CMSC 23700 **Introduction to Computer Graphics** **Project 4**
Winter 2008 **February 21**

Precomputing a shadow texture

Due: Wednesday, February 27 at 3pm

1 Summary

The final project involves rendering outdoor scenes. In the first part of this project, you will load a heightfield that defines a terrain and will precompute shadow information for the terrain using ray casting. For this project, you can work in groups of two (except for Ph.D. students taking the course for systems credit).

2 Description

Shadows are one of the most important visual cues for understanding the relationship of objects in a 3D scene. As discussed in class, there are a number of techniques that can be used to render shadows using OpenGL. For this project, we will use shadow maps.

3 Heightfields

Heightfields are a special case of mesh data structure, in which only one number, the height, is stored per vertex. The other two coordinates are implicit in the grid position. If s_h is the horizontal scale, s_v the vertical scale, and $\mathbf{H}$ a height field, then the 3D coordinate of the vertex in row i and column j is $\langle s_h j, s_v \mathbf{H}_{i,j}, s_h i \rangle$ (assuming that the upper-left corner of the heightfield has X and Z coordinates of 0). By convention, the top of the heightfield is north; thus, assuming a right-handed coordinate system, the positive X-axis points east, the positive Y axis points up, and the positive Z-axis points south. The heightfield is typically represented as a linear array of height samples, with the i, j element at index $iw + j$, where w is the width of the heightfield. Because of their regular structure, heightfields are trivial to triangulate; for example, Figure 1 gives two possible triangulations of a 5×5 grid. For this project, we will use the ROAM algorithm, which uses the triangulation shown on the left of Figure 1. Note that the direction of the split for a triangle can be determined by the sum of the row and column indices of the upper-left corner. If the sum is even, then the split runs from the NW corner to the SE corner, and if the sum is odd, the split runs from the SW corner to the NE corner.

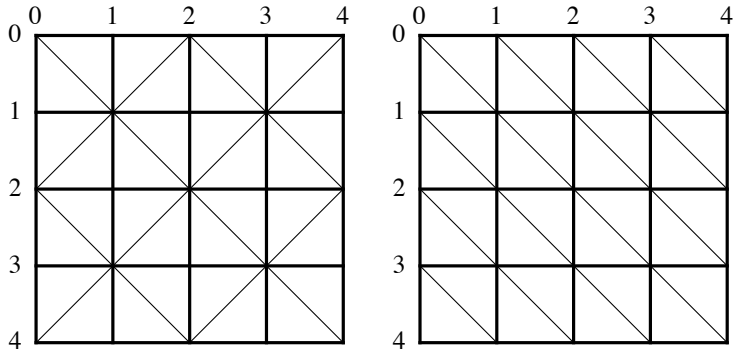

Figure 1: Heightfield triangulations

4 Shadows

The map format includes the direction of the sun (specifically, the vector for a directional light). In this project, you will precompute a shadowmap from the heightfield that can be blended with the surface texture to result in a shadowed landscape. For each grid cell in the heightfield, you should compute a 2×2 texture sample (*i.e.*, the shadow texture will have size $2^{n+1} \times 2^{n+1}$ for a 2^n wide heightfield. Use one byte per texel, with 0 meaning shadowed and 255 meaning lit. You can determine if a texel is in shadow by casting a ray from its center back towards the direction of the light (see Figure 2). Using this information, you precompute the lighting information and

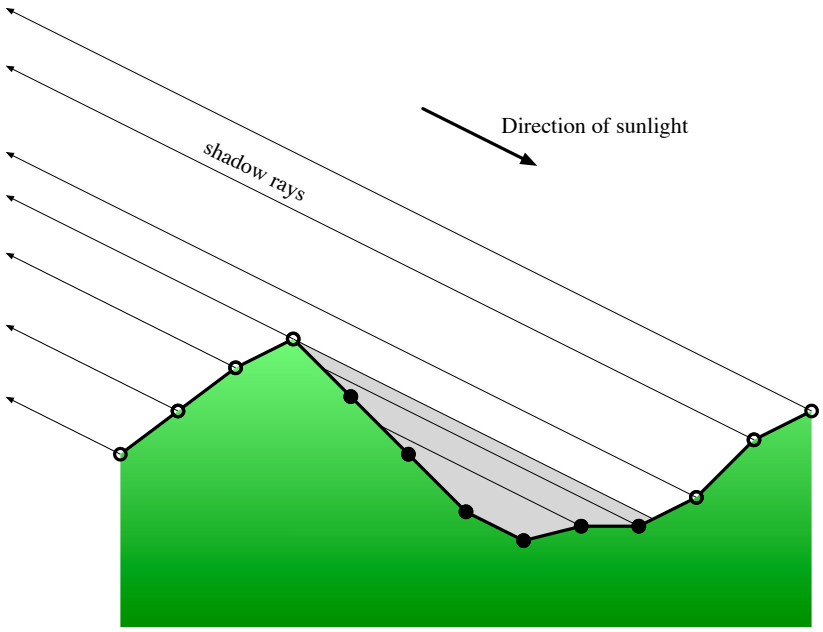

Figure 2: Computing shadows by ray casting

luminance map for the terrain. A better result can be obtained by computing several samples per

texel and averaging their values.

There are various aspects of the geometry that you may be able to exploit to improve the time it takes to compute the shadows. For example, shadow rays will always hit a back-facing triangle before they hit a front-facing one.

One issue that you will have to address is the maximum texture size supported by the hardware. To avoid problems with this limit, we will tile the textures in 1024×1024 chunks (*i.e.*, each tile covers 256×256 grid cells.

5 User interface

For testing purposes, you should render either the heightmap as a textured image, the shadow map as a grayscale image, or a blending of the two. We will supply a one-dimensional texture for texturing the heightmap; you should use the terrain's height (Y coordinate) as the texture index. You should render these images using orthographic projection. The user interface will allow one to switch between the rendering modes. The sample code supports the following keyboard commands:

`h`	render the heightmap as a textured value
`s`	render the shadow texture as a grayscale value
`b`	render the heightmap as a textured value with shadows
`q`	quit the viewer

6 Input format

A terrain data set is represented as a directory containing various files that define the scene to be rendered. For this part of the project, you will need the following files:

- `map` — this file contains information about the terrain data set, such as scale, feature locations, and the direction of the sun.
- `hf.pgm` – this file contains the height-field data.

Later stages of the project will add more files to the data set. We will provide code for loading the input data.

6.1 Map file

The map file contains summary information about the terrain, plus the names and positions of various terrain objects. We will provide code for importing the terrain description. The main entry-point to this API is the function

```
Map_t *LoadTerrain (const char *terrain, float vehiclePos[3]);
```

This function takes the name of the *directory* containing the terrain data set and returns a *map* object, which contains in-memory versions of the data.

6.2 Height-field data

The heightfield data is stored as a *Portable Grey Map* (PGM) file with 16-bit samples. Its dimension will be $2^n + 1$ samples on a side (*i.e.*, $2^n \times 2^n$ grid cells). The horizontal scale (*i.e.*, distance between grid points) and vertical scale are given as part of the map file. The `LoadTerrain` function reads in the heightfield information.

7 Submission

We will set up a **gforge** repository for each group on the Computer Science server (see the *Lab notes* (Handout 2) for more information on **gforge**). We will collect the projects at 3pm on Wednesday February 27th from the repositories, so make sure that you have committed your final version before then. You will also be expected to demonstrate your code during Lab in Week 8 (February 27).

History

2008-02-21 Added description of how grid cells are split.

2008-02-21 Original version.