Rendering Project 1 Report

B03902082 資工三 江懿友

A) Heightfield ray-intersection 演算法

首先使用 pbrt Heightfield::Refine() 的方法， 把 heightfield 切成許多三角形，並且把這些三角形存起來。因為這些三角形是 heightfield 切成的，所以不會有退化的三角形（3點共線）。

計算 ray 的交點的時候，則使用 3D DDA 的演算法找出光線碰到的 voxel，並且把那個 voxel 上的三角形拿來判斷相交；因為這些三角形都是 heightfield grid 切成的，所以每個 voxel 上都恰好有兩個三角形。就這樣按照 3D DDA 的順序掃過 voxel 就能找出 ray intersection 或是判斷沒有交點了。

B) Smooth shading 演算法

Heightfield 初始化的時候先算出每個三角形的法向量，並且用每個 vertex 相鄰的面的 normal 加總後當作每個 vertex 的 normal 傳給 TriangleMesh（總之就是先算好每個 vertex 的 normal 並交給 TriangleMesh 存起來）。

GetShadingGeometry() 被呼叫的時候，我們會先算出交點在三角形內的 barycentric coordinate 重心座標，並且使用重心座標當作權重和使用前面存起來的 per vertex normal 內插出交點上的 normal 並且將他 Normalize 成單位向量，然後算出剩下的偏微分後把 DifferentialGeometry 回傳回去。

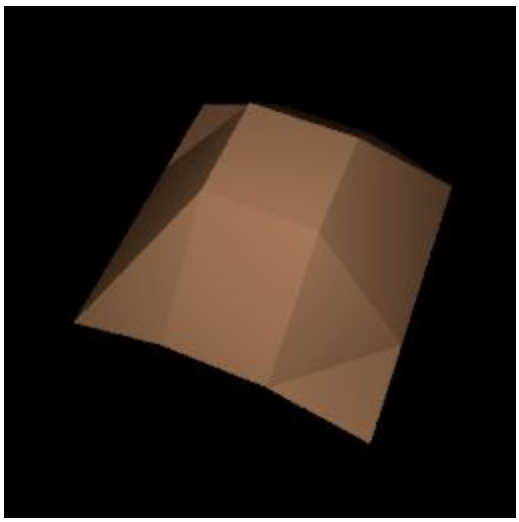
C) 與 pbrt 的 heightfield 的效能比較

	Default	No smoothing	Smoothing
hftest.pbrt	0.0s	0.0s	0.0s
landsea-0.pbrt	0.3s	0.5s	0.5s
landsea-1.pbrt	0.3s	0.5s	0.5s
landsea-2.pbrt	0.2s	0.4s	0.4s
landsea-big.pbrt	0.6s	0.9s	0.9s
texture.pbrt	0.2s	0.3s	0.3s

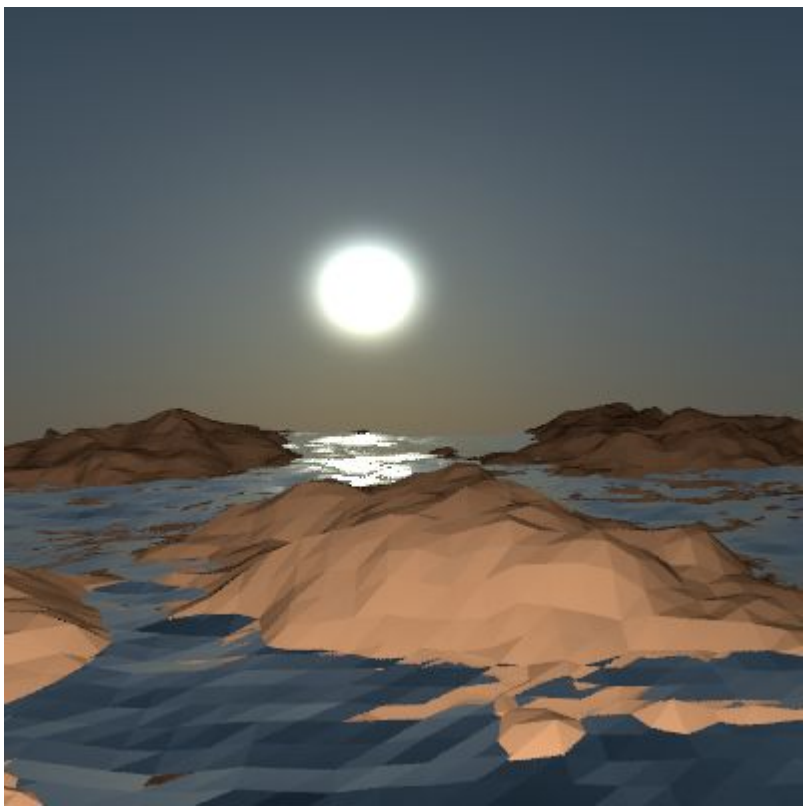
D)所有結果圖

Default:

