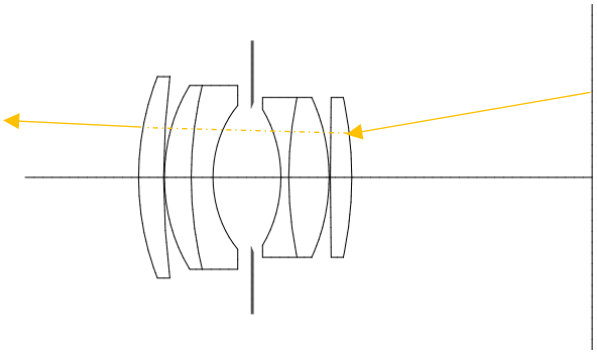
Digital Image Synthesis

R05944045 陳卓晗

Date submitted: 11/24/2016

1. Implementation

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



在本次實作中，Realistic Camera 模擬光線從底片出發，用在真實的透鏡系統中進行折射之後形成的折射光線來進行 ray tracing 的計算。如上圖所示，圖左中的表格代表透鏡組中每個透鏡的參數，包括半徑、厚度、折射率、光圈大小等。圖右中最右邊是追蹤光線射出的底片所處的位置，光線從這裡出發，觸碰到第一個透鏡之後開始折射，如果光線行進中沒有被光圈遮擋的話，最後就從最左邊的透鏡折射出去進行後續的計算。

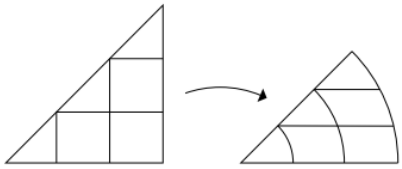
根據實作中用到的參數，計算過程主要可以分為以下幾步：

1) Camera Transformation

Realistic camera 中的 camera transformation matrix 和 projection 中大致相似但略有不同。Raster space 和 camera 的位置都是在 camera

space 的原點處。而 camera 的透鏡組會在 camera 的正前方。所以 tracing ray 會從 z = 0 的位置出發。另外，由於經過透鏡之後底片的成像會上下左右顛倒，所以矩陣也需要經過上下左右對調處理。screen[4] 的值可以通過 filmdiag 和 film->xResoution，film->yResolution 求出。

2) Sampling



為了計算穿過整個透鏡的光線對底片上某一點的能量總和，我們需要對經過整個透鏡的光線進行積分計算，這很難辦到，所以採用 Monte Carlo 方法來對這個積分進行近似。即對整個透鏡進行多次隨機的平均採樣。在這裡使用了 pbrt 書中的方法將一個$[-1, 1]^2$的正方形映射到一個同心圓上，這樣可以使透鏡上的採樣更加平均，pbrt 同樣提供了該方法的實作 ConcentricSampleDisk()。採樣之後我們就能得到從底片發出的初始光線的方向。

3) Intersection

由於透鏡其實就是一個球體，所以需要求的就是球心在 z 軸上的球和射線的交點。在 pbrt 第三章 shapes 裡面也提到過球體和射線的求交算法。需要注意的是如果透鏡是凸透鏡，那麼我們需要取比較遠的交點，如果透鏡是凹透鏡，需要取比較近的交點。

4) Refraction

光線在觸碰到透鏡的表面時會發生折射，為了計算折射之後的光線，光線的入射點已經在之前的 intersection 的步驟中算出。根據提供的講義，求折射之後的光線有三種方法，ppt 中的對比分別如下：

Whitted's Method

√	/	×	+	
	1			$n = \eta_2/\eta_1$
	3	3	2	$I' = I/(-I \cdot N)$
			3	$J = I' + N$
1	1	8	5	$\alpha = 1/\sqrt{n^2(I' \cdot I') - (J \cdot J)}$
		3	3	$T' = \alpha J - N$
1	3	3	2	$T = T'/\lvert T' \rvert$
2	8	17	15	TOTAL

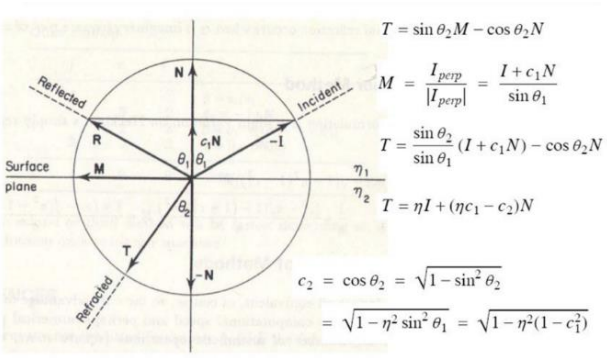
Heckbert's Method

√	/	×	+	
	1			$\eta = \eta_1/\eta_2$
		3	2	$c_1 = -I \cdot N$
1		3	2	$c_2 = \sqrt{1 - \eta^2(1 - c_1^2)}$
		7	4	$T = \eta I + (\eta c_1 - c_2)N$
1	1	13	8	TOTAL

