光柵化與光線追蹤

電腦圖學中的兩大渲染方式

製作:許展維

CONTENT

目　　錄

介紹

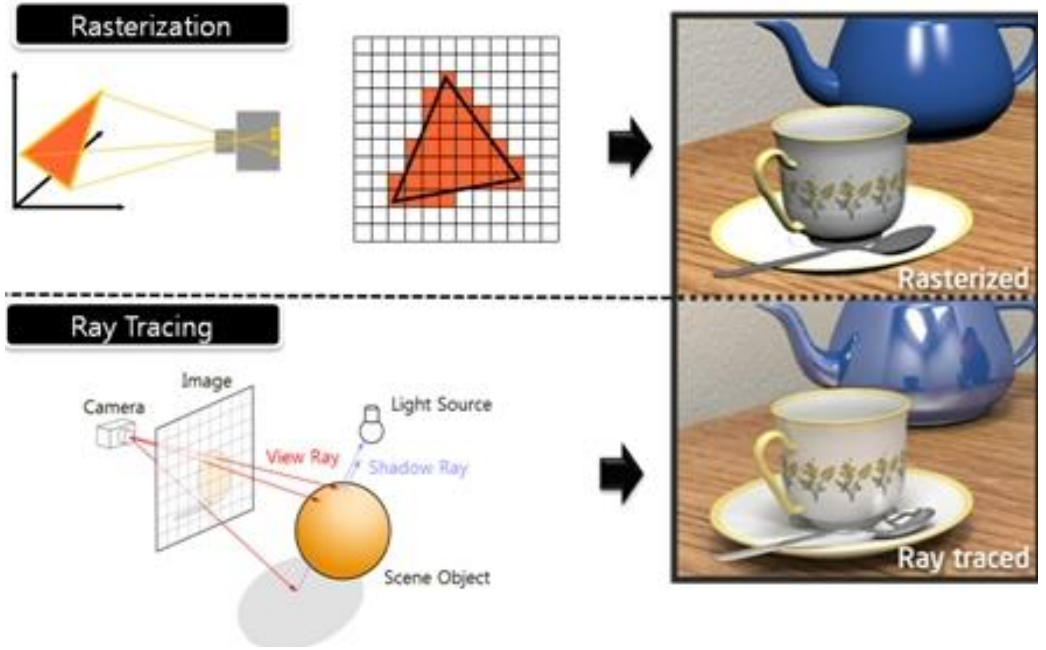
電腦顯示的3D物體, 大致分成兩種主要的渲染方式

1. 光柵化(Rasterization)
2. 光線追蹤(Ray tracing)

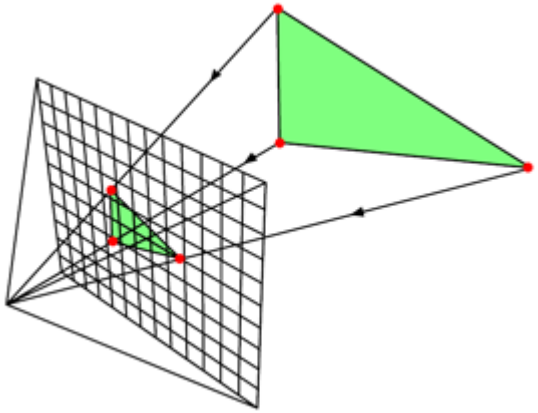
從技術上介紹兩種繪製方式的差別

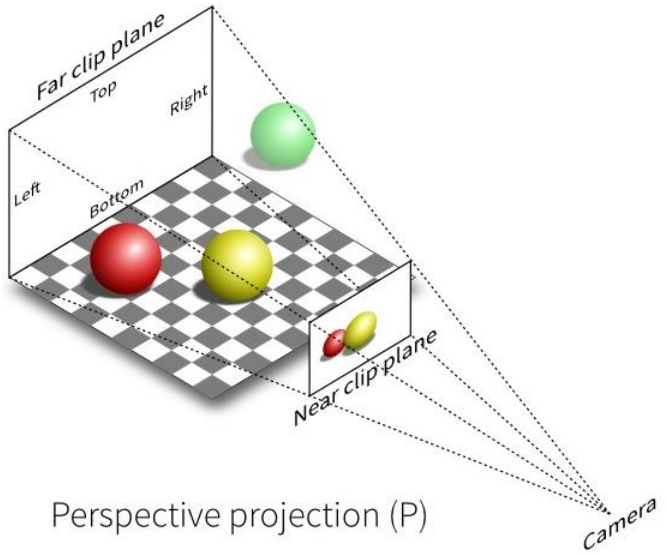
光線追蹤為什麼需要加速?

展示光線追蹤的實際應用



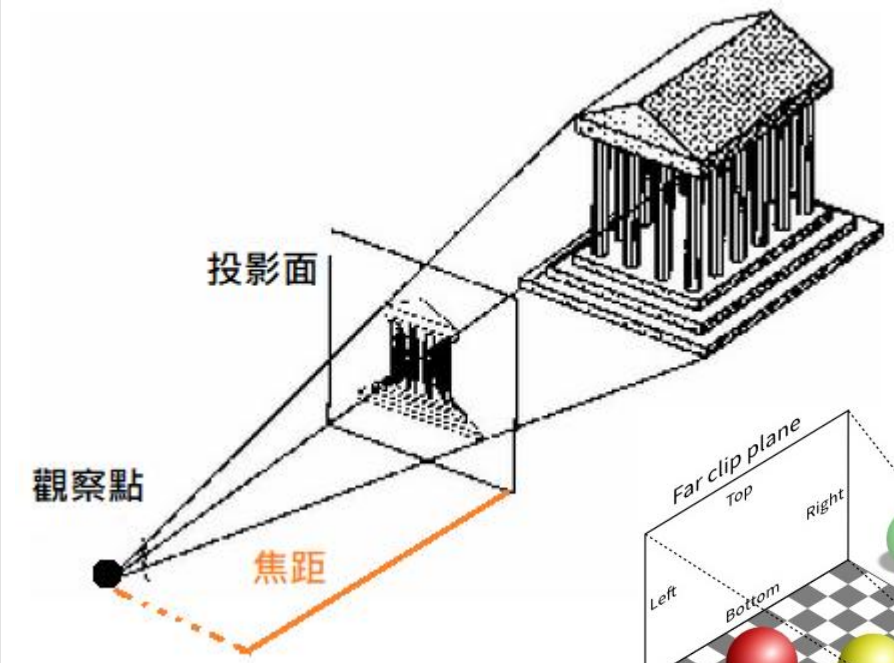
光柵化 Rasterization

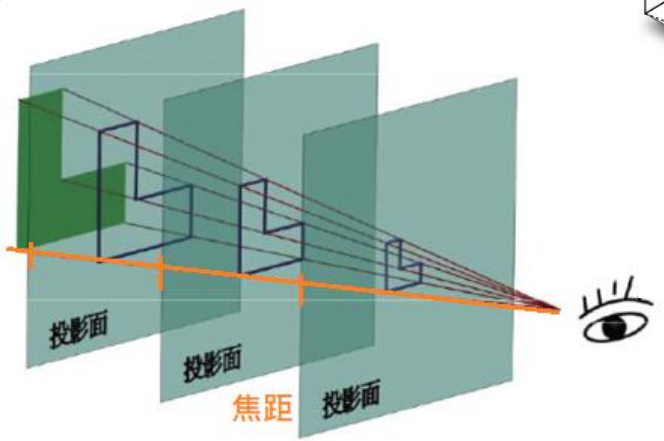


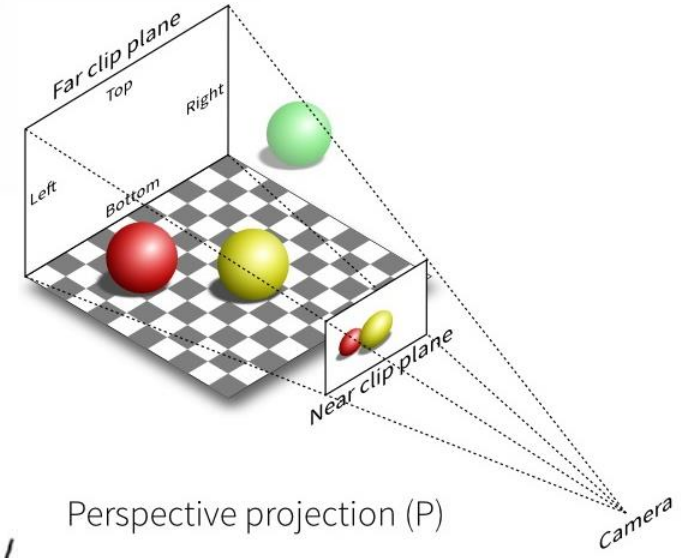


Perspective projection (P)

