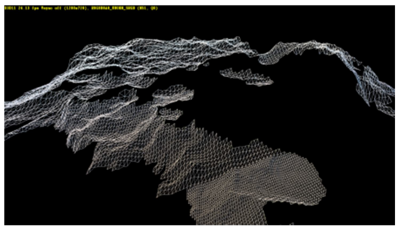
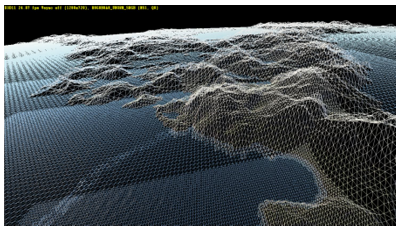
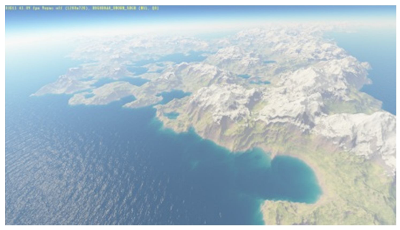
Efficient Terrain & Ocean Rendering for a Real Size Planet

Silviu S. Andrei
Glen Ellyn, USA
silviu.andrei@gmail.com


(a) Refraction map


(b) Terrain & Ocean


(c) Final image

1. Introduction

Full-scale planetary rendering is a broad and active area of research. Its uses range from games to astronomy and simulation applications. Many problems arise when attempting to render both high altitude and close-up scenes of a full-scale planet, especially when the planet has an ocean surface and even more if water surface displacement and refractions in shallow water are required. Some of the most common problems to overcome are: Performance (especially for close-up scenes of shallow water), Z-fighting (between the ocean and the terrain surface in high altitude scenes) and a seamless transition of the shorelines from high to low altitude.
In this work I will present a visibility culling and LOD method which is targeted to solve or minimize all of the above mentioned problems.

2. The method

As a prerequisite I should mention that the terrain in my implementation is maintained in a quad tree structure where each node has 29x29 vertices and a 256x256 terrain height-map. The same nodes are used for rendering both the terrain and the ocean surface which is displaced by a water height-map. For the water refractions, the scene is rendered in a traditional, two pass way. The first pass is used to render the underwater scene in a refraction texture and the second pass is used to render the terrain and ocean surface.

The main problem when rendering the shorelines from high altitude is that they will be defined by the intersection of the ocean surface polygons and the terrain polygons which have a very coarse level of tessellation at high altitudes. Also, the popping effect when the nodes are split is very visible across the shorelines because of the same reason. In order to solve this issue, the ocean is rendered differently when viewed from high altitude. Instead of rendering 2 surfaces, only the terrain surface is rendered, but all vertices that lie below the sea level, are raised to the sea level and all pixels that (according to the terrain height-map) lie below the sea level are rendered using the water material ID. This way, there will be no z-fighting and the shorelines will be defined by the node's height-map, not by the tessellation level of the terrain nodes. Also, a transition from high altitude to ground-view rendering is required for frames which contain both pixels rendered using the space-view technique and pixels rendered using the ground-view technique. This transition is achieved by clipping out space-view pixels that are further away from the camera than a given threshold and clipping in ground-view pixels that are closer than the threshold. The space-view rendering method also ignores ray bending due to refraction, but from high altitude, the difference is negligible and therefore the border between the two rendering methods is not visible.

A significant gain in speed is obtained by culling away geometry that does not need to be rendered. For example, for the refraction map, any geometry below the maximum water visibility depth or above the maximum wave height can be clipped away; for the terrain surface, any geometry below the minimum wave height can be clipped away and for the ocean surface, any geometry above the maximum wave height can also be clipped away.
These geometries can be culled in 2 stages. The first stage rejects entire nodes on the CPU side based on the minimum and maximum vertex height of the node and the second stage rejects entire triangles in the geometry shader based on the triangle's vertices.
To accelerate rendering even more, the terrain and ocean nodes are rendered using the same draw call. The geometry shader will later decide if it needs to emit a triangle for the ocean surface, terrain surface, both or let the pixel shader decide based on the node's terrain height-map. When the geometry shader emits a triangle it attaches a material ID to the vertex output structure to let the pixel shader know which rendering method to use.

Pseudo-code of the geometry shader:

```
if (triangle max distance to camera > distance to camera threshold)
{
  Set MaterialID to "unspecified" for all 3 vertices;
  Set each vertex height to sea level if it is below sea level;
  Emit triangle;
}
if (triangle min distance to camera < distance to camera threshold + delta)
{
  if (triangle max height > min wave height)
  {
    Set MaterialID to "terrain" for all 3 vertices;
    Emit triangle;
  }
  if (triangle min height < max wave height)
  {
    Set each vertex height to: sea level + water height-map value;
    Set MaterialID to "water" for all 3 vertices;
    Emit triangle;
  }
}
```

delta = number chosen to create a small overlap between the high altitude rendering and the low altitude rendering method

SIGGRAPH 2012, Los Angeles, California, August 5 – 9, 2012.
ISBN 978-1-4503-1435-0/12/0008