Other Method

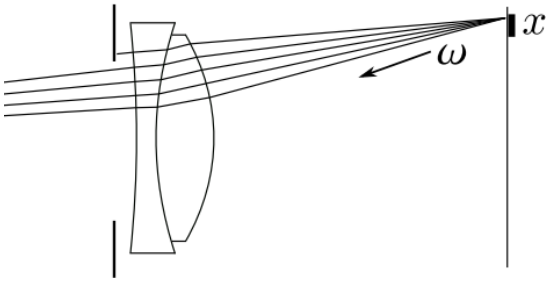
√	/	×	+	
	1			$n = \eta_2/\eta_1$
		3	2	$c_1 = -I \cdot N$
1		2	3	$\beta = c_1 - \sqrt{n^2 - 1 + c_1^2}$
	3	3	3	$T = (I + \beta N)/n$
1	4	8	8	TOTAL

其中 Whitted’ s Method 需要最多的計算次數，Heckbert’ s Method 則需要計算 23 次，Other Method 需要計算 21 次。



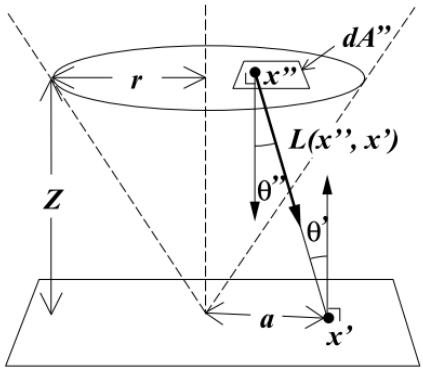
但是由於 Heckbert’ s Method 只需要計算一次除法，而且可以被優化，所以使用 Heckber’ s method 來計算折射之後的光線的計算量會略少與其他幾種計算折射的方法。

5) Aperture



光線在折射的過程中，有些光線會射出透鏡組的範圍之外，或是被透鏡組中間的快門光圈所遮擋。這一部分光線對最終的成像沒有貢獻，也不需要進行後續的折射計算。所以，在每次折射過程中計算光線是否有落在對應透鏡的光圈範圍之外，若有的話可以直接終止計算并返回 0.0f。由於透鏡光圈的圓心都是在 z 軸上，所以只要判斷在透鏡的交點處$x^2 + y^2 \leq aper^2$是否成立就可以，如果不成立表示光線落在光圈之外。

6) Exposure



真實的相機受到 vignetting 的限制，當光線和透鏡的角度過大時光線會減弱，造成在成像的邊緣會逐漸變暗的效果。對於每條採樣的光線，根據 Kolb 論文中的算法取$\text{weight} = \text{L}\frac{A}{Z^2}\cos^4\theta$。因為我們 是根據最右邊的透鏡做的採樣，所以其中 Z 是底片到最右邊透鏡的距離，即 filmDistance，A 是第一塊光圈的面積。論文中又提到，當 x’ 和 x’ ’ 之間的夾角很小時，θ′可以假設成與 x’ 和圓盤的中心的夾角。

