Jump to...

project #2
project #3
final project
assignments

project #1: Height field

Assigned: 2016/10/6
Due: 2016/10/27 2:00pm

Project description

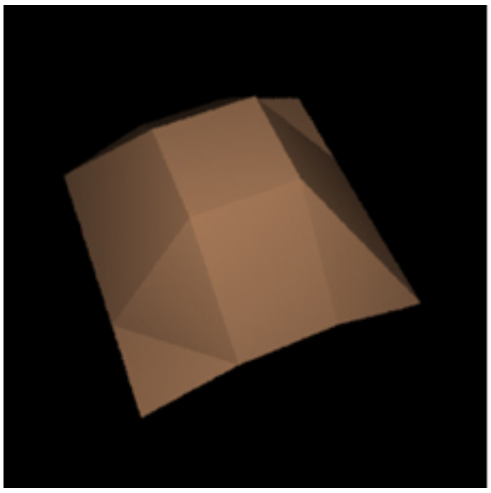
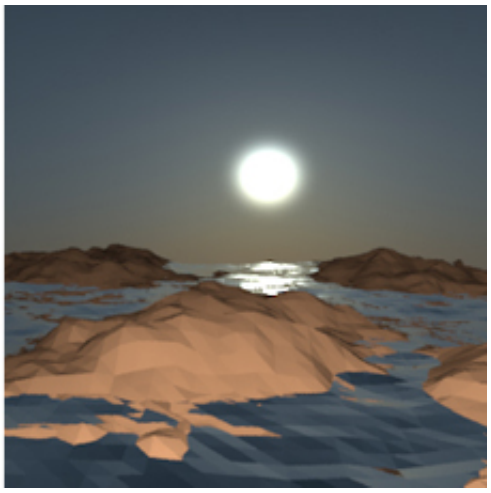
In this assignment, you are asked to improve pbrt's heightfield class. A heightfield is a 2D array which samples a height function z(x, y). It indicates how high the sampled point (x, y) is and can be used for modeling things such as terrain.

pbrt has provided an implementation of heightfields which converts a heightfield into a triangle mesh. It consumes more storage and could be less efficient. In this assignment, you are asked to provide an alternative heightfield class which does not convert into triangles. You can employ the DDA method used in uniform grid for the intersection routine for heightfields.

This project is based on the similar ones from Stanford and Virginia.

Step 1: Download test scenes

Download this file containing scenes, heightfield data and textures. Run pbrt on scenes, hftest.pbrt and landsea-0.pbrt, and you should see images that look like the following:

The 4x4 heightfield in hftest.pbrt (left image) is designed for debugging. The landsea-0.pbrt contains two 64x64 heightfields (land and sea) and there are several views of this scene (0-3). landsea-big.pbrt contains a 1000x1000 heightfield and will likely take a while to render and consume loads of memory in the process.

Step 2: Add a new heightfield class

To add a shape (or a class in general) into pbrt, you need to perform the following steps.

The following example assumes that you are adding a class called Heightfield2.

1. Add the source files heightfield2.cpp and hightfield2.h into the shapes/ directory. You can copy them from heightfield.cpp and heightfield.h and modify them. Change the class name from Heightfield to Heightfield2. In addition, change the CreateHeightfieldShape function to CreateHeightfield2Shape accordingly.
2. Add the following lines to the file, api.cpp, into proper locations.

   ```
   #include shapes/heightfield2.h
   ```

   and (around line #339)

   ```
   else if (name == "heightfield2")
     s = CreateHeightfield2Shape(...);
   ```

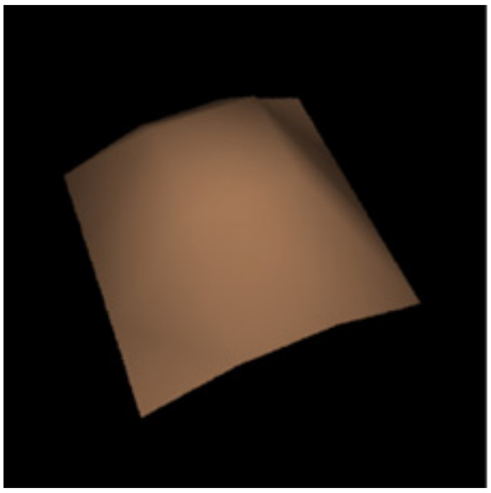
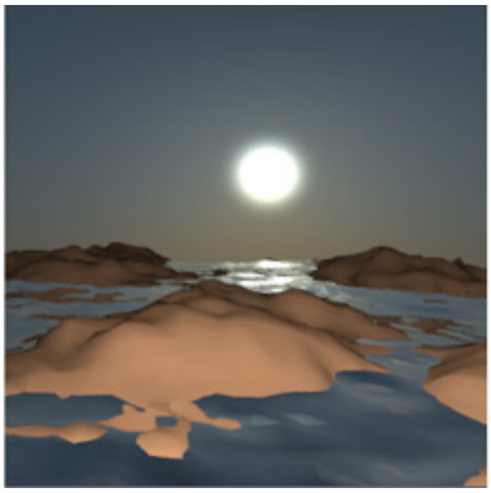
3. Change CanIntersect() to return true so that pbrt will not call Refine to refine it into a triangle mesh. Implement Interset() and IntersectP() functions to directly intersect a ray and the heightfield. Your implementation of these routines will likely use an acceleration structure like 2D uniform gird or quad-tree.
4. Rebuild. To invoke heightfield2, in your pbrt scene file, change Shape "heightfield" to Shape "heightfield2"

Step 3: Add normal and texture information

Add the following features:

- Compute the u and v coordinates so that your can texture map the heightfield. You can test this by rendering the texture.pbrt scene which maps the land heightfield with a colored grid texture.
- Perform Phong interpolation of normals across each heightfield triangle.

You need to understand the differential geometry structure and the GetShadingGeometry() function to return and store the above information. With these steps, you should render into the following images

Note

TA has provided a new interface file api.cpp so that it is easier to link and test your submitted project. Please read this note for the instruction of downloading the file and submitting your project.

Submission

You have to turn in the source code for your class and a very brief report in html format. The report should describe what acceleration algorithm you have implemented for intersection computation. Report the times spent for rendering landsea-big.pbrt with pbrt's heightfield and your own heightfield. Please be as fair as possible when measuring the execution time for both. For example, you should use the same machine and compilation setting for both. Please read this note for submission instruction.