Non-Photorealistic Rendering of Stylized Block Diagrams

**James E. Mower**

**Department of Geography and Planning**

**AS 218, University at Albany**

**Albany, New York 12222 USA**

**jmower@albany.edu**

**ABSTRACT:** This paper will introduce object-space procedures for extracting silhouettes, slope lines, and drainage features from DEMs to direct the non-photorealistic rendering of landforms as stylized block diagrams. The ultimate goal of this work is to produce fully-automated tools that can imitate the style of illustrations characterized by traditional pen and ink techniques. Through the implementation of a Java 3D API prototype, this project establishes a testing platform for the application of stylistic elements to linework representing surface form lines. This project will begin to explore the aesthetic effects of slope ranges, angles between adjacent surface facets, and drainage features on the rendering of silhouette lines, surface creases, and hachures. It also introduces the use of adaptively resampled TIN as a basis for perspective rendering applications.

**KEYWORDS:** Non-photorealistic rendering, silhouette, DEM, block diagram

Introduction

Block diagrams are large-scale illustrations of surface features and their underlying geologic structures, drawn from a particular point of view following the rules of artistic perspective. Armin Lobeck, whose *Block Diagrams* has remained one of the most influential guides to their manual construction, argues that these illustrations gain much of their effectiveness from their visual simplicity (Lobeck 1958, p. 1). This paper will explore the application of his aesthetic to the automatic construction of block diagrams from digital elevation models (DEMs) under the rubric of non-photorealistic rendering. Following a review of earlier work on automated systems, I will introduce a new testing platform for stylistic experimentation, provide the relevant algorithms underlying its implementation, and discuss the contributions of particular graphic elements that contribute to good expression of landscape form.

Previous Related Work

Over the past decade, the rapid development cycle of fast, relatively inexpensive graphics processing chips along with concomitant developments in 3D application programming interfaces (APIs) has fostered a period of intense growth in perspective rendering applications in cartography, GIS, and other non-geographic application areas. Much of this work has been related to realistic renderings of landscapes, whether for gaming, site planning, or other uses that require a relatively ‘complete’ description of a scene. Another rendering paradigm, non-photorealistic rendering (NPR), provides a closer analogy to the construction of traditional perspective illustrations in cartography by fostering the communication of specific, often scientific information through a language composed of symbols with well-understood meanings (Kennelly and Kimerling 2006, p. 36). This definition is consistent with the goals of other cartographic forms that use the rules and constraints of generalization to establish a message for an image and to focus a viewer’s attention upon it.

Starting with an image space approach, Saito and Takahashi (1980) lay out many of the ground rules for NPR using 2D digital images as their base data. Their successful demonstration of techniques for generating simplified line drawings from photographed 3D objects established a vocabulary for further NPR research on the extraction of edges, rendered as silhouette lines, and components of surface curvature, represented by hachures. Isenberg, et al. (2003) provide a comprehensive classification of NPR modeling techniques for both image and object space approaches along with definitions of terminology. Kennelly and Kimerling (2006) provide an excellent overview of those approaches that are specific to renderings of landforms, placing them in the context of the manual methods defined and described by Raisz, Imhof, Lobeck, and other 20th century cartographers and illustrators.

Most current NPR research in landform representation has focused on object space approaches, deriving perspective line drawn images entirely from 3D spatial data represented in a DEM. Lesage and Visvalingam (2002) use this approach to implement p-stroke sketching, which is itself closely related to the inclined contour method of Tanaka (1932). Buchin, et al. (2004) model surface slope to apply scanned, hand-drawn slope and loose lines from a look-up table, dependent on surface slope and lighting conditions.

This paper will focus on the development of a robust technique for representing silhouettes and hachures from adaptively resampled DEMs in the form of triangulated irregular networks (TINs). By resampling a DEM with greater tolerance for surface error away from a selected viewpoint, triangle areas remain relatively constant, given constant surface variability, with distance from the viewpoint under perspective rendering. It is thus possible to extract and render features from a resampled DEM without need for line simplification in a post-rendering operation. This paper will also introduce the extraction of drainage features from DEMs to aid in the rendering of view-independent features (creases) that are essential to surface interpretation

Modeling Surfaces with Linework

Unlike most photorealistic rendering applications which determine shading values for polygonal surface facets based on the angle between incident illumination and the surface normal, this project constructs line images as if they were lit strictly by an ambient (non-directional) light source. And while virtually all modern photorealistic rendering systems use some kind of hidden surface algorithm to ensure that surfaces near a viewer block the view of those further along the viewer’s line of sight, this project must also account for the viewer’s position in constructing silhouettes.

Throughout this project, all facets are represented as 3D triangles within a variable resolution TIN (VTIN). Each VTIN triangle maintains a list of its neighboring triangles. The VTIN is also indexed by edges, where each edge record includes the IDs of its bordering triangles.