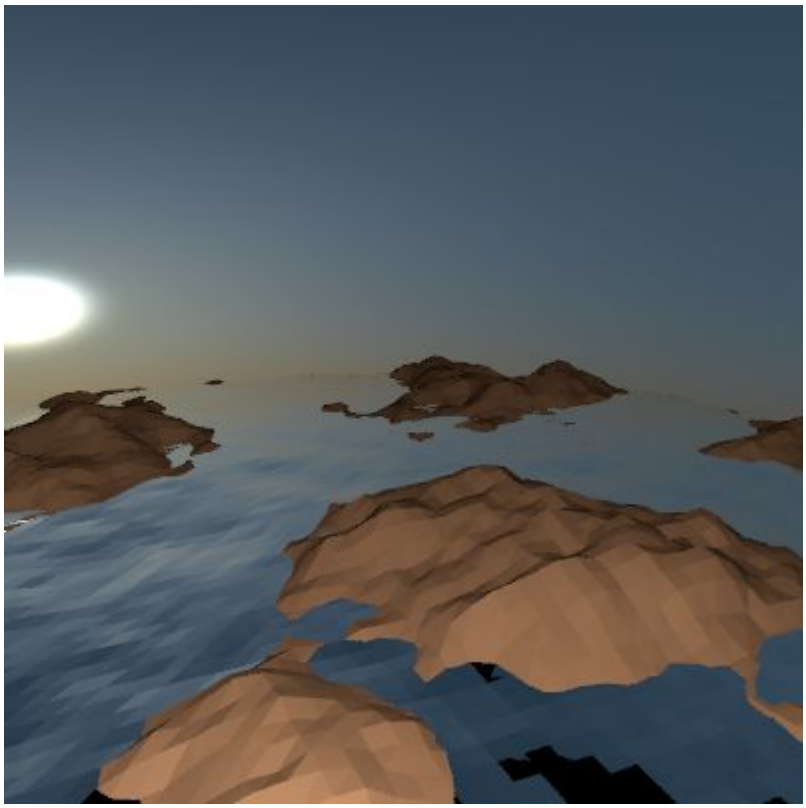
- hftest.pbrt



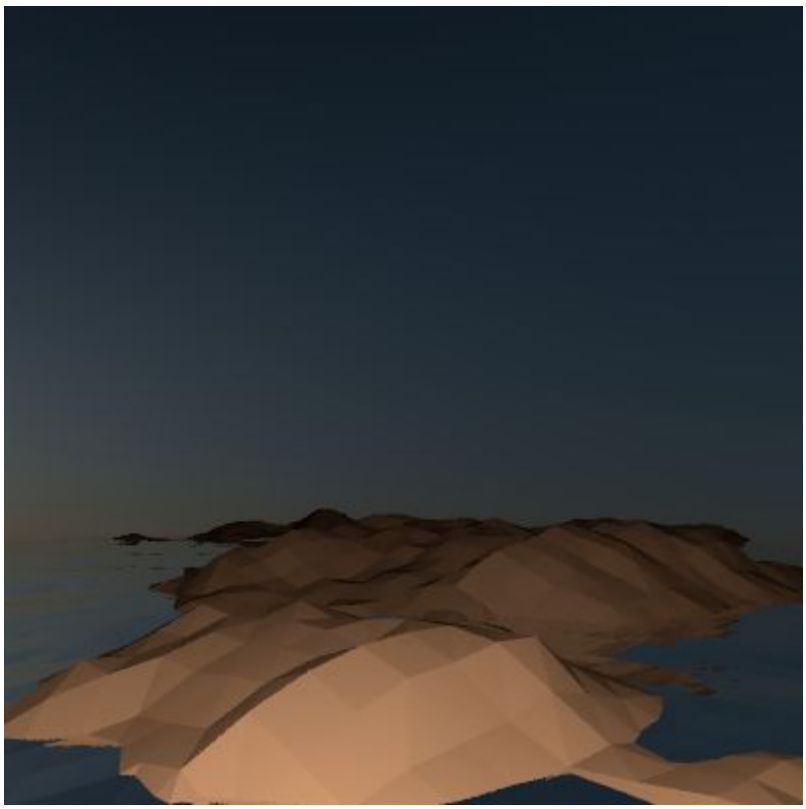
- landsea-0.pbrt



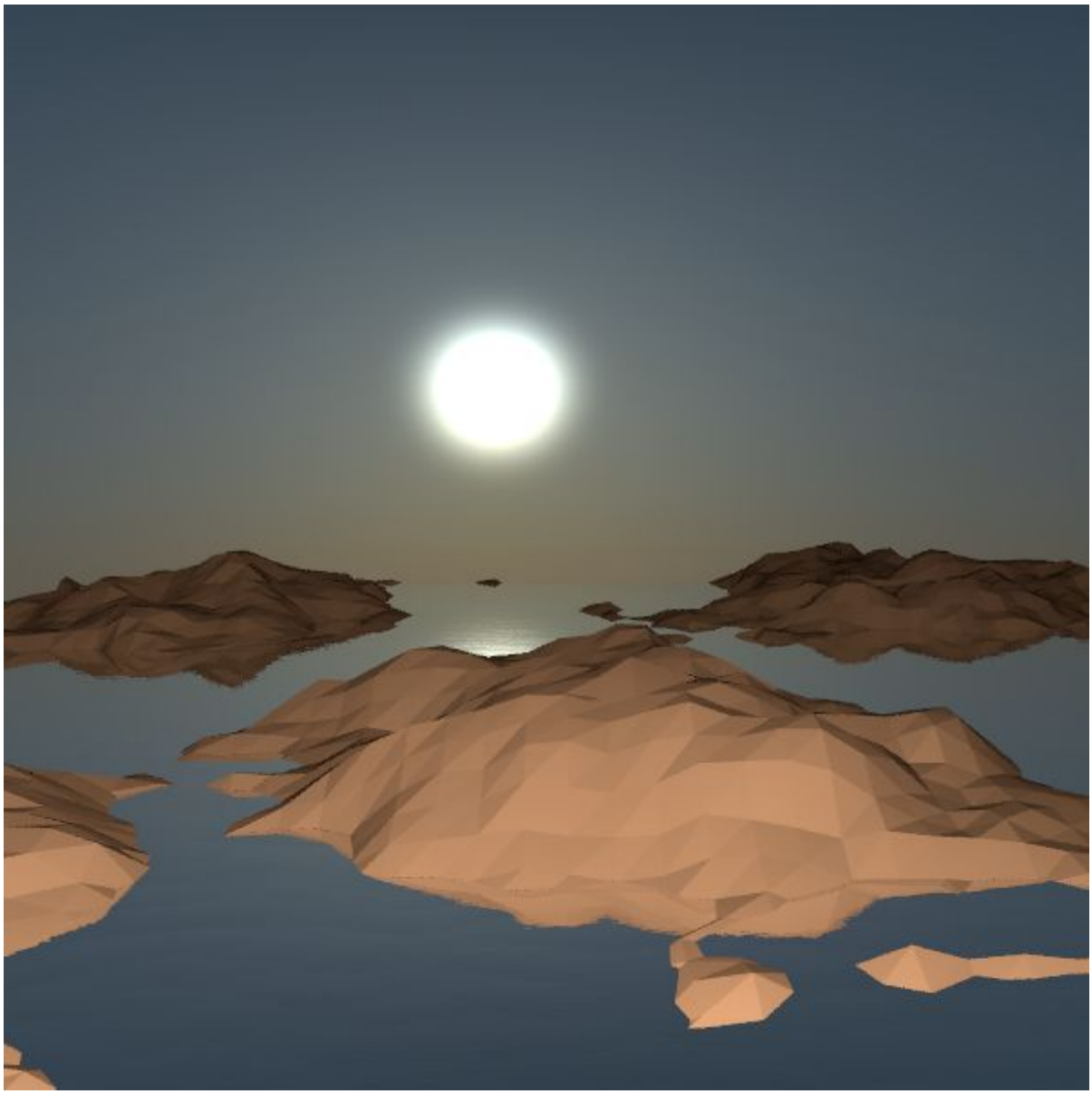
- landsea-1.pbrt



