CPS 713

TERRAIN RENDERING FOR A GEOGRAPHIC INFORMATION SYSTEM

Paul Coddington

Northeast Parallel Architectures Center

November 1994

SPECIAL FEATURES OF TERRAIN RENDERING

Rendering of Terrain

Terrain rendering is a special case of 3-D volume rendering:

- Firstly, the objects are static – only the viewer moves.
- Secondly, there is only a single simply connected object – the land surface.
- Thirdly, a terrain model is basically a 2-D surface with a variable elevation in the third dimension, which is much simpler than a general 3-D problem. Terrain rendering is often called a 2.5 dimensional problem.

For these reasons, there are a number of optimizations and simplifications that can be made for a terrain rendering program that cannot be done for a general purpose volume rendering program.

Portability and Scalable Rendering

For best results, the rendering must be done on a high-performance computer. However the renderer should be portable, so it will run on parallel computers with different numbers of nodes, high-end graphics workstations, and even low-end platforms like Macs.

Our strategy is to make a scalable algorithm that will always run in near real time, by (gracefully) sacrificing image quality when there is less compute power available. This can be done relatively easily using polygon-based rendering by coarsening, i.e. reducing the number of polygons used.

This method requires handling two key problem areas which are interesting research topics:

- Keeping a constant frame rate for different computers and different scenes
- Creating optimal coarsenings of the polygons

Terrain Object Representation

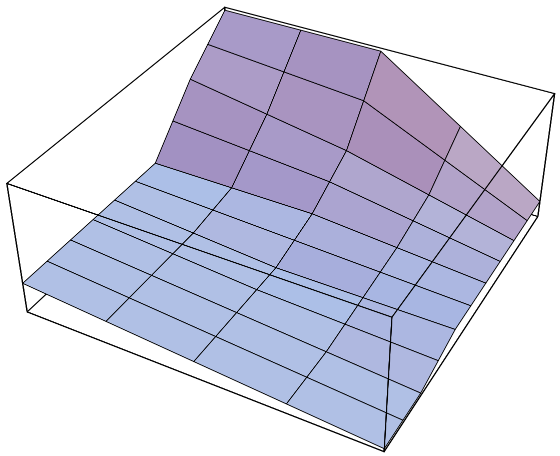
A terrain object is defined by an elevation (height) field and an image. The elevation field encodes information as elevation y for a specific (x, z) coordinate. The image gives the color associated with the coordinate. This information can be obtained from a digital elevation model (DEM) and a satellite image of the terrain.

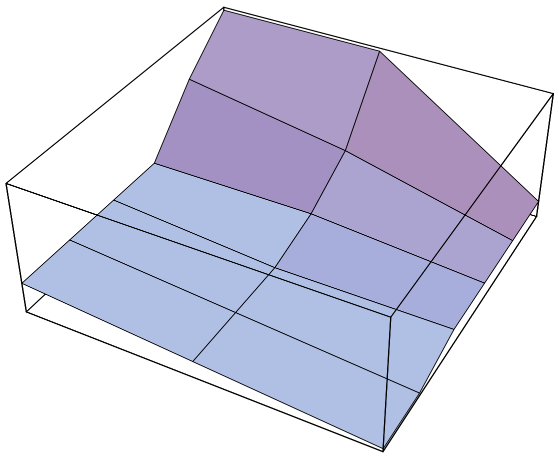
A terrain is a simply connected surface (i.e. no disjoint sections), so terrain rendering is a special case of volume rendering, sometimes described as 2.5-D rather than 3-D. This allows some optimizations to be used in the rendering process.

The terrain data set is stored as a number of two dimensional grids. Each grid contains a subsection of the terrain data set. The data is stored as triangular polygons made by triangulating the square mesh used as the basis of the DEM.

Polygon Representation

The terrain data can be stored at different levels of resolution. Coarser resolution polygons can be used for terrain further from the viewer. This will greatly reduce the computational requirement. Can do this very simply by just using a coarser mesh, but may be more effective to use an adaptive scheme with a very coarse mesh for flat regions (lakes, plains) and a less coarse mesh for mountains.





Shading and Shadowing

Because the light source remains constant and terrain is virtually non-reflective, shading and cast shadows do not change from one view to another. Thus a significant portion of the rendering process, the computation of shadows and shading, can be done in a preprocessing step.

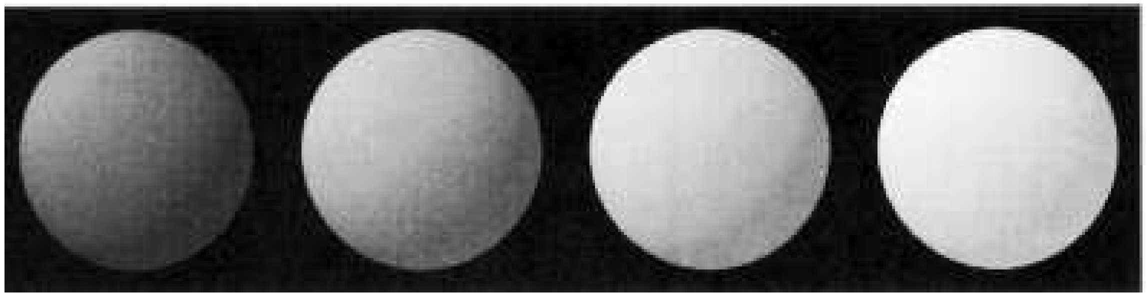
Shading can be modeled as two separate processes:

Diffuse light is uniformly reflected from surfaces, i.e. does not depend on the angle of the surface to the viewer, only to the light source.
Ambient light is radiated from all the objects in the scene, and is independent of the light source.

