Advanced Computer Graphics
Assignment III-Runtime experiment

張洪瑞
ZHANG,HONG-RUI
40171212H@ntnu.edu.tw

I. 實驗背景

本次實驗將透過光線追蹤(ray-tracing)技術，進一步應用在較大型測試資料之物體碰撞測試上。其中針對各場景執行時間進行了基本分析，並提出相關數據以作為下次作業效能改進的實作探討。

程式碼部分主要由 c++11 撰寫，演算法部分在光線與三角形碰撞的實作則採用 Möller-Trumbore [1]。實驗過程中，僅對於些微參數作調整，其餘程式碼部分均保持固定。

II. 光線追蹤演算法

有別於先前作業中較簡單之場景(單一球體及兩面三角形)，以固定大小的陣列儲存重複的物體，我使用動態的陣列(std::vector)儲存輸入檔案所讀取的三角形(兩筆測試資料各為 970 片及 69,668 片左右)。需特別注意的是，我們無法從載入的順序得知該物體在幾何上距離的遠近，為了避免先畫物體的被後面物體掩蓋掉情況，我們需根據演算法回傳的距離值 *t* 判斷在特定方向所呈現的物體影像。

根據 Möller 及 Trumbore，我們可把光源(觀察者)座標 *O* 和光行進(視線)方向之向量 *D* 以下向量關係式表示:

$$P = O + tD \tag{1}$$

同時，每片空間中的三角形，給定頂點座標 *A*，*B*，*C*，即其位置係數 *u*，*v*，*w*。其位置 *P* 滿足下式:

$$P = vA + uB + wC \tag{2}$$

一旦光線碰撞到三角形，(1)與(2)兩式相等，即:

$$O + tD = vA + uB + wC \tag{3}$$

欲解 *t*，若 *t* 為正值代表物體在光源(觀察者)的前方，若有上述所提到之遠近物體重疊之情形，仍須找到最小的 *t* 所對應的三角形，進一步透過表面顏色(RGB 值)，環境光(ambient)、反射光(diffuse)及鏡射光(specular)係數將物體渲染出來。

每次碰撞結果回傳為真時，呈現在畫面上的像素，必須根據 Phong model [2]的計算決定。以下簡述其相關原理:

A. 表面顏色(RGB)

每一片三角形皆有一個對應的 RGB 值，分別表示影像在電腦中紅(Red)、綠(Green)和藍(Blue)，範圍從 0-255，由輸入資料提供。

B. 環境光(ambient)

本次沒有給定光源的顏色，故光源預設為白色光，效果等同為環境光係數 *Ka* 乘上表面 RGB 值。

C. 反射光(diffuse)

反射程度 *Id* 由關係式 $Id = Ii \times N \cdot L$ 決定，整體效果為 *Id* 乘上係數 *Kd* 後之結果。本次僅實作光源表面反射，故底部之三角面無法呈現出上面物體之倒影。

D. 鏡射光 (specular)

鏡射係數 *Ks*，鏡射程度 *Is* 由關係式 $Ii \times (N \cdot L)^n$ 決定。

III. 實驗與結果

A. 測試問題集

本次給定之測試資料為 Blender Suzanne 以及 Stanford bunny [3]。Suzanne 共由 970 片三角形所構成，Stanford bunny 則包含了 69,451 片三角形。

表 1: 物體個數

	Hw2	Suzanne	Bunny
Size	3	970	69,668

其他參數包括:屏幕解析度，預設為 25x25 個窗格(frame)。解析度影響畫面中所看到物體的清晰程度；當畫面被切分成愈細的窗格時，進行光線碰撞的次數愈多，也能愈準確的反應出物體表面的形狀。

B. 實驗設定

處理器規格為 Intel Core i7 4700HQ 處理器，支援 4 核心 8 執行緒；主計憶體 DDR3L 1600 MHz SDRAM。該規格之處理器亦曾被使用於量測 KITTI benchmark 所提供之 Odometry 測試資料集在無人車駕駛之電腦視覺處理的光線追蹤技術上[4]。