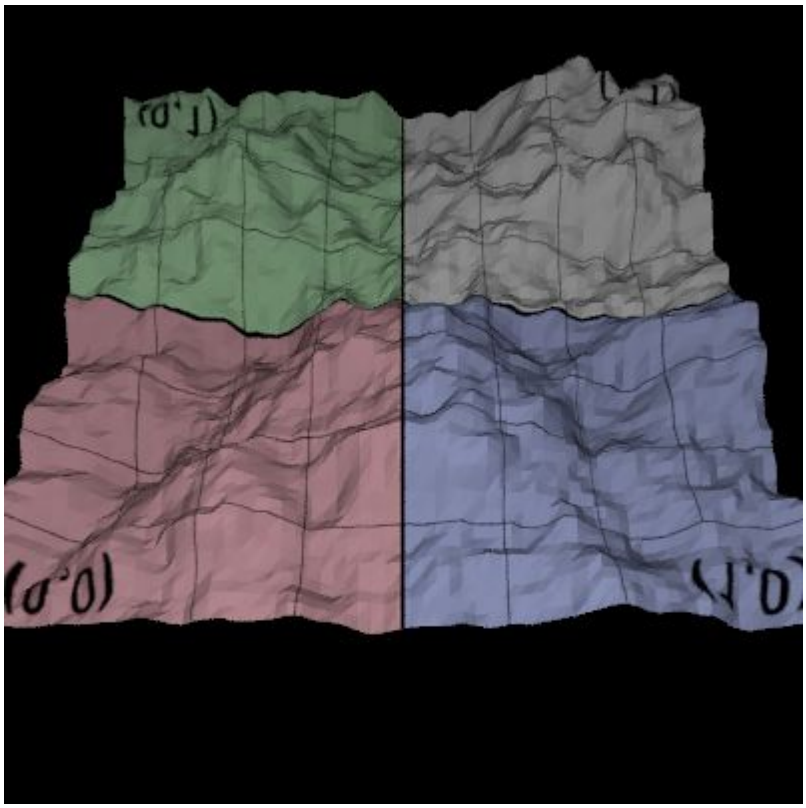
- landsea-2.pbrt



- landsea-big.pbrt

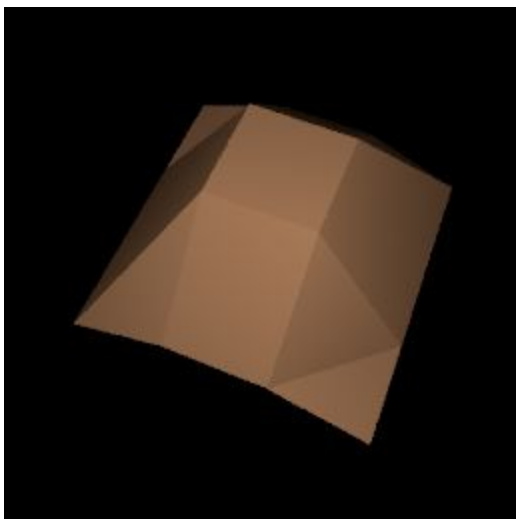


- texture.pbrt

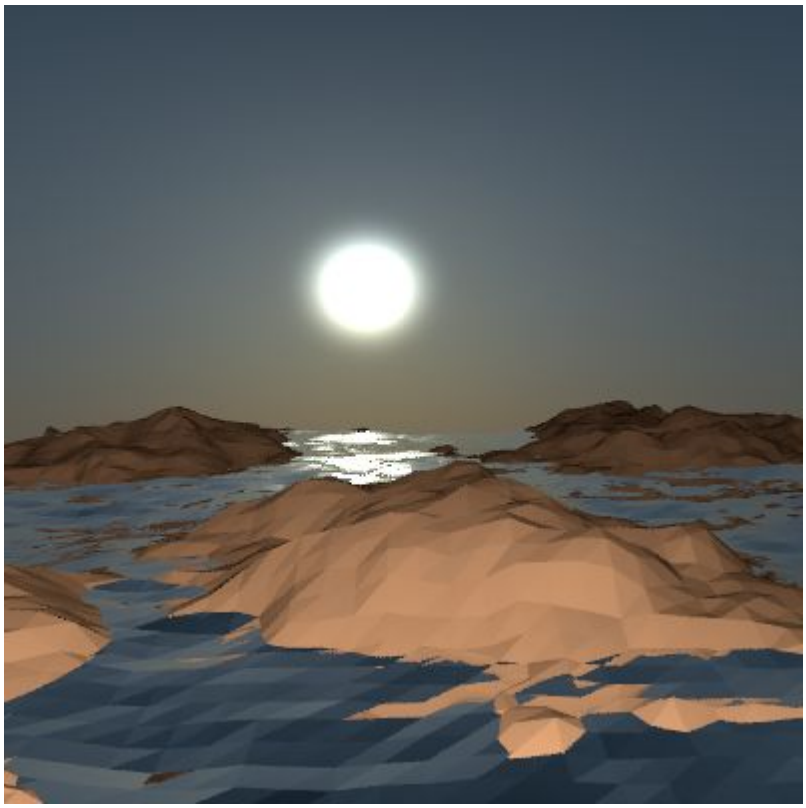


My implementation (No smoothing):