光柵化：透視投影

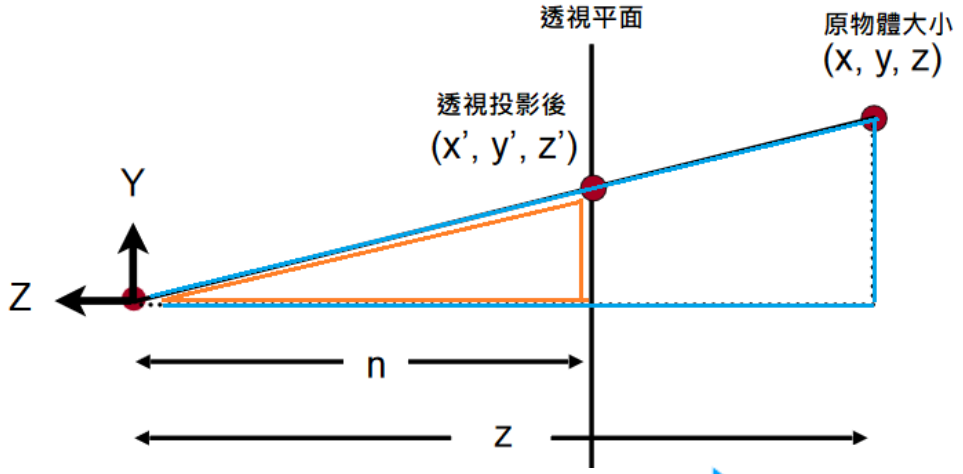






Perspective projection (P)

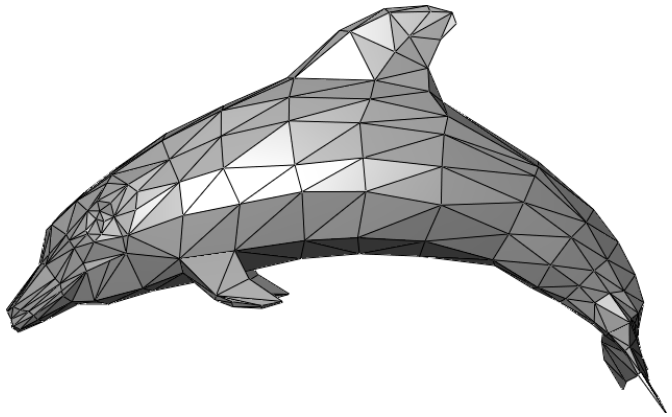
- 透過模擬透視圖的方式, 給定一個投影面來顯示物體
- 以模擬現實中近大遠小的效果
- 而這個轉換的過程, 稱為透視投影

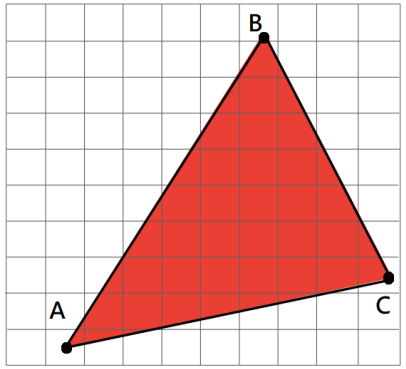


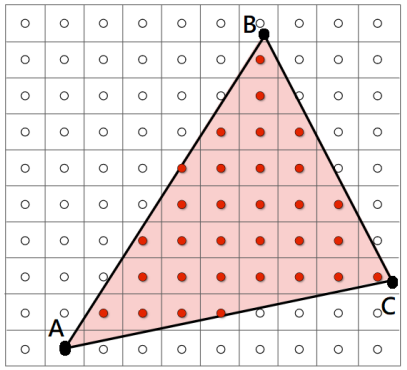
透過相似三角形的比例
y'/n = y/z

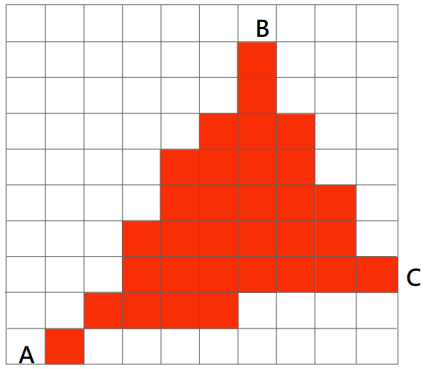
$$y' = \frac{n}{z} y$$

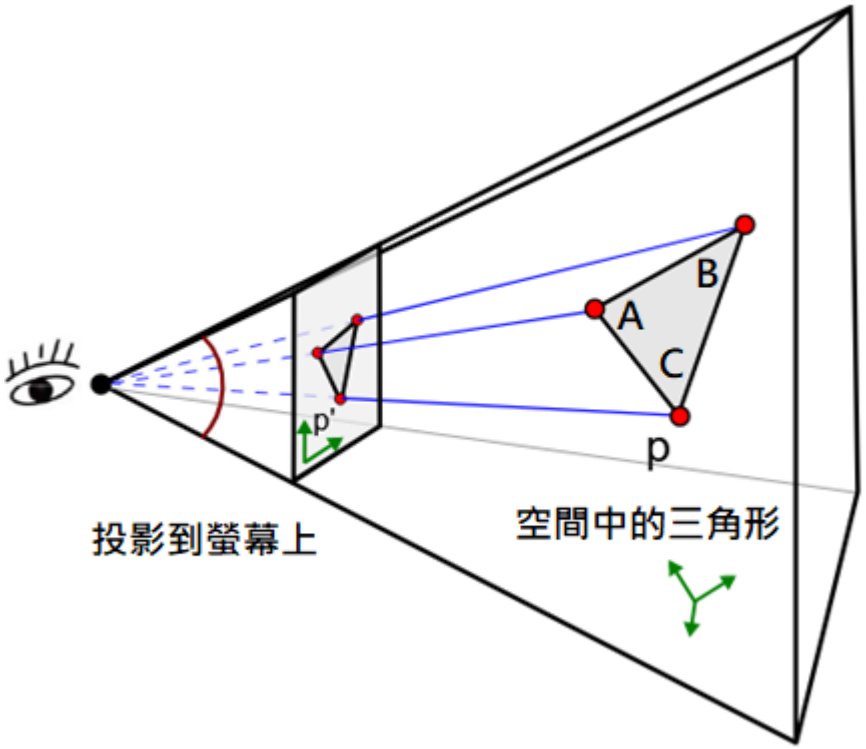
光柵化 : 三角形





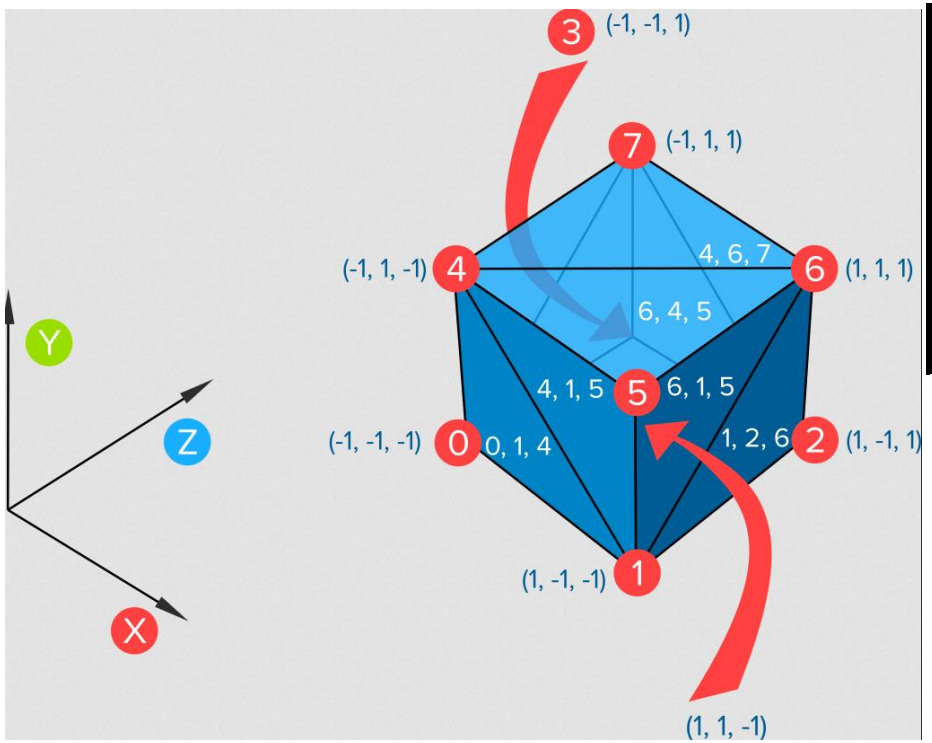


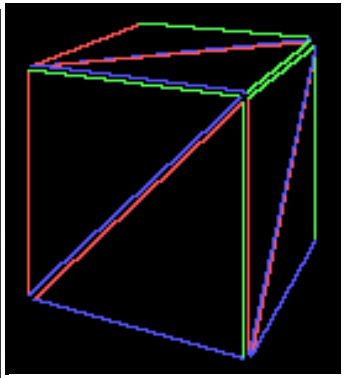




- 三角形是構成一個平面最簡單的形狀
- 一個物體其實是由無數個三角形組合而成
- 顯示時就是先將空間中3個點投影到螢幕上, 再將所包圍的區域填上對應的顏色
- 這個過程叫做光柵化(Rasterization)

