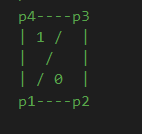
Rendering HW1報告

楊騏瑄

R05922101

1. **heightfield-ray intersection 演算法**

把整個heightfield的1\*1空間做類似grid的分割，但是只有xy方向的而已。接著修改3DDA的方法變成2D的版本，用來確定光線會進入哪個格子，才去做該格子內部的intersect判斷。光線沒有相交到的格子就不用做內部intersect的判斷以節省時間。在格子內部將四邊形切割成兩個三角形如下圖



如果光線從x+跟y-方向進入此格的話就先對1號三角形求intersect，從x-跟y+的方向進入就先求0號三角形。具體判斷進入的方向可以使用NextCrossingT的大小來比較。決定好要先跟哪個三角形求intersect之後仿造traingle::intersect的方法求出有無交點以及differential geometry。當一有交點就可以馬上return true節省時間，否則就要讓光線繼續移動到下一個格子在做該格的內部intersect判斷，直到光線完全走出heightfield都還沒有交點時就return false。

1. **smooth shading 演算法**

在剛開始heightfield的constructor先針對每一個三角形計算他的normal，並且normalize後存到所屬的三個頂點上(用+=的方法)，當全部的三角形計算完後再去對每個頂點normalize(相當於求平均)。在getShadingGeometry傳入該點的座標先去計算他屬於哪個三角形以及三頂點的權重，接著把三個頂點的normal取出並乘與對應的權重即是該點內插的normal。而dndu、dndv是用所屬三角形三頂點的normal相減後除以對應的du、dv長度得到。

1. **與 PBRT default heightfield 在 performance 上的比較**

原本的程式跑landsea-big要10秒，但是heightfield則是跑了23分鐘(皆沒有smooth shading)，推測是intersect的部分出現問題，但是還找不到解決的辦法。而在比較小張的圖像是landsea-0原本的跑了1秒多，heightfield則跑了10秒左右。

1. **所有結果圖**

|  |  |  |
| --- | --- | --- |
| Heightfield | Heightfield(without smooth shading) | Heightfield(with smooth shading) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