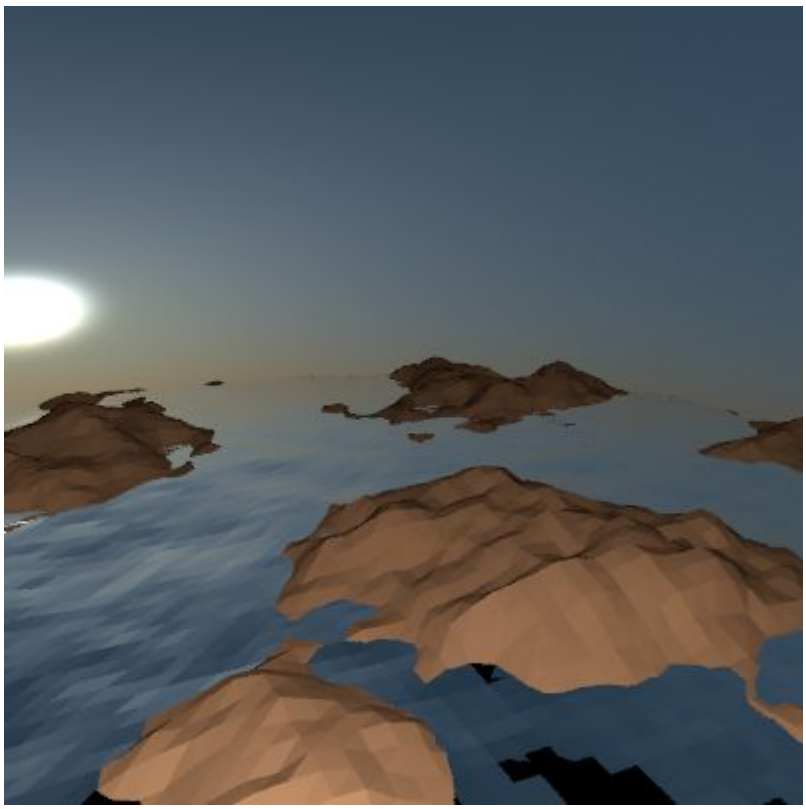
- hftest.pbrt



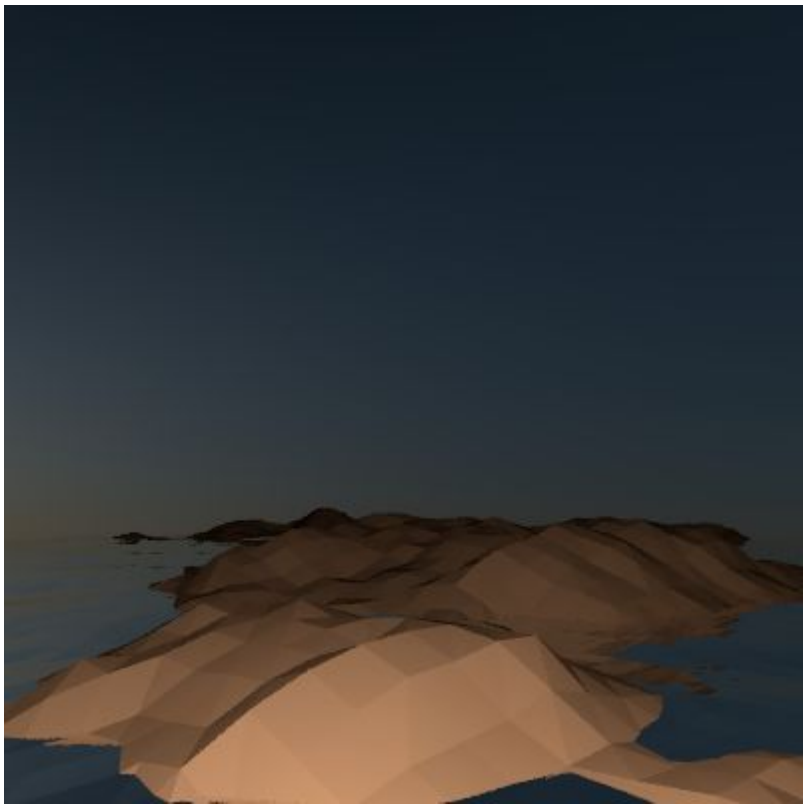
- landsea-0.pbrt