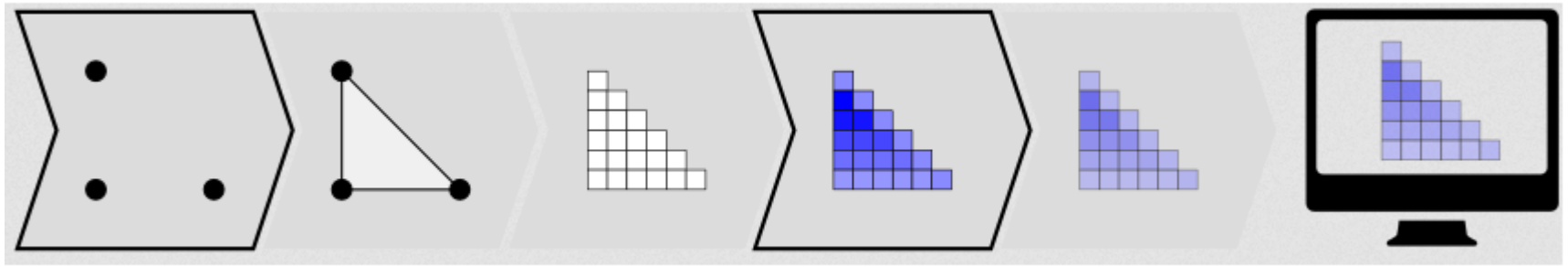
光柵化：繪製一個物體





- 一個正方體可以透過12片三角形組成
- 經過透視投影與光柵化繪製，可以在螢幕上得到正確的畫面
- 但是感覺怪怪的
- 少了著色(Shading)，也就是光影

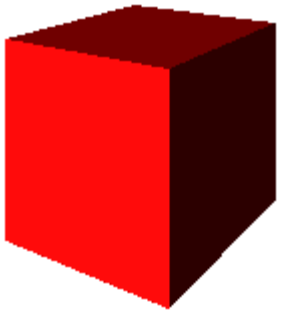
1.計算投影後的點



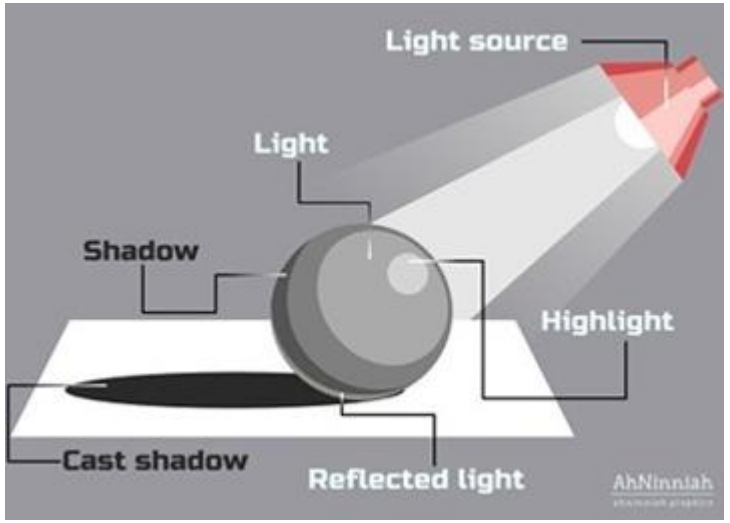
2. 將指定的點組成一個三角形

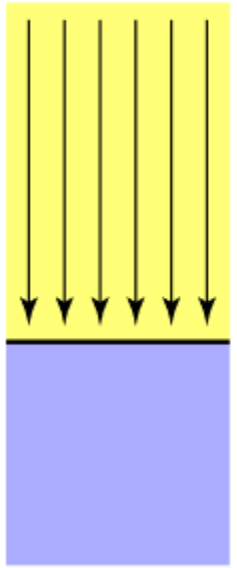
3. 光柵化轉換成像素

最後顯示在螢幕上

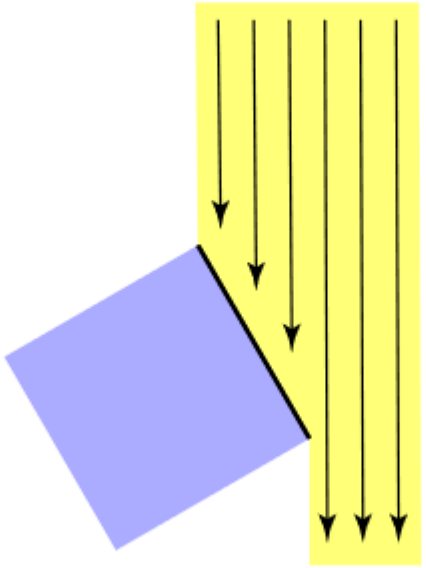


光柵化：光影著色

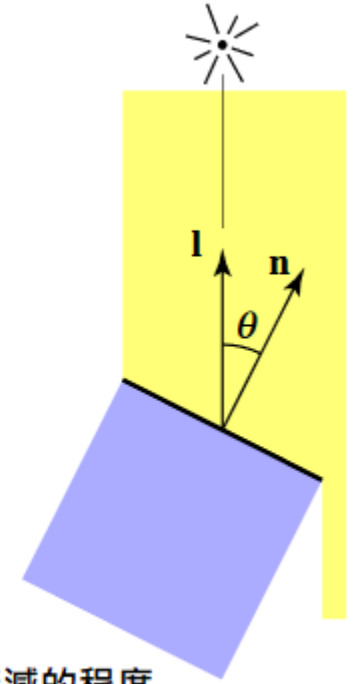




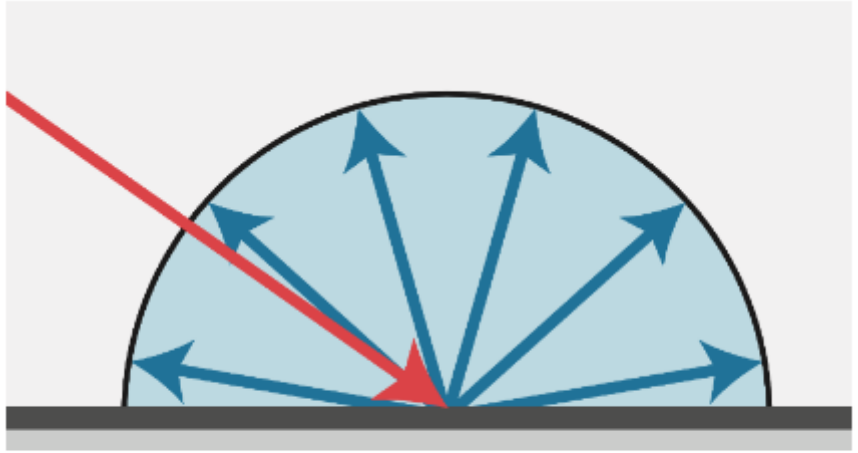
光完全打在平面上
所以顏色最為飽和



當物體旋轉後，接收的光變少了
所以顏色就會暗一些

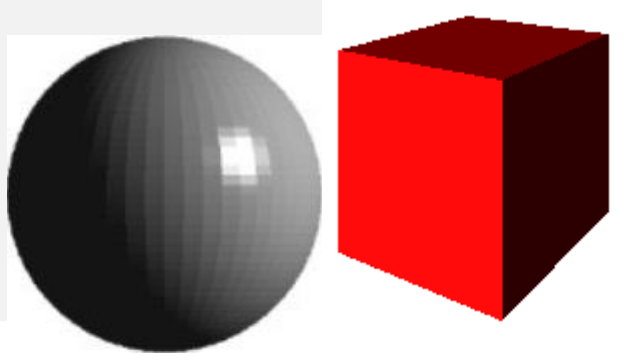


顏色衰減的程度
根據光線與平面的夾角決定



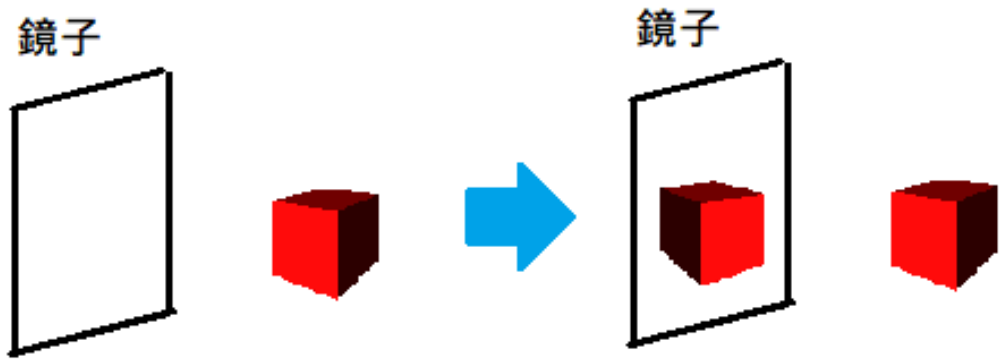
漫反射: 光打到平面後,會向四周平均的散射,所以該平面的顏色只會被光線的角度影響,與觀察者的視角無關