C. 實驗結果

以下表格展示三筆資料進行光線追蹤碰撞比較。根據執行時間分析，其時間複雜度之成長與三角形個數呈現正比。

表 2: 執行時間(單位:sec)

	Hw2	Suzanne	Bunny
Average	.108	17.497	1,139.174
Best	.110	17.251	1,146.303
Worst	.213	17.506	1,151.747

所幸在針對此次實作的兩筆測試資料，尚無需考慮折射與光線遞迴部分，計算 Stanford bunny 最壞情況下所得之執行時間，約二十分鐘左右。惟一般場景所包含之物體個數遠大於此，若是要將此傳統之作法拓展到商業用途，勢必就「每經過一個窗格必須檢查光線是否有交到物體表面」的檢查這部分進行改良。關於此部分，近幾十年來已有學者提出諸如 BVH 及 SEADS[5]等避免窮舉的演算法，我將於下次的報告中提出相關實驗成果。

接下來我將展示實驗過程中因碰撞判斷條件改變造成之輸出圖像結果。

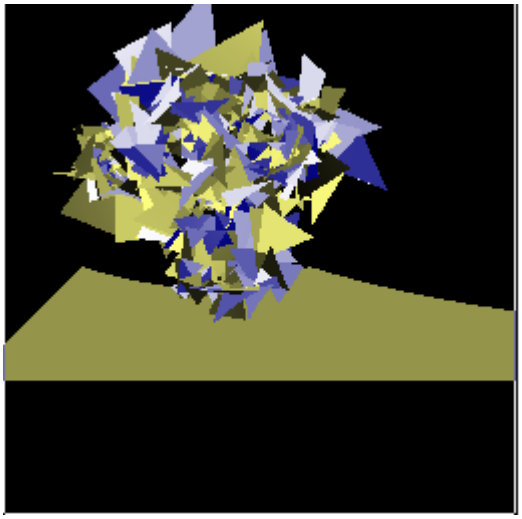

圖 1: 未處理多餘三角形覆蓋之 Suzanne

整體而言，我在實驗 Stanford bunny 之渲染正確性遠高於 Suzanne。個人推測會造成結果明顯差異之原因為 Stanford bunny 中三角形之分布較 Suzanne 均勻。圖 1 可看出我初次針對 Suzanne 進行碰撞渲染所得出之結果，Suzanne 的臉部主體為後方藍色部分，上方被多餘的黃色三角片所覆蓋。在這些多餘的三角片中，有許多是視線「後方」的物體。理論上， Möller-Trumbore 演算法所回傳的 t 值可用於判斷物體是否位於視線前方。原先所設定之門檻為一接近零之常數。圖 3 為將該門檻值限制更小時，得出的結果。儘管至實驗結束前，Suzanne 臉上之多餘三角形仍無法完全消除，其表面輪廓已約略可辨識。

類似的發現亦產生在 Stanford bunny 成像的過程中，該筆測試資料包含了一個完整的兔子，在處裡每次碰撞的三角形遠近問題時，若不對此部分進行特別處裡，一樣會產出完整的模型。比較之後可發現細微差異，兩隻兔子身上的絨毛，圖 3 為未做三角形遠近處理的成像，略顯粗糙，且圖片像素顆粒較大。圖 4 則在物體材質上有柔軟的起伏，與標準輸出檔較雷同。