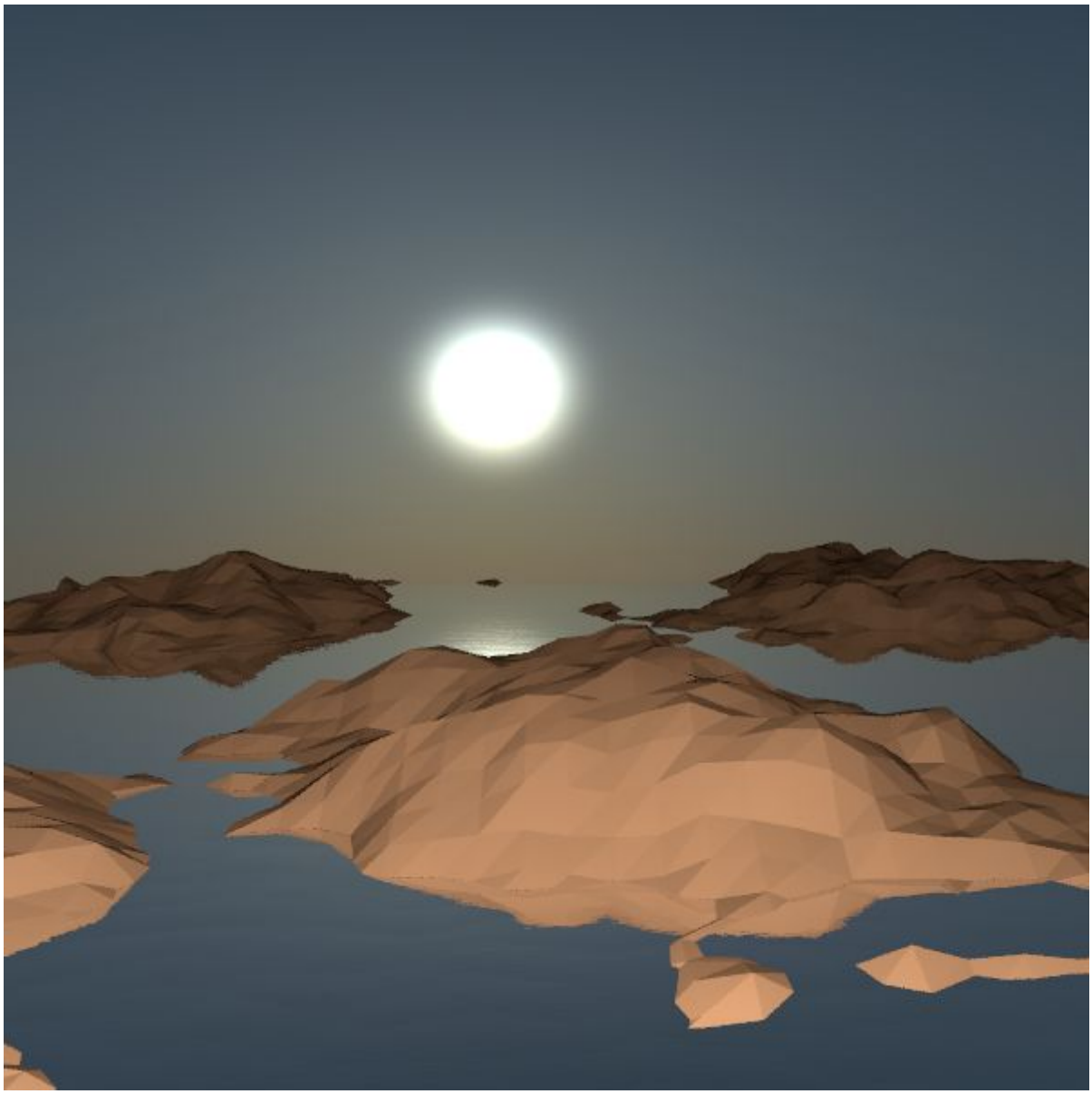
- landsea-1.pbrt



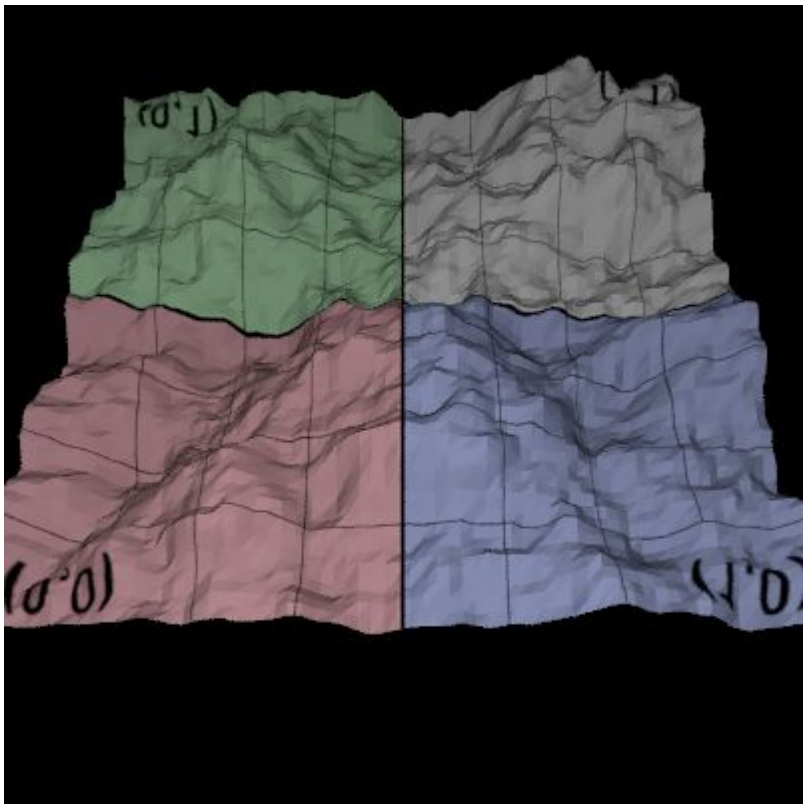
- landsea-2.pbrt



- landsea-big.pbrt