- 光柵化渲染中,光影都是透過假設來近似現實
- 實際物理上應該更嚴謹
- 最基本的漫反射光加入後：

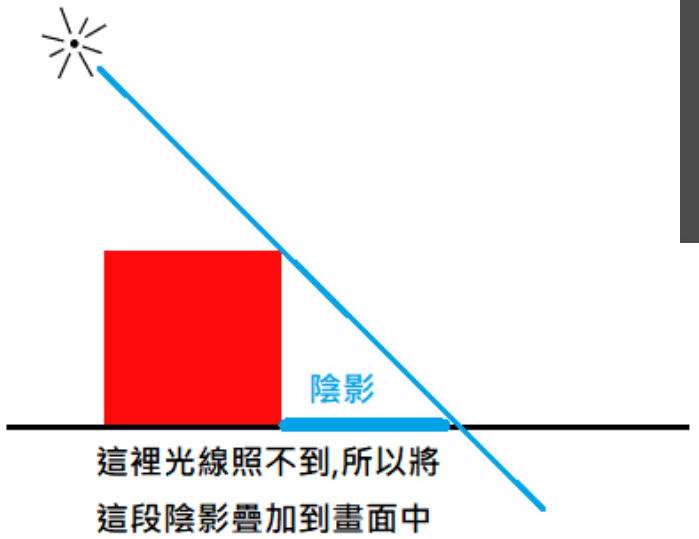


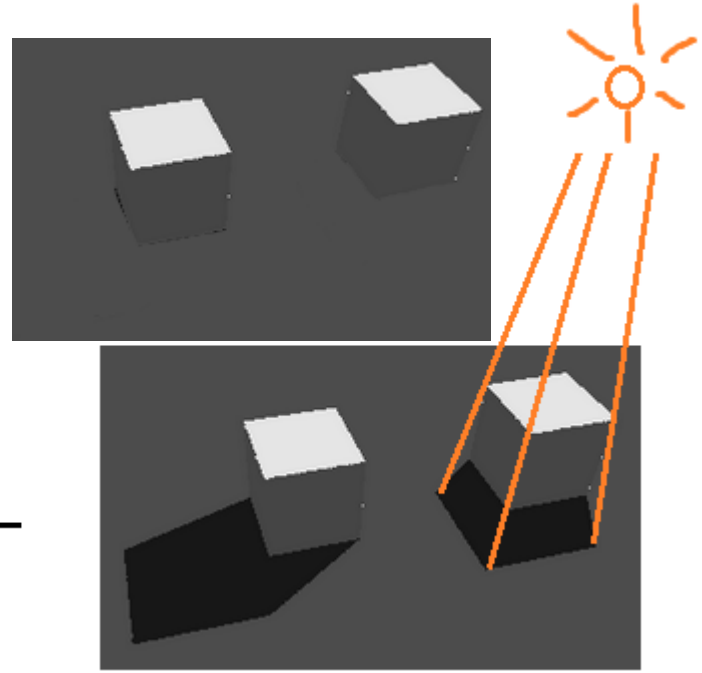
光柵化：鏡面、玻璃、陰影

- 光柵化渲染, 本質上只是把一個物體投影到平面,沒有辦法模擬真實光線的反射、折射、陰影
- 所有關於陰影、玻璃、鏡子等需要模擬光線的狀況，都是透過二次貼圖來近似，所以相對不真實



直接在鏡面貼上物體顛倒的畫面
而不是真的模擬光線的鏡面反射成像



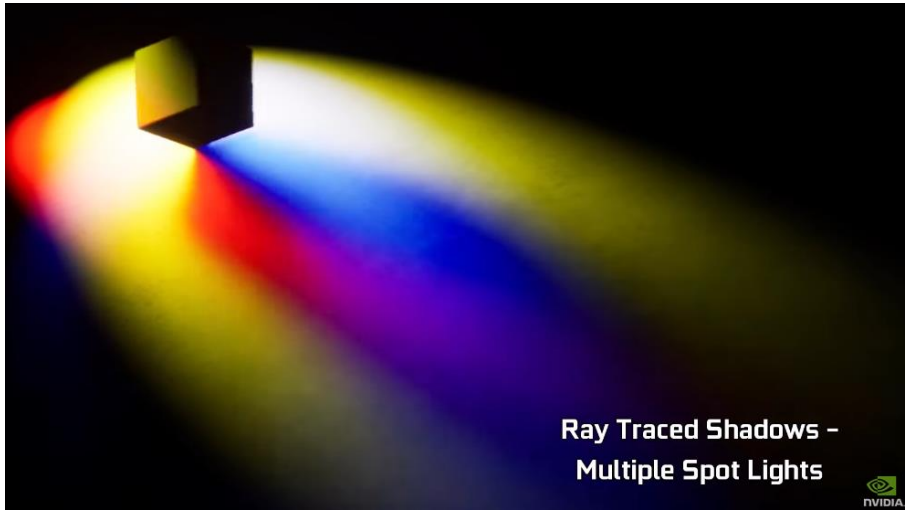


2077鏡面反射BUG,照鏡子都會變光頭

光柵化：優缺點

- 最主要的優點就是快
- 光柵化渲染,本質上只是把三個點投影到平面,並將它填滿顏色, 所以運算速度快
- 電腦不足以即時模擬光線追蹤時的最佳方法

光線追蹤顯示多光源的陰影



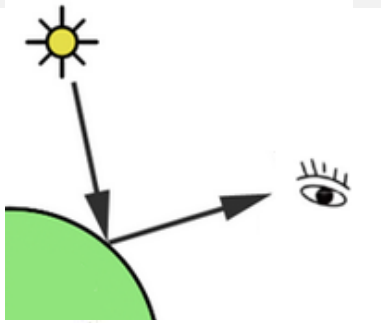
光柵化顯示多光源的陰影

- 缺點也很明顯
- 就是不夠真實, 幾乎大多數狀況都是用假設的方式來代替模擬現實狀況
- 很難正確模擬多光源與真正的鏡面反射與折射

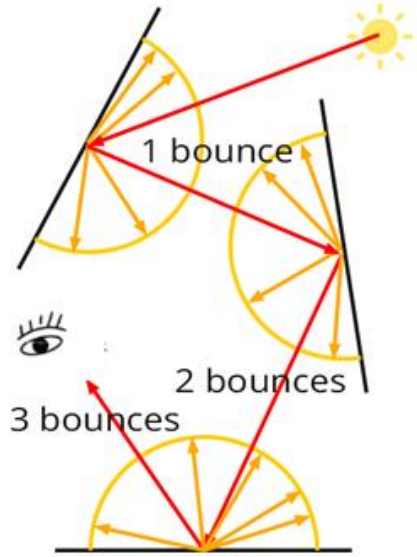
光線追蹤 Ray tracing

光線追蹤：顯示原理

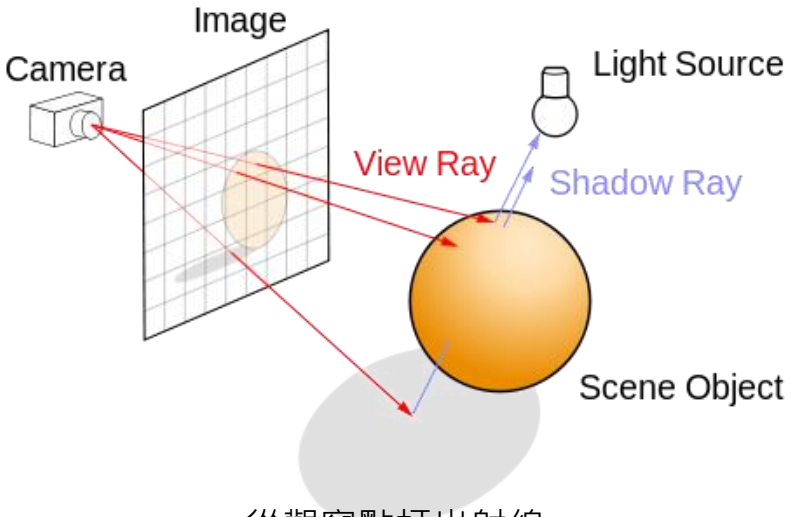
- 我們看的到物體是因為光線打在物體上, 然後再反射到眼睛, 跟相機的像素感測器類似
- 反過來說, 也可以從眼睛的位置, 往外打出射線, 去判斷這條射線經過各種反射後, 有沒有打到光源