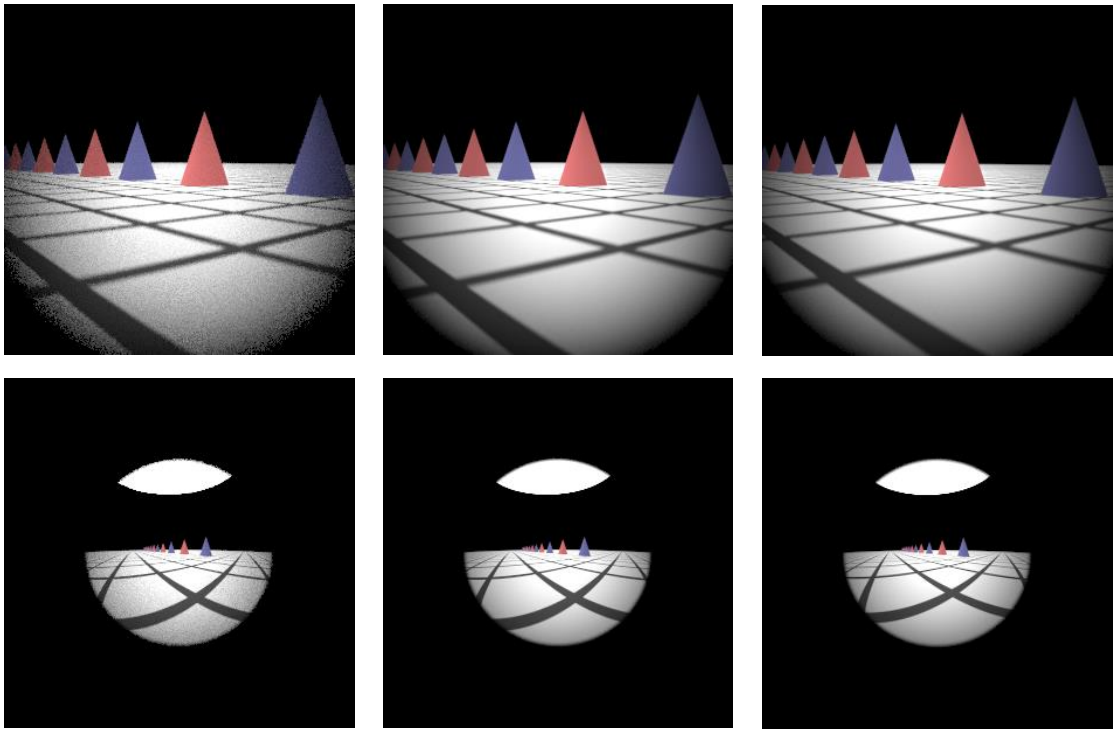
2. Performance

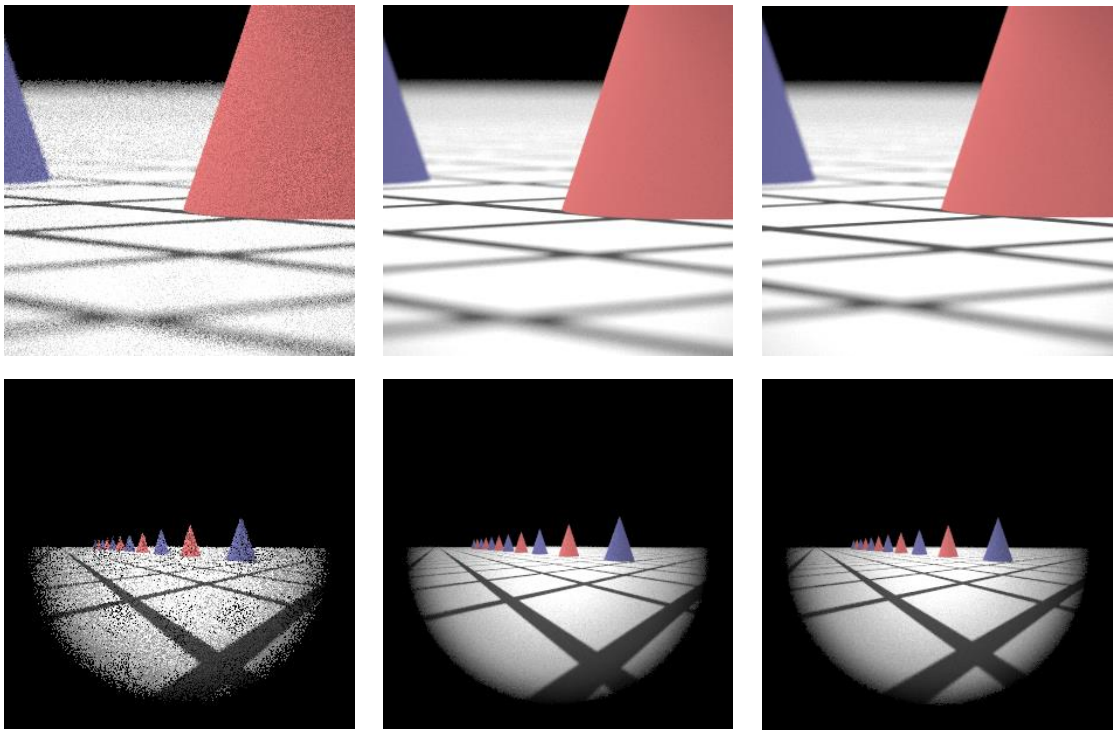
Environment	CPU	Memory	Cores	GPU
Windows 10	Core i7 2.70GHz	8GB	8	GTX 970m

	32ref	512ref
Dgauss	48.3s	825.6s
Fisheye	34.3s	548.7s
Telephoto	42.7s	708.5s
Wide	21.2s	357.2s

3. Results

My Implementation 32ref | My Implementation 512ref | Reference





注意到我的實作結果中圖像的距離比 reference 中的距離近了一點點。猜測是因為在 camera space 中 camera 所定義的位置不同造成的。若把最左邊的透鏡處設為原點，那麼相當於整個照相機往後移了一段距離，看到的圖像“範圍”會稍大一點。