Silhouettes

A silhouette segment is a rendered edge separating a visible and invisible facet, where visibility is determined by the angle made between the viewer’s line of sight and the facet’s surface normal (Figure X). In this sense, visibility has nothing to do with the presence or absence of obscuring surfaces between the viewer and the facet in question. A silhouette can be thought of as a ‘visibility ridge’ from the viewer’s point of view. Silhouette segments are computed locally as specified by Algorithm Build Silhouettes:

Algorithm Build Silhouettes

To build silhouette lines:

1. For each facet,
   1. Determine the angle between the viewer’s line of sight and the surface normal
   2. If the angle is between 0 and 90 degrees, mark the facet as visible. Otherwise, mark it invisible
2. For each edge,
   1. If one of the bordering facets is visible and the other is invisible,
      1. If neither of the 2 non-edge vertices on the bordering triangles is higher than either of the edge vertices,
      2. Render the edge as a silhouette segment

Note that hidden line removal is not an integral part of this algorithm. Instead, silhouette segments are rendered along with their referent surfaces, themselves rendered with emissive lighting in the background color of the rendered frame. Emissive lighting provides no differentiation in surface shading with respect to light source direction and surface normals. Surfaces rendered in this manner are invisible against the background, yet supply appropriate masking for silhouette segments (Figure X). [talk about z offset in later implementation section]

Step 2.1 specifies that an edge should be rendered as a silhouette segment if and only if neither of the non-edge vertices in either bordering triangle is higher than either of the 2 edge vertices. This condition prevents the rendering of silhouettes for the feature shown in Figure X.

Slope Lines

On a planimetric topographic map, hachures are linear symbols representing paths of steepest slope on a surface patch, drawn perpendicular to contours. On a block diagram, slope lines perform a similar function. Although hand-drawn examples are typically rendered as complex curves, this project renders short slope lines as straight-line segments for initial simplicity of implementation (Lobeck 1958, p.33). Other slope lines in this project are rendered as sections of drainage networks and are discussed in the following section on surface creases. The following algorithm, Build Slope Lines, outlines the first of the two processes (Figure X):

Algorithm Build Slope Lines

To build slope lines:

1. For each facet,
   1. Find the coordinates of the facet centroid
   2. Find the equation of the plane of the facet
   3. If the facet slope exceeds the preset minimum value,
      1. Find the equation of the line of steepest descent through the facet centroid
      2. Render the segment of the line within the facet up to a preselected margin bordering the edge of the facet.

Following common practice for manually constructed block diagrams, the algorithm creates slope lines of uniform width. Although it could be modified to scale the width of the rendered line to the slope of the facet, this is largely unnecessary. As vertical resolution decreases away from the viewpoint, the screen space areas of rendered triangles remain relatively uniform given constant surface variablility. Therefore, the separation of rendered slope lines remains a good visual indicator of surface slope regardless of the 3D distance of the surface patch from the viewpoint. [NEED SOME IMPLEMENTATION EXAMPLES HERE]

Surface Creases

In the NPR literature, a crease is a line representing an edge that carries important visual information independent of the viewer’s line of sight (Isenberg, et al. 2003, p. 28). For block diagrams, hydrographic features such as ridges and drainage channels would typically be resented by creases which can be extracted and rendered in a number of different ways. One such method examines the angle made by normals of adjacent triangles and renders their shared edge as a crease if the angle falls within a given range (Figure X). However, this analysis does not provide particularly good results for topographic surfaces. In relatively high-resolution elevation models (such as that used by this project), ridge and channel features rarely exhibit sudden slope breaks along connected triangle edges (Figure X). I found that a drainage accumulation analysis provided a visual solution much closer to that of hand-drawn illustrations. Algorithm Build Creases summarizes this approach:

Algorithm Build Creases

To build creases:

1. Initialize the drainID for each facet to RIDGE
2. For each facet,
   1. Find the centroid of the neighboring facet with the steepest downhill slope from the current facet. Note its ID as the current facet’s drainID. If a facet has no downhill neighbor, call it a pit.
3. Initialize all facets with 1 unit of input water and 0 units of accumulated water
4. While any but pit facets contains greater than 0 units of input water,
   1. For each facet,
      1. Decrement its total input water value and increment both the drainID facet’s input water value and its accumulated water value by the total input water value.
5. For each facet,
   1. Render a line between the current facet’s centroid and that of the drainID if the drainID facet’s accumulated water value is greater than a preset threshold value.

Build Creases provides a relatively localized method for constructing drainage lines. Steps 1 through 2 create a drainage direction map, indicating the drain for each triangular facet. Following step 2, those triangles with no uphill source are on a ridge. Pits are triangles that have no exterior drainage.

Most drainage direction models built from data originally extracted from grid cell DEMs (like that of the current project) will find numerous ‘false pits.’ A false pit commonly occurs when a grid sampling pattern extracts an elevation from a stream channel but its 8 surrounding neighbors all remain uphill on the stream bank above it (Mower 19xx). Since the channel itself is not explicitly extracted, channel samples are rarely adjacent to one another and thus have no drainage outlet. Unlike a hydrographically realistic drainage model, the crease algorithm does not bother, for now, to ‘flood’ false pits to assume a regular drainage pattern.