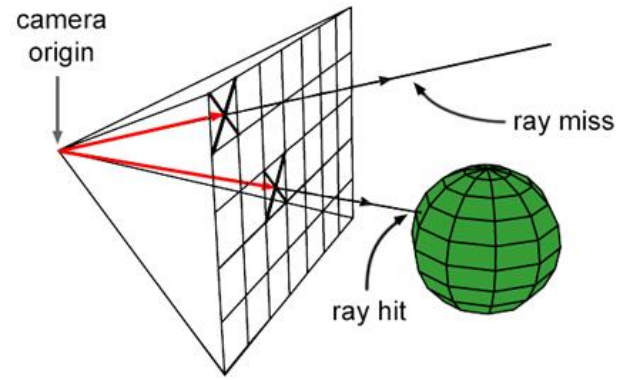
光會向四周均勻的輻射

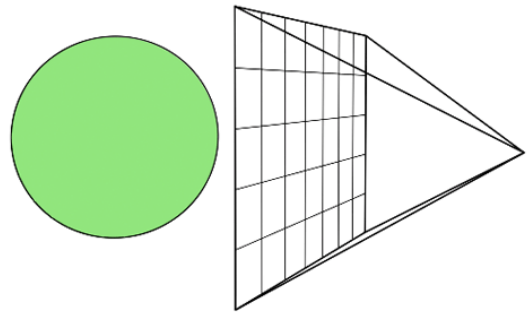


光線的多次反射

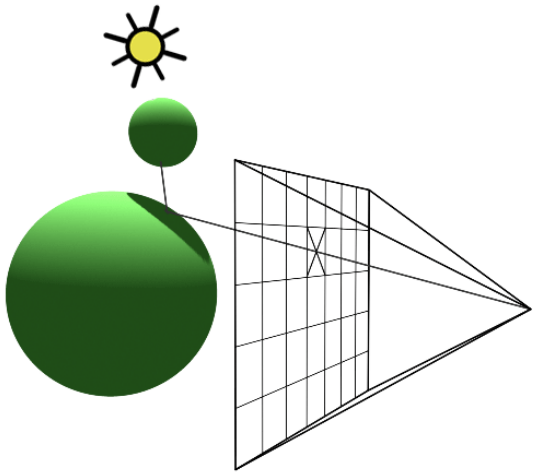


從觀察點打出射線





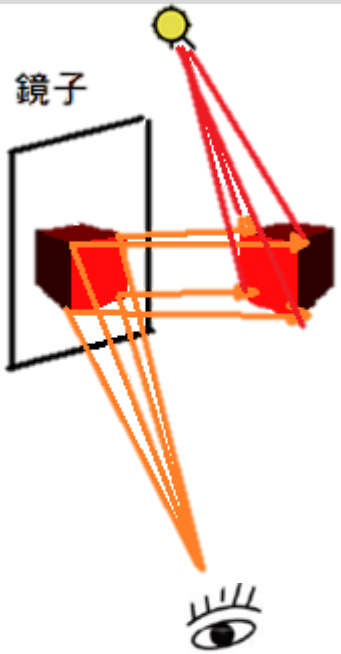
實際繪製的過程

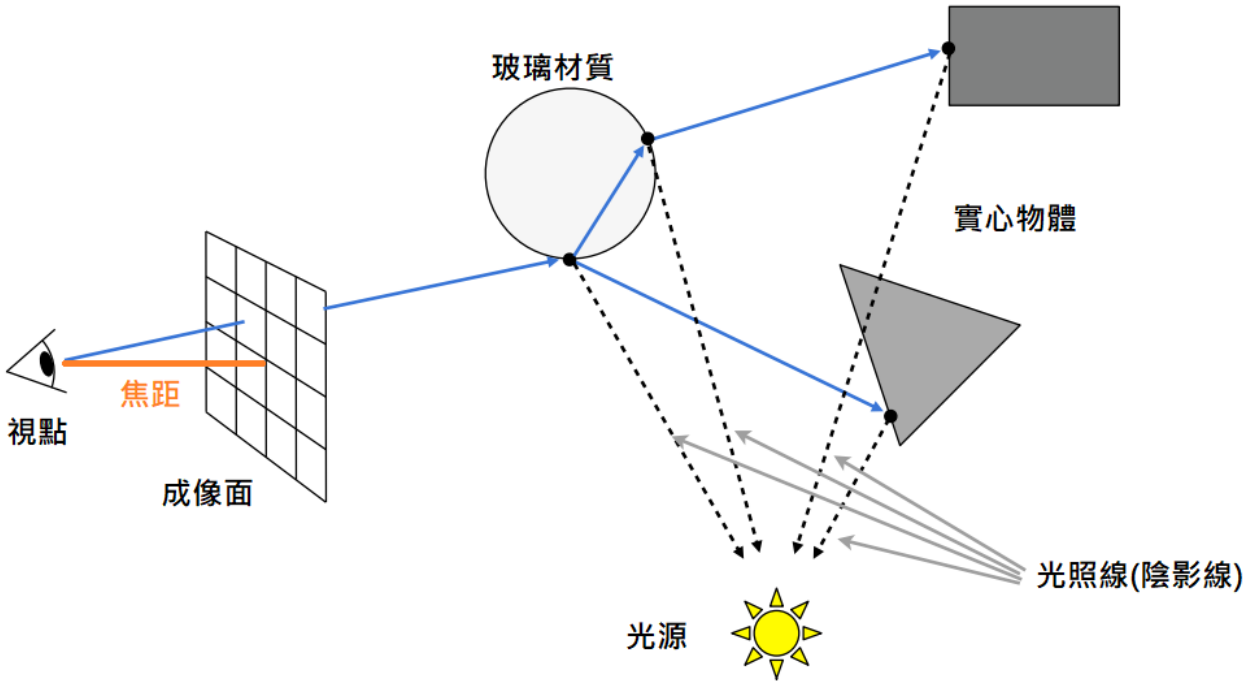


陰影如何產生

光線追蹤：反射與折射

- 透過實際模擬光的行徑路線, 即可正確的繪製各種不同材質表面的物體
- 像是光的反射、折射等等





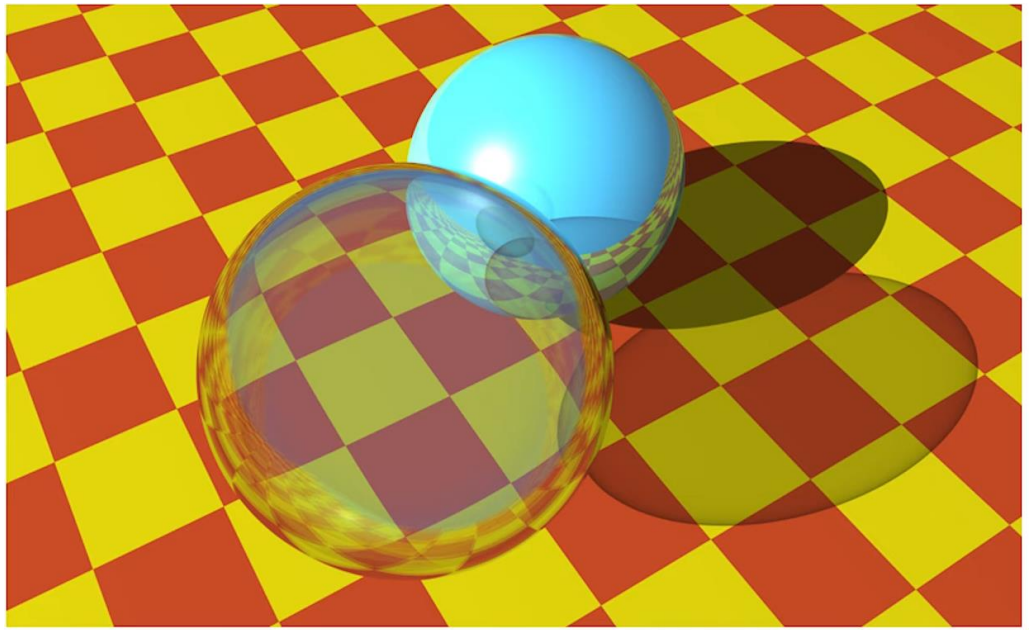
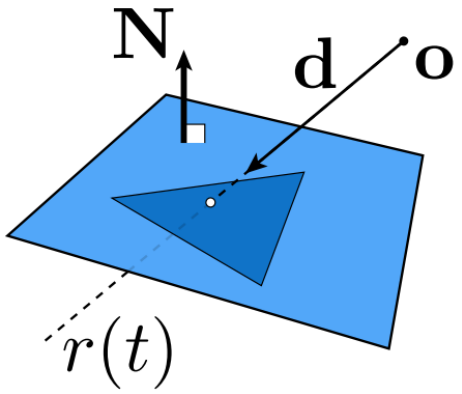


Image now generated in real time in NVIDIA OptiX™ (was 74 minutes per frame in 1980)

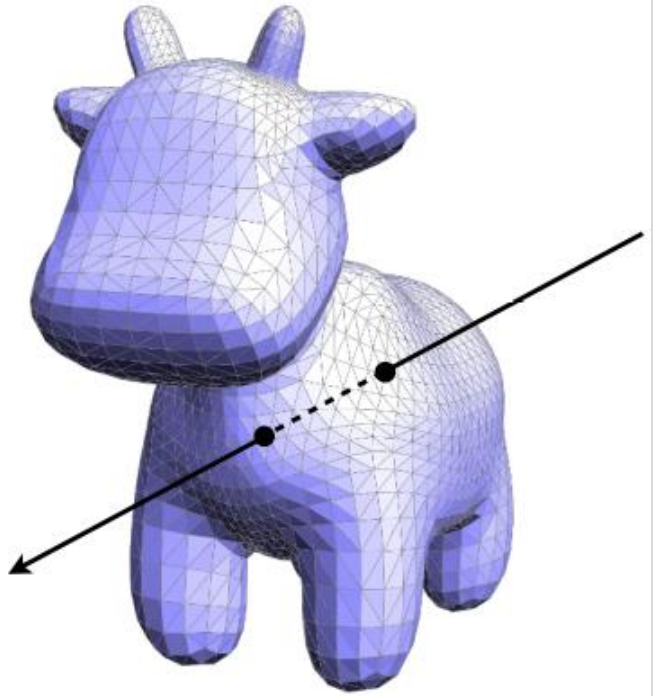
光線追蹤：射線到底打到誰

打出一條光線後:

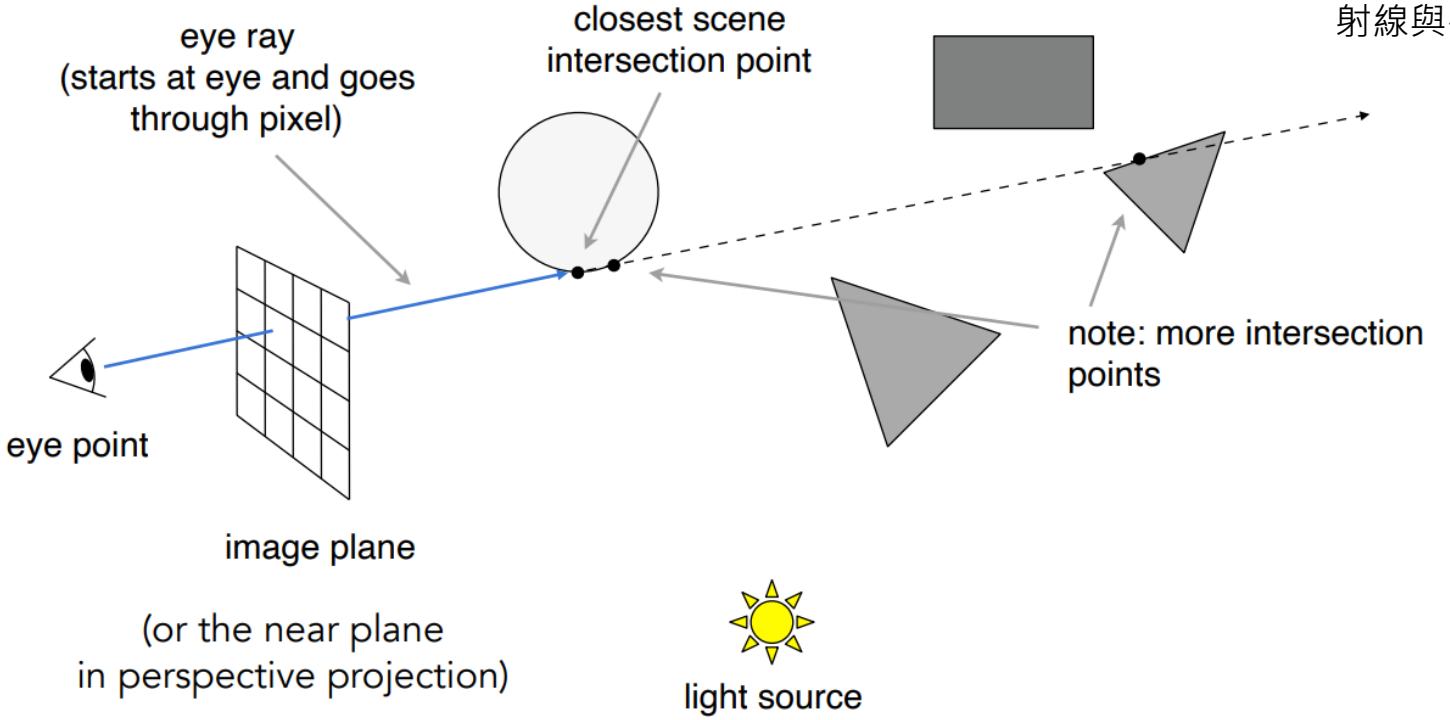
1. 打到哪一個表面
2. 往哪反射與折射
3. 一個物體有幾千甚至幾百萬片三角形,要怎麼快速知道打到哪一片