Diffuse light is blocked in areas of shadow, so must compute the areas in shade (occluded from the light source). Ambient light is not affected by shadowing.

Shading Model

Different materials have different parameters for ambient and diffuse reflection, which are used in a standard Lambertian shading model to calculate the intensities for ambient and diffuse light.

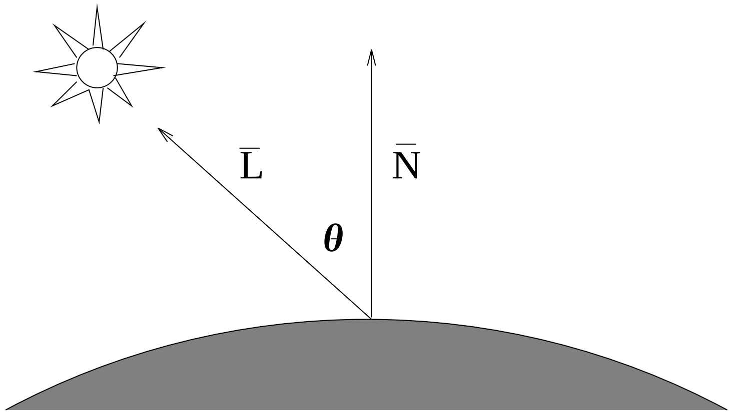


Example of a shaded terrain.

Shading Model

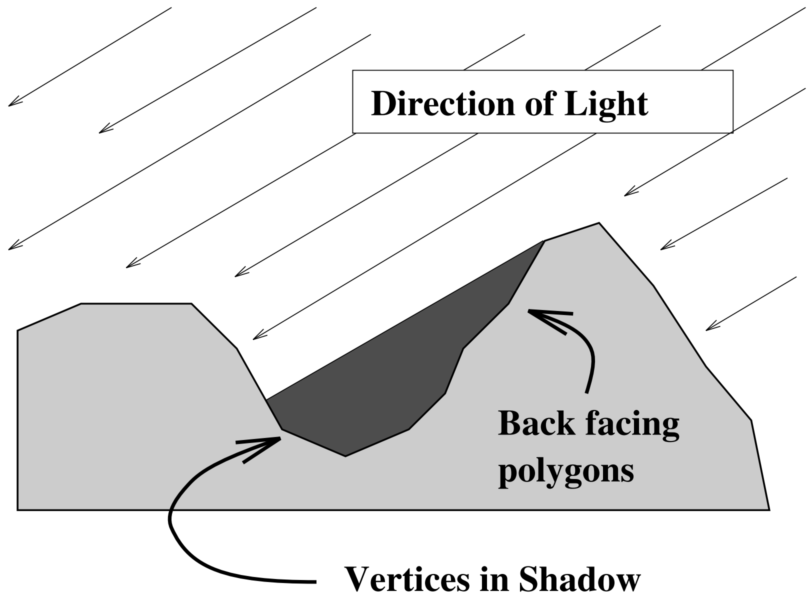
Ambient light	$I = I_a k_a$
Diffuse light	$I = I_p k_d \cos\theta = I_p k_d \overline{N} \cdot \overline{L}$
Together	$I = I_a k_a + S_i I_p k_d \overline{N} \cdot \overline{L}$

$$
\begin{aligned}
I &= \text{total intensity} \\
I_p(I_a) &= \text{diffuse (ambient) intensity} \\
k_a &= \text{ambient reflection coefficient} \\
k_d &= \text{diffuse reflection coefficient} \\
\overline{N} &= \text{normal to the surface} \\
\overline{L} &= \text{position of the sun} \\
\theta &= \text{angle between sun and normal} \\
S_i &= 0 \quad \text{if in shadow} \\
&= 1 \quad \text{if not in shadow}
\end{aligned}
$$



Shadowing

A Z-buffer algorithm is used to calculate the area in shadow, which is identified by finding the back-facing polygons.



REMOTE DISPLAY OF RESULTS

Client/Server Model

Since real-time rendering is so compute-intensive, it must be done on a high-performance computer to get the best results. However we want to be able to display the results on a workstation on the user's desk, or even on a Mac or PC in a high school.

The results must therefore be sent interactively across a high-speed network from the rendering engine to the display monitor. The rendering engine will be running on a high-performance computer, or perhaps a high-end workstation with special graphics hardware.

We must therefore deal with issues of networking and image compression.

Networking Issues

Require about 30 frames per second for realistic animation. A typical frame size would be 640×480 pixels, and ideally each pixel would have 24 bits of color information. This would require about 220 Mbits per second of bandwidth to send the raw image.

Current FDDI and ATM networking technologies available at NPAC can achieve about 100 Megabits per second, however actual performance (using software such as PVM) can be an order of magnitude less than this. This is clearly a difficult but crucial problem that needs to be addressed.

There are three ways of reducing the bandwidth required:

- Reduce the image size (not a favorable option)
- Reduce the frame rate (10-20 fps is probably adequate)
- Reduce the image size by compressing the image

Image Compression

Images need a lot of data, and animated images are even more of a bandwidth hog, so there is a lot of interest in methods for compressing images and video. JPEG compression is fast becoming the industry standard for images, MPEG for video. Other methods include Wavelet and Fractal compression.

JPEG decreases image quality (i.e. is lossy), but in areas that are insensitive to human vision. At compression ratios of 10:1 the image is virtually indistinguishable from the original, and generally can achieve 20:1 without serious loss of image quality.

However standard JPEG compression algorithms are much too slow for real time animation (about 0.2 seconds to compress a 320×200 image on a DEC Alpha). Fast parallel or hardware implementations are needed. Decompression is fast, and cheap special purpose chips will soon become standard features.