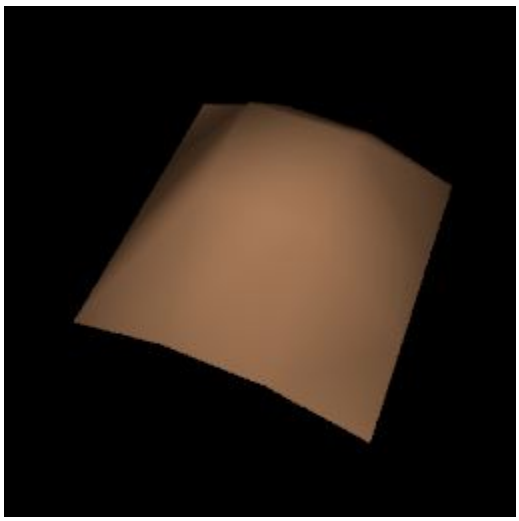


- texture.pbrt

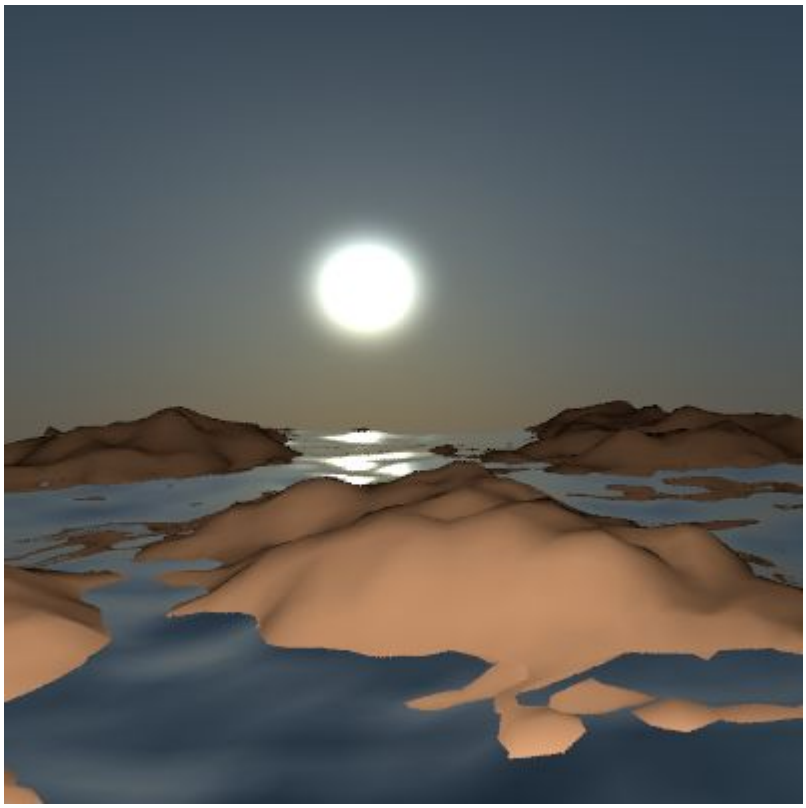


My implementation (With smoothing):