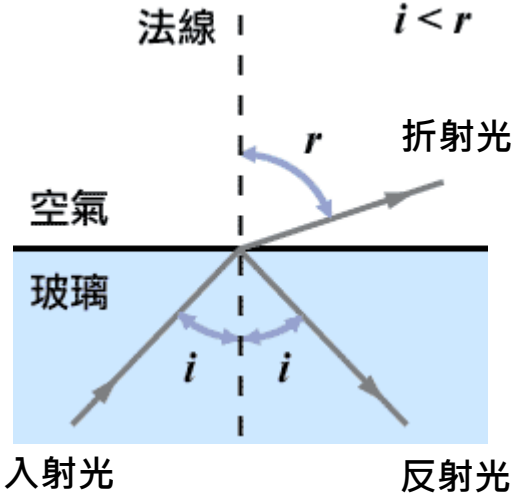


射線與平面求交點



射線打在模型上

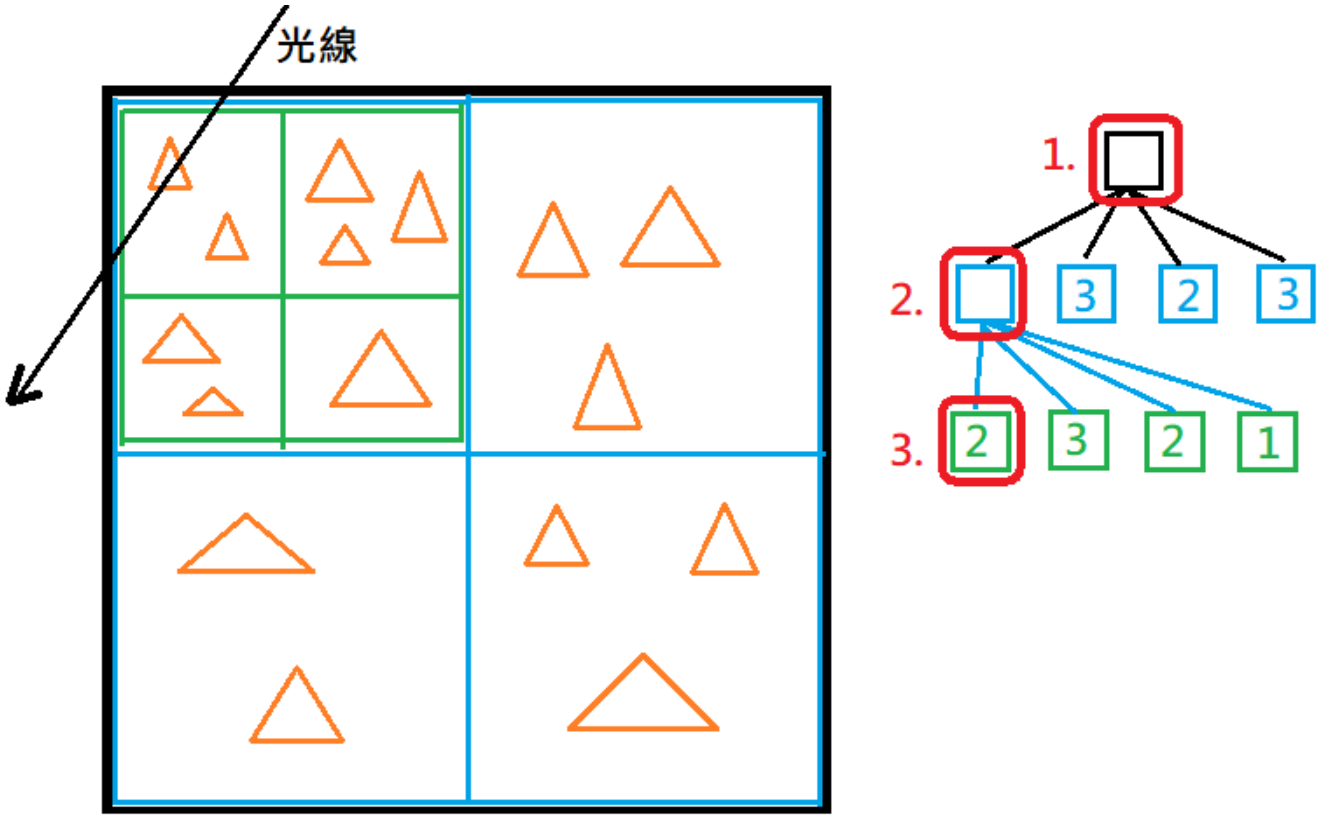


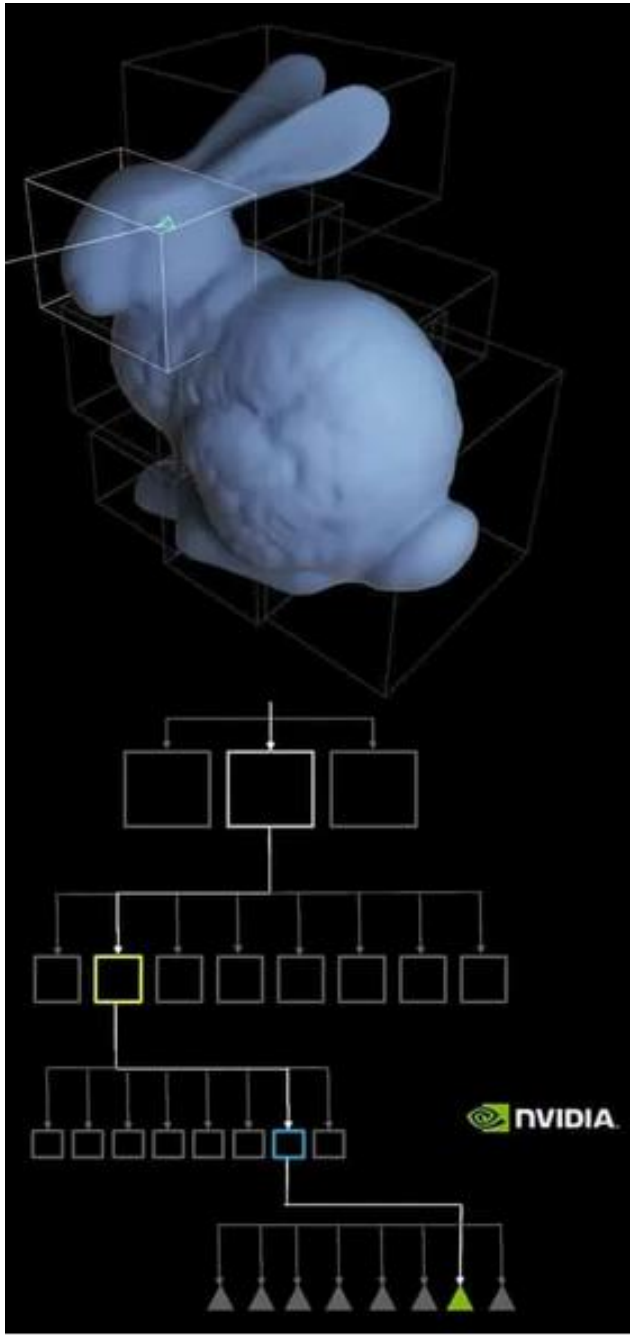


光線追蹤：加速射線檢查

透過空間劃分的方式, 快速排除大量不可能的物體

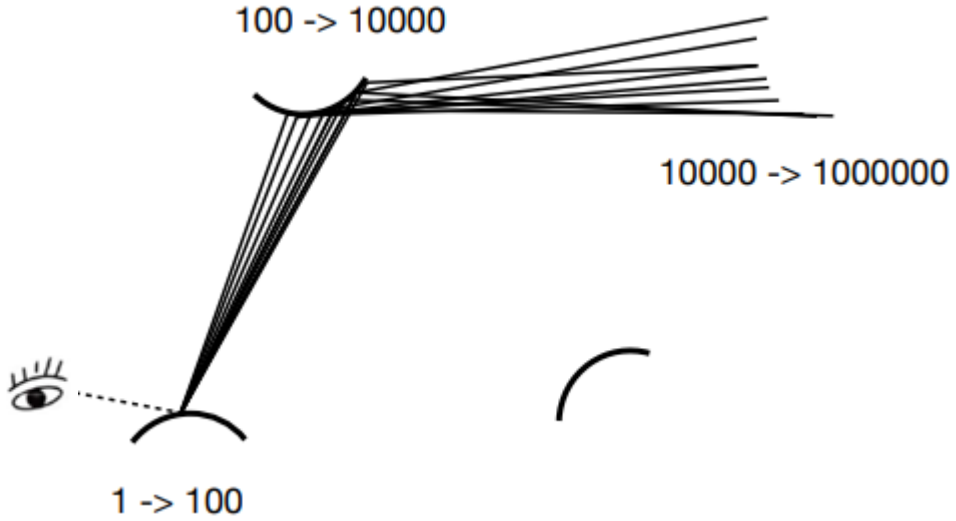
而這就是RTX所做的其中一項硬體優化



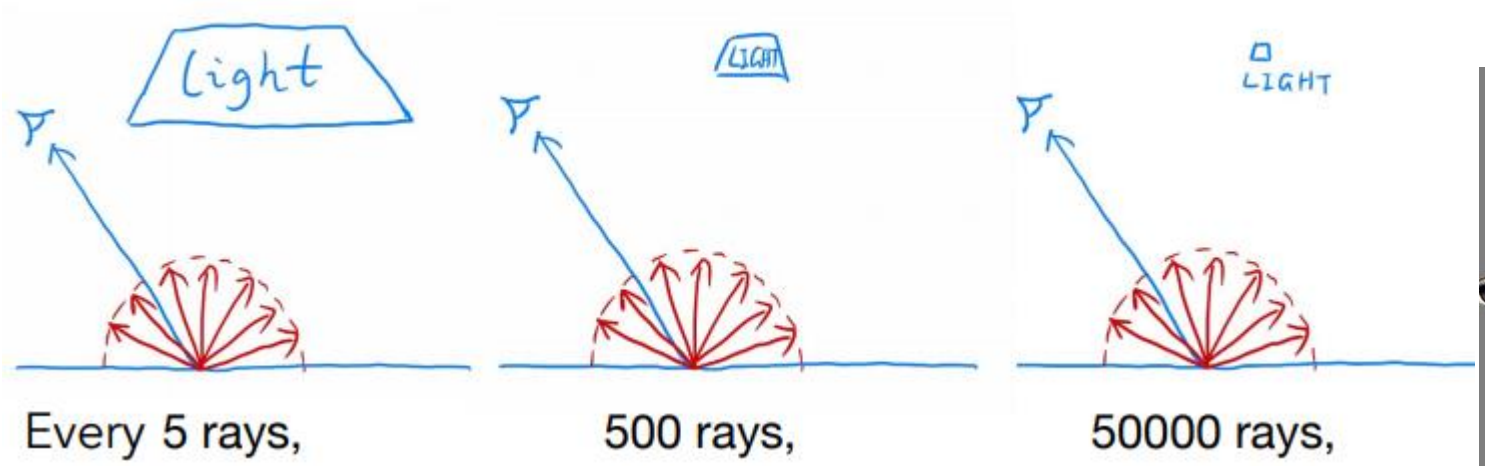


光線追蹤：需要幾條光線

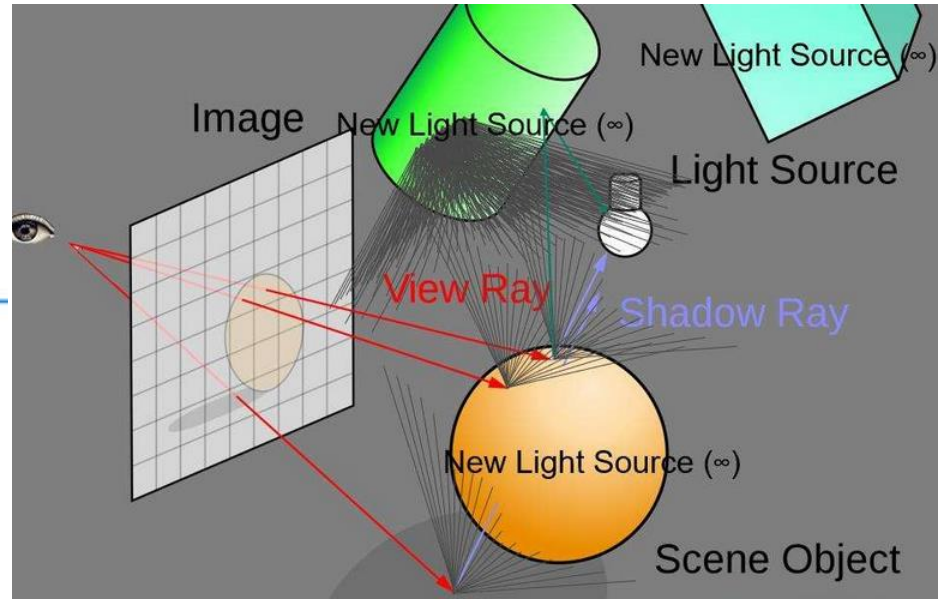
- 反射一條光線就足夠了嗎
- 實際物體受到光後, 表面會反射無數條光線, 如果模擬的射線數量不夠, 就打不到光源, 導致沒有顏色