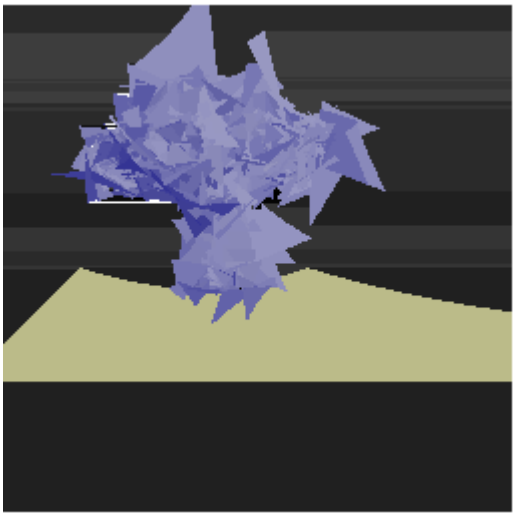

圖 2: Suzanne 輸出結果(曝光後)

圖 3: Stanford bunny（未做三角形距離遠近之處理）

Stanford bunny 另一部分較 Suzanne 完整之處在其底下黃色三角形可見物體的陰影部分。在給定的標準輸出圖檔中，Suzanne 與 Stanford bunny 底下皆有倒影反射，相較於主體在諸如此類之細節在在說明 Phong model 看似簡明扼要，卻無往不利。

與前一次作業的測資比較，同樣為單一場景，CPU 消耗率在處理 Blender Suzanne 以及 Stanford bunny 時均由作業系統配置了 12 單位的 CPU 資源(單執行緒)，其平均 CPU 在開始進行渲染處理時平均 CPU 可達 12.4~13 之間。

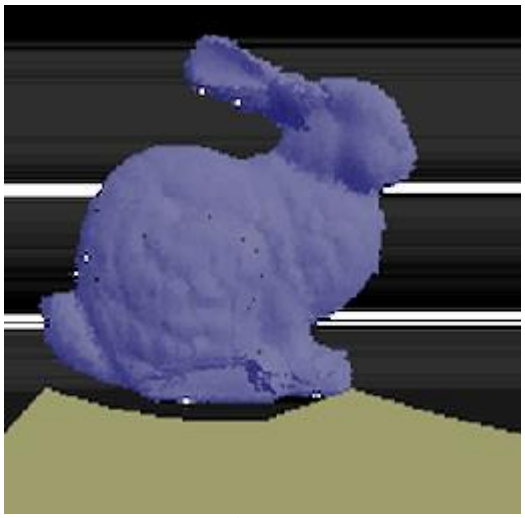

圖 4: Stanford bunny

IV. 心得與討論

碰撞渲染最終得出的成果，儘管正確率較先前提升，同時也觀察到背景出現不明的條狀紋帶。由於計算背景時，若沒有任何碰撞發生，不會計算出顏色。此在靜態影像中的渲染可謂異常。個人推測在每次計算碰撞時，重複計算使得前次資料的碰撞軌跡留下殘影，故希望將來在引入包圍體面積時能重新檢視相關儲存物體資料結構的應用，來消弭此一現象。

V. 結論

我們可以觀察到，未做加速處理前，平均執行時間接近線性成長。傳統的光線追蹤演算法在面臨大型物體的碰撞測試時，勢必尋求效能上的改善以獲得整體效能的高效化。能夠在本次實驗中體認到演算法在圖學優化上的重要性，我想一面也必須仰賴硬體的配合，如現今圖形處理器產業方興未艾，若是加上創意融入電影動畫中，計算機圖學的前景相信將會是榮景可期。

參考文獻

[1] Möller, Tomas, and Ben Trumbore. "Fast, minimum storage ray/triangle intersection." *ACM SIGGRAPH 2005 Courses*. ACM, 2005.

[2] Phong, Bui Tuong. "Illumination for computer generated pictures." *Communications of the ACM* 18.6 (1975): 311-317

[3] Turk, G., & Levoy, M. (1994). The stanford bunny.

[4] Rawashdeh, Samir A., and Mohamed Aladem. "Toward autonomous stereo-vision control of micro aerial vehicles." *Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS)*, 2016 *IEEE National*. IEEE, 2016.

[5] Fujimoto, Akira, Takayuki Tanaka, and Kansei Iwata. "Arts: Accelerated ray-tracing system." *IEEE Computer Graphics and Applications* 6.4 (1986): 16-26.