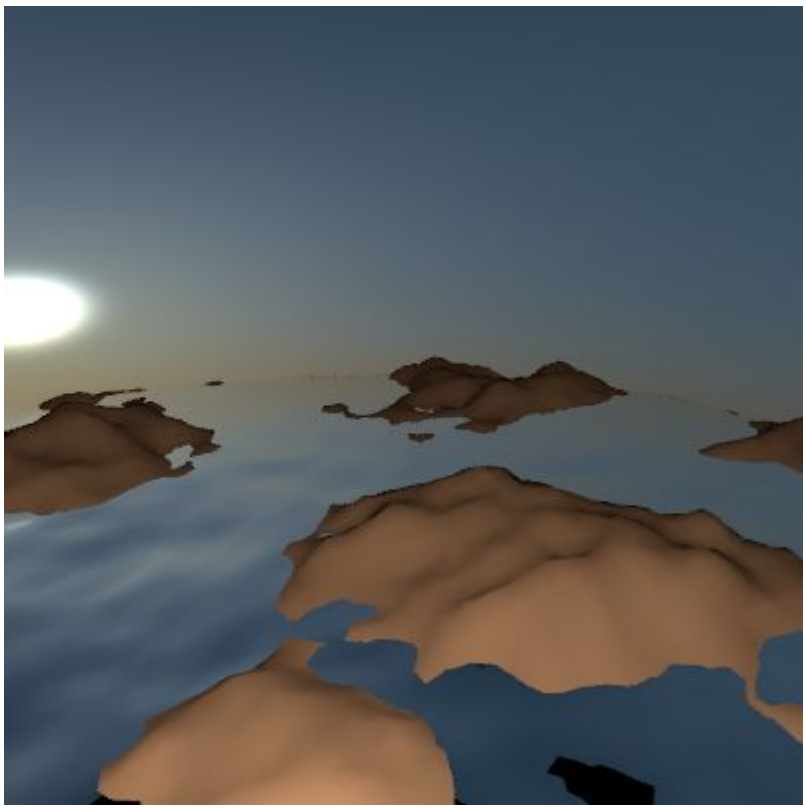
- hftest.pbrt



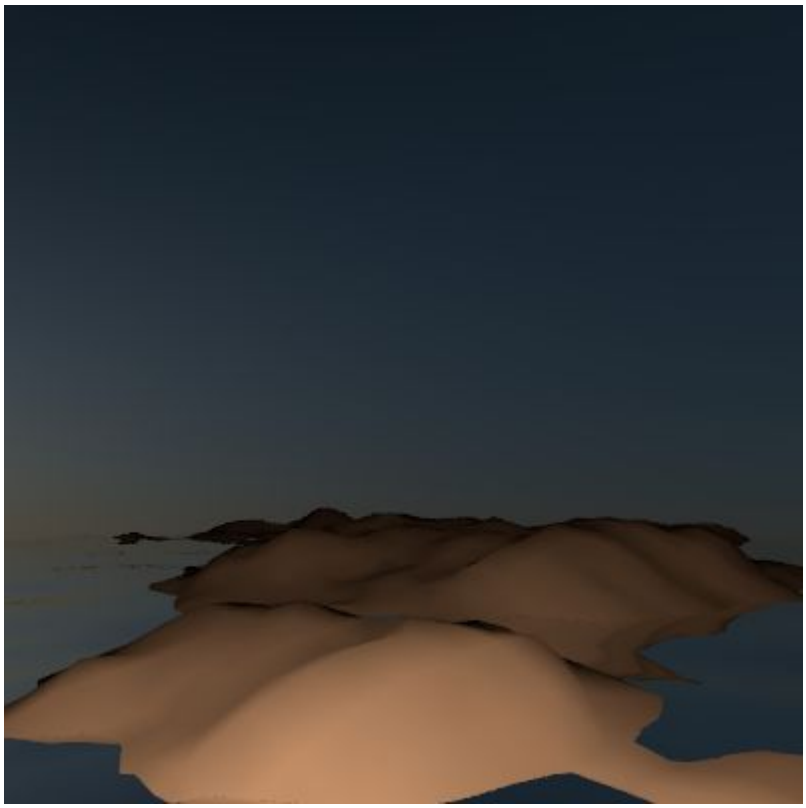
- landsea-0.pbrt



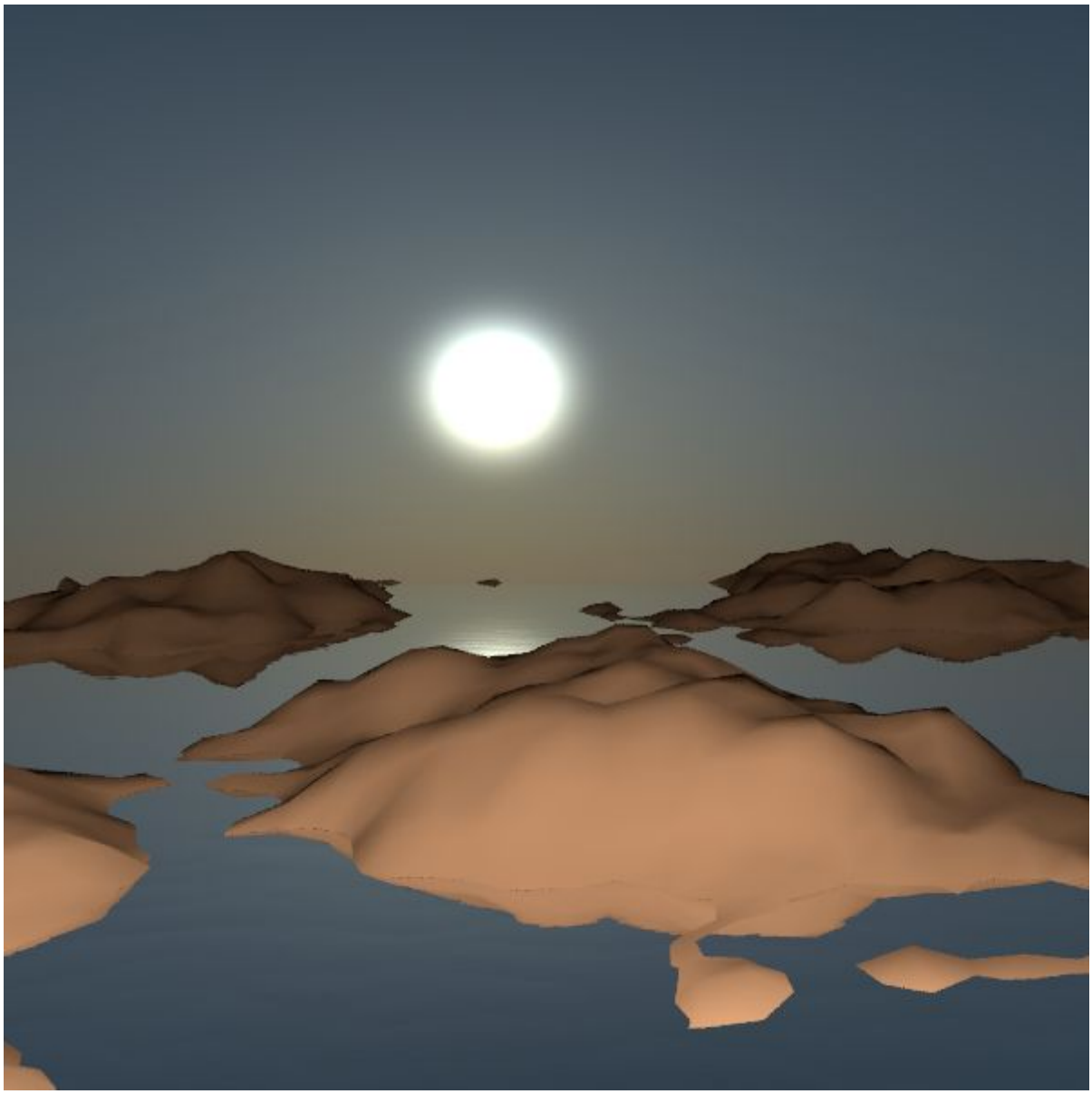
- landsea-1.pbrt



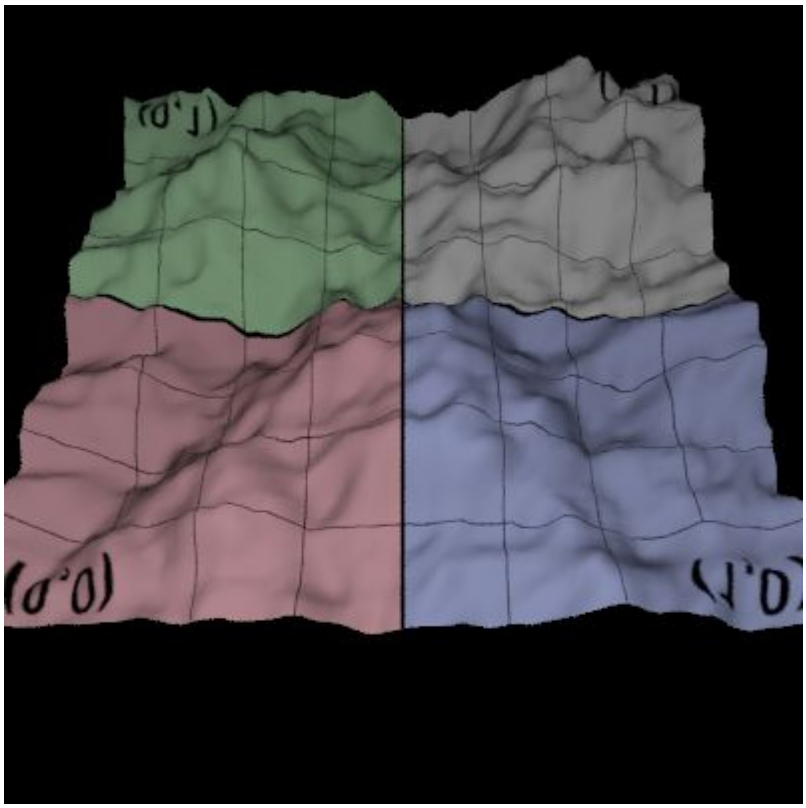
- landsea-2.pbrt



- landsea-big.pbrt



- texture.pbrt



E) 執行環境

OS: Linux
Memory: 16GB
CPU model: Intel(R) Xeon(R) CPU E3-1231 v3 @ 3.40GHz
CPU frequency: 3.40 GHz (Max 3.80 GHz) (Frequency may vary due to turbo boost)
CPU core: 8 cores