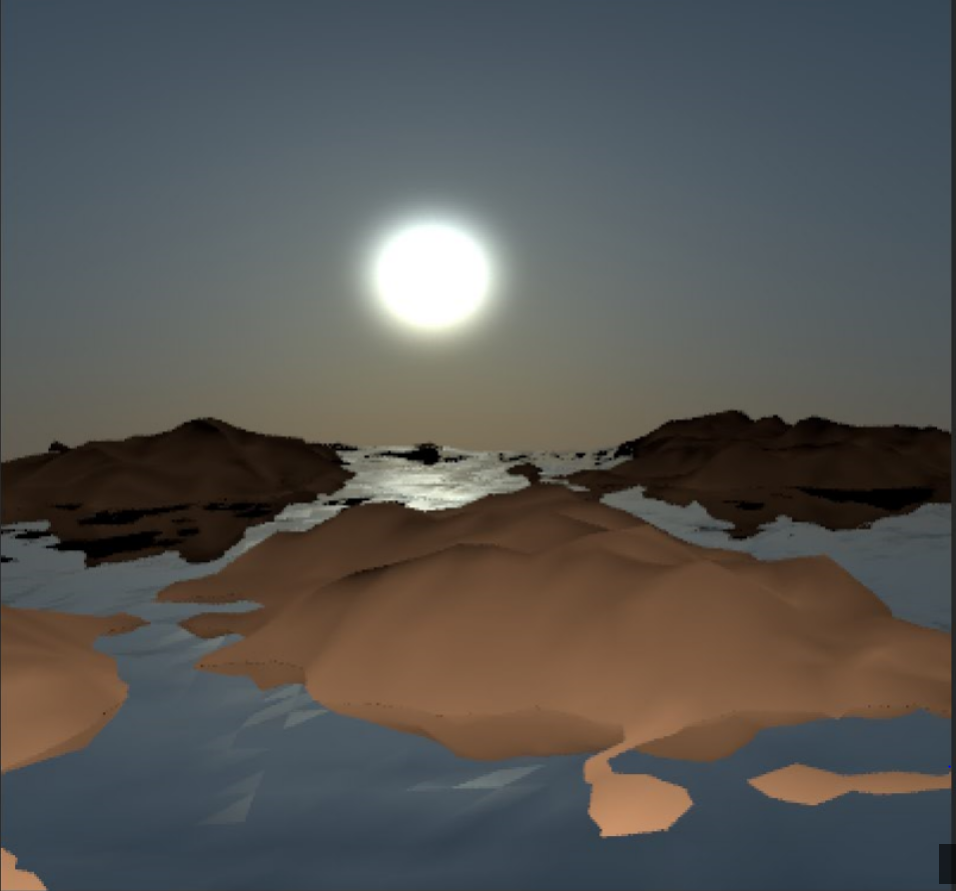
1. **執行環境**

CPU: Intel Xeon E5345(8 cores 2.33GHz)

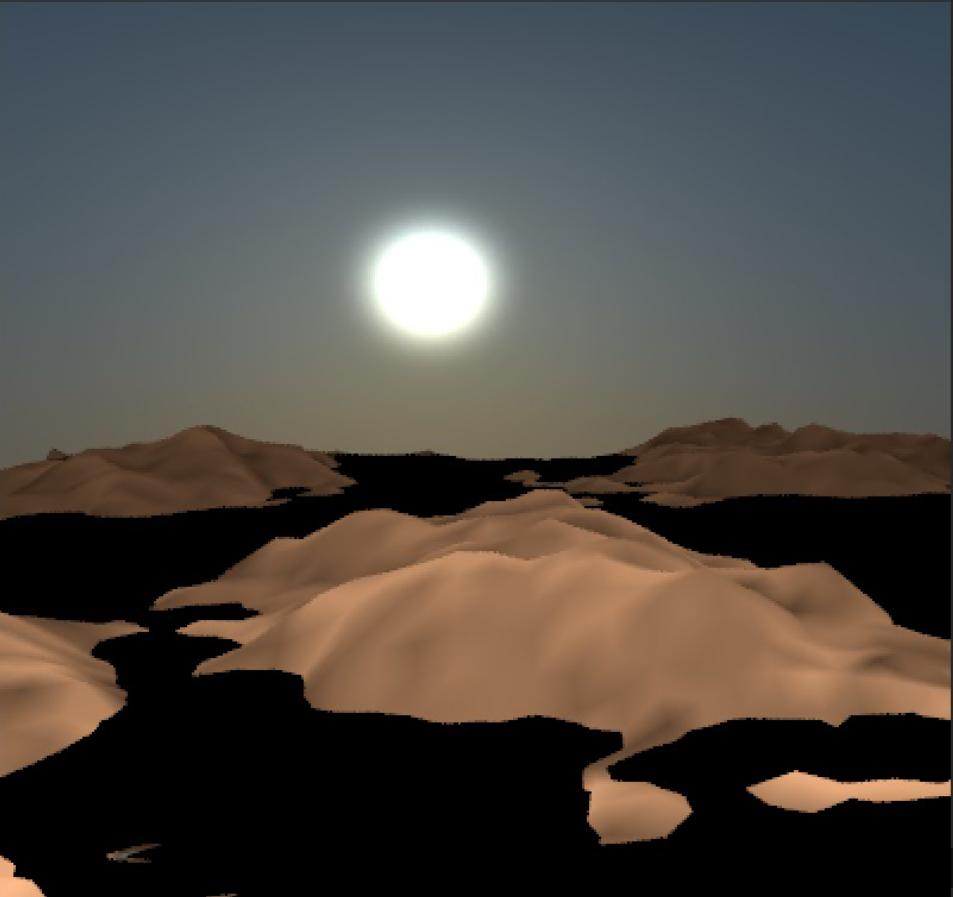
MEM: 16G

OS: linux 64 bit

1. **其他**
   1. 在建構各頂點的normal時把xy的方向正負號搞錯(相當於z反向了)會產生像鏡面一般的效果(如下圖)



* 1. getShadingGeometry如果把ss跟ts多轉換了一次o2w會造成海洋的部分變成黑色，但不清楚原因(如下圖)



* 1. 如果再intersect回傳的dg當中uv沒有在0~1之間的話，會使得texture在貼上的過程中重複用到(如下圖)