Steps 3 through 4 run a drainage accumulation model through the drainage direction map. At each step of the algorithm, an uphill neighbor transfers its water supply to its downhill neighbor (indicated by drainID) and tracks the total amount of water passing through the downhill neighbor over the life of the procedure. Step 4 ends when all the remaining water is in the pits,

Step 5 inspects the drainage accumulation of each downhill neighbor of a facet. If the accumulated value is above a preset threshold, a line segment is rendered between the 2 triangle centroids. Depending on the value of the threshold, the procedure will draw more or less of the stream network detail. For the moment, at current threshold values, ignoring false pits does not negatively impact the line drawing. However, given that drainage networks are promising crease generators, future enhancements may require an additional flooding procedure to remove potential image artifacts.

The Implementation

The block diagram implementation has been written in Java and the Java 3D API. Although earlier work on this and related rendering projects used C# and the Direct3D API, I found that Direct3D (the Microsoft 3D rendering library) provided insufficient control for line widths, critical to the successful implementation of this project. Since OpenGL provides such control, and since the Java 3D API makes OpenGL calls on a client computer by default, this environment provides a natural platform for experimentation and collaboration across the Internet.

Adaptive Resampling and TIN creation

The project data originally came from grid cell DEMs in native USGS 1:24000 format produced by the New York State Department of Environmental Conservation and distributed by the Cornell University Geospatial Information Repository (CUGIR). For ease of use, the data was transformed into 10 km by 10km grids from which a dense TIN (DTIN) was created (Mower in press). The DTIN is formed by a recursive resampling of the region files such that every retained sample is recorded along with an error term that measures its orthogonal distance from its parent plane. A variable TIN (VTIN) is created by scanning the DTIN and selecting those samples whose error term is greater than a tolerance whose value increases linearly with distance from a specified viewpoint. The VTIN provides a relatively constant number of samples per pixel for perspective rendering applications.

Results

Figures A shows a rendered surface with flat ambient shading for reference. Figure B displays the same scene, rendering the surface with emissive shading and including silhouette lines. Figure C adds slope lines, and Figure D adds drainage creases.



Figure 1: A gratuitous graphic. Source: http://www.hgiis.com/images/pic\_app\_cartography.jpg.

This line is centered and uses the Figure Caption Char label in the window at the upper left.

When you need to insert an equation, just go ahead and use the Word equation editor. The equation can be formatted with the Equation Line One, Equation Line Two template. For example:

E = m c2

There is no need to number equations, unless it is important to refer to them. Tables can be treated the same as figures, and embedded into the Word file. They should be referred to in the text, and have their own number sequence, independent of figures. They also need captions. Be sure to try and avoid figures and tables rolling over or getting split between pages. You can usually move them around until they appear on their own page. If not, you can force a new page using the Insert Tab and Break menu choice. Use the standard table tools in word, or import the table from Excel or equivalent (Table 1).

Table 1: A meaningless table, added to show the basics of formatting.

|  |  |  |  |
| --- | --- | --- | --- |
| ***Distance from Point A (mm)*** | ***Distance from Point B (mm)*** | ***Value assigned*** | ***Purpose*** |
| **1.2** | 1.3 | 1000 kP | Instruction |
| **3.1** | 2.1 | 1020 kP | Distraction |

Many papers follow an IMRAD format, that is Introduction, Methods, Results, Analysis and Discussion (Davis, 2005). Of course there are many different paper formats. Follow the one that communicates your findings best. Use passive voice for preference, avoid grey literature, and use the third person.

Future Work

Drainage calculation on the fly

Resampling to VTIN on the fly

Conclusions

If we all follow this easy to use template, hours of valuable conference time will be saved in preparing the conference proceeding. There is no word or page limit on your paper, but papers should be in the order of 12 pages long, single spaced, and including references and figures, etc.

REFERENCES

Davis, M. (2005) ***Scientific Papers and Presentations*,** 2ed. Oxford: Elsevier Academic Press.

Smith, A. (1999) Maps: A Laugh a Minute. ***Hungarian Journal of Cartographic Logic***, 11, 6, pp. 145-180.

Smith, A. and Jones, B. (1991) More Cartographic Hilarity. ***Cartography and Geographic Information Science.*** 22, 1, pp. 123-145.

Smith, A. Jones, B. and Smythers, C. (1992) Two Years of Cartographic Humor. <http://www.funnymaps.com> Last visited 5/19/2008.

**Keith C. Clarke**, Professor, Department of Geography, University of California, Santa Barbara. Santa Barbara, CA 93106-4060. E-mail:<[kclarke@geog.ucsb.edu](mailto:kclarke@geog.ucsb.edu)>.