每個表面模擬100條光線, 僅追蹤3次就變成一百萬條



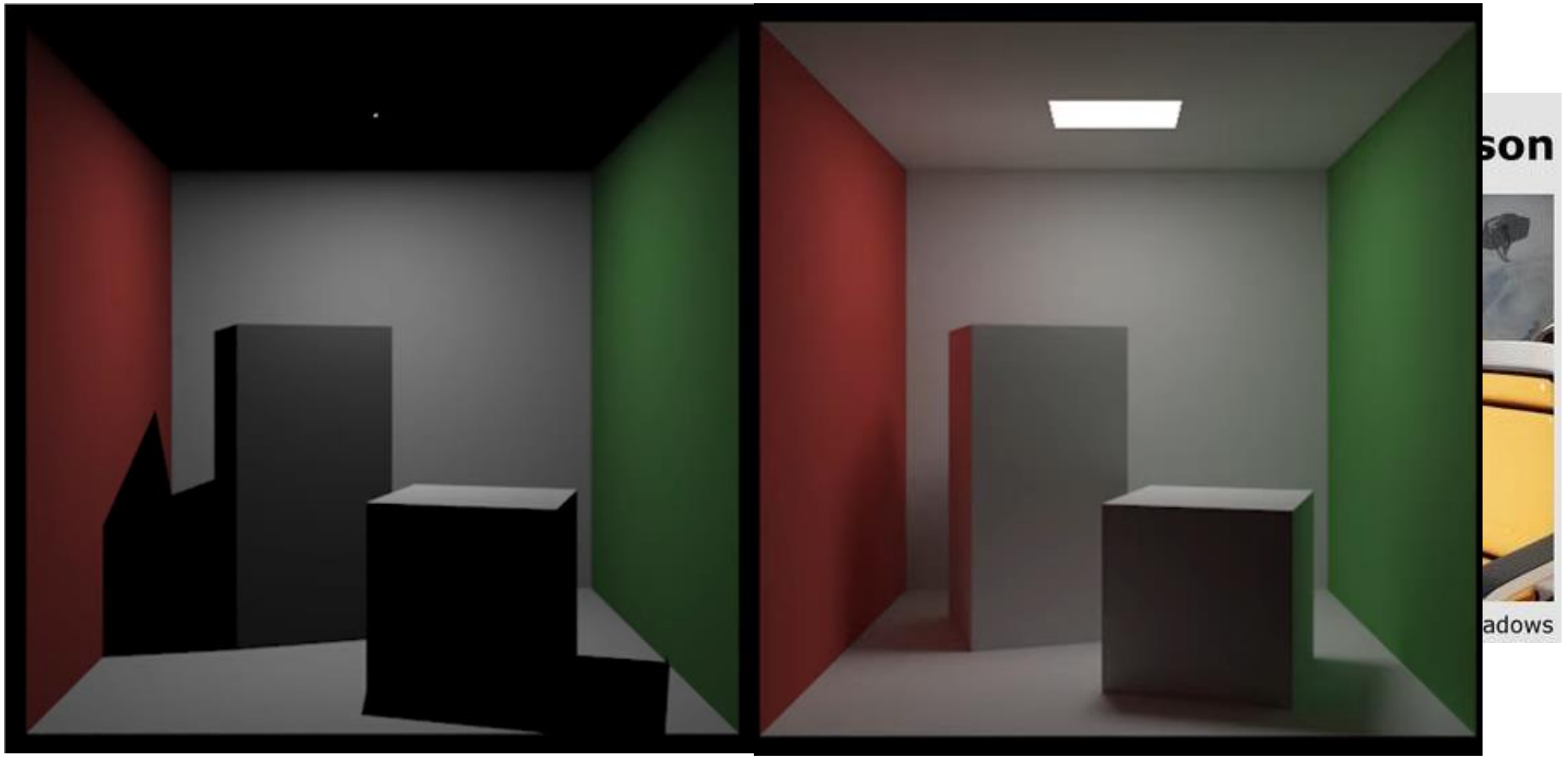
不同光源大小所需要的射線數量



光線追蹤：光線數量的差別

- 當光線數量不足時, 畫面就會出現噪點
- 因為模擬的光線不夠多, 導致理論上該被光照到的區域, 卻沒有被射線打到,導致這個位置的顏色與周圍有落差

光線追蹤：效果比較



光柵化:陰影貼圖

光線追蹤:全局光照

光 線 追 蹤 的 應 用


FULL PRESENTATION
REAL-TIME RAYTRACING
View Ray
Shadow Ray
UNREAL
ENGINE

光線追蹤的應用：即時光線追蹤

光線追蹤很真實, 但沒辦法以60FPS運行遊戲

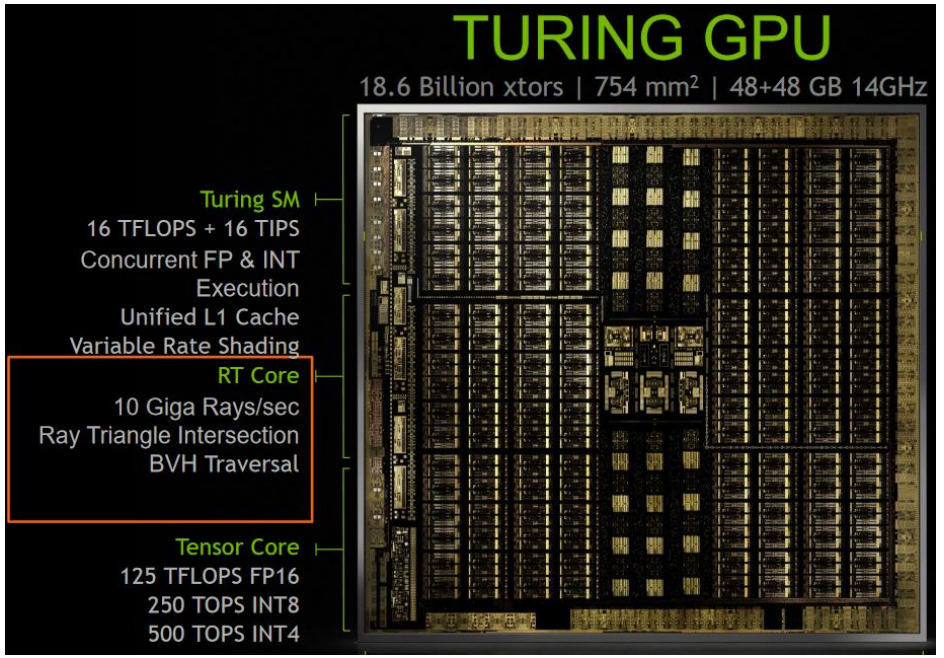
畫面1920x1080, 每個像素1000條光線, 每條反射5次
渲染一張畫面就需要100億次(10Giga)光線運算

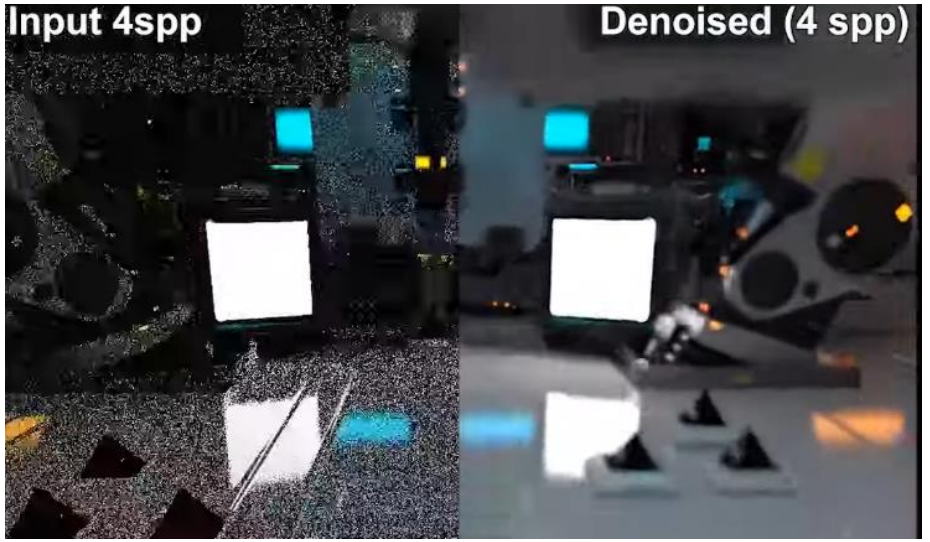
為此RTX顯卡針對光線追蹤做了兩大優化

1. RT Core:射線檢查/空間劃分, 更快找到光線打到的平面

2. Tensor Core : 加速噪點消除, 可以減少模擬的光線量

使效能與畫面品質達到一個平衡





光線追蹤的應用：實際展示

RTX的陰影處理:
https://youtu.be/tjf-1BxpR9c

RTX30系列光追展示:
https://youtu.be/E98hC9e__Xs?t=1163

Real Photo

Simulation

參考資料

GAMES101
https://sites.cs.ucsb.edu/~lingqi/teaching/games101.html

An Overview of the Ray-Tracing Rendering Technique
https://www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-overview/ray-tracing-rendering-technique-overview

https://developer.nvidia.com/

光线追踪-降噪与硬件加速
https://huailiang.github.io/blog/2020/ray2/

光線追蹤如何實現即時3D繪圖？ - 電子工程專輯
https://www.eettaiwan.com/20200810ta31-how-ray-tracing-enabling-real-time-3d-graphics/

深入RTX实时光线追踪技术：原理、接口、算法与应用
https://on-demand.gputechconf.com/gtc-cn/2018/pdf/CH8804.pdf

報　告　到　此　結　束