**Project Title:** NPR Terrain Rendering (Line Drawings – Slope Steepness)

**Student Name:** Ian Stewart

**ID:** 10045484

1. **Introduction**
   1. **Goal**
      1. **What did we try to do?**

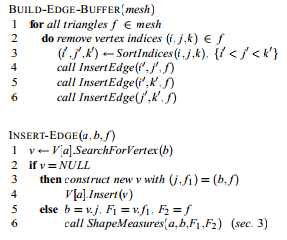
My goal was to render a terrain model as a precise line drawing. I wanted the line drawing to be as realistic as possible to show as much detail as possible about the terrain. The reason for choosing a terrain model was due to all of the hills (slopes) it contains. Since hills do not render as a silhouette or boundary, I thought this would be a difficult task. I wanted the line drawings to simulate shading (i.e. a sun shining down from above the terrain) and I wanted to be able to move the “sun” around the scene and move the camera in real time. I wanted to be able to change the style of the lines (dotted for example) and the color of the lines. I also wanted to be able to shade the model (in gooch for example) and apply textures to the model with the lines still visible.

* + 1. **Who would benefit?**

Anyone wanting renderings of precise line drawings would benefit since the slope characteristics of the object will be clearly visible. Line drawings can represent any shape if used properly. If a terrain is rendered as a line drawing, the technique will be able to extend to any shape with curvature (slope). Specifically, the technique could be used in artistic, scientific, and technical drawings.

* 1. **Previous Work**
     1. **What related work have other people done? 🡨 For each paper write a maximum half-page summary describing the key technical idea, approach used**

Paper [1] presents a system for NPR of precise line drawings over triangle meshes. It describes an edge-buffer technique that automatically calculates shape measures at each edge in order to determine whether to draw the line or not (it extends the edge buffer technique of rendering line drawings with a new *ShapeMeasures()* function). The edge-buffer technique described is the same as the one described in class except for one difference: each edge now contains adjacent triangle faces F1 and F2. From [1]:



The ShapeMesures() function (called from InsertEdge(), line 6) is the important part. It uses the information about an edge’s adjacent faces to decide whether to draw the edge or not. The paper mentions many different measures. The dihedral angle (creases in assignment 3) is mentioned, however, the paper also describes some more measures that follow from surface geomorphometry. Parameters that characterize the terrain surface (morphometric variables (MVs)) are produced from the Digital Elevation Models (DEMs) that model the surface geomorphometry. DEMs simply consist of elevation points on a regular grid. MVs include local shape measures such as slope, aspect, pro-ﬁle curvature, plan curvature, etc. MVs can be calculated by approximating the differential operators at every point in the DEM. The paper describes a 9-point finite difference scheme to approximate these derivatives using a, b, F1 and F2 (parameters of ShapeMeasures()), which leads to the shape measures of slope steepness, slope aspect, and mean curvature. The paper then talks about adjusting the thickness and marking styles to create better line drawings. I will not go into details about these results since I will not be implementing them.

* + 1. **When do previous approaches fail/succeed?**

The shape measure of slope steepness successfully shows steep variations in an object which is what I want for the terrain model. The other approaches of slope aspect and mean curvature are still successful when “indicating hatching marks as clusters of parallel lines following a particular drawing direction,” and when detecting concave and convex formations across the mesh respectively. They however do not create the exact effect I am looking for (slope). From [1]:



* 1. **Approach**
     1. **What approach did we try?**

I tried the slope steepness approach for the reasons mentioned above.

* + 1. **Under what circumstances do we think it should work well?**

The slope steepness approach as described in the paper [1] shows that it should work well for rendering curved objects with line drawings. Since a terrain model has lots of hills/mountains, it should be able to render them properly.

* + 1. **Why do we think it should work well under those circumstances?**

Slope steepness should work well for rendering curved objects since it is a measure of rate of change of elevation (“an angle between 0 and 90 degrees between a horizontal plane and a tangential to land surface plane at the same point”). Thus, as long as the minimum value for GA is set properly (reasonably higher than 0 and lower than 90), it should only draw lines that are getting more parallel to the z-axis, which should show curvature.

1. **Methodology**
   1. **What pieces had to be implemented to execute my approach?**
2. Edge buffer data structure – an array of lists of edges. Each edge consists of V (second vertex (first is the array index)), T1 (triangle face 1), T2 (triangle face 2), Acmodel (artist flag for crease model), Acuser (artist flag for crease user), A\_SlopeS (artist flag for slope steepness), F (front facing), B (back facing), Fa (front absolute), and Ba (back absolute). As specified in assignment 3, if FB = 11 the edge is part of the silhouette, if FB = 01 or 10 the edge is part of the boundary.
3. The artist flags had to be set according to their respective definitions using T1 and T2. Creases are set by calculating the angle between the normal of T1 and T2. Slope steepness is set from the derivative approximation mentioned in paper [1].
4. Edge buffer had to be rendered properly with shading in mind (drawing them closer to the camera to manage the z-buffer properly and drawing them on a second pass).
5. User interface with all relevant options functioning properly. See section 3.1 for all options I wanted.
   1. **For each piece ...**
      1. **Were there several possible implementations?**
6. Yes there were. I implemented the edge buffer as a 2D array (number of vertices by maximum number of adjacent vertices to any vertex). This implementation works well with all of the models provided since most of them have a small number edges from any single vertex (usually around 10 or less). In particular the terrain model has exactly 6 edges from each vertex. However, there is some empty space in a 2D array and an array of linked lists or similar data structure could have been used instead to manage memory better.
7. No, slope steepness is calculated as specified in the paper [1].
8. No, the z-buffer had to be taken into consideration.
9. No, a functional user interface was required.
   * 1. **If there were several possibilities, what were the advantages/disadvantages of each?**
10. The advantage of an array was it was easier to implement do to random access and ran faster. Linked lists are more memory efficient (pointer size should be irrelevant considering the large size of the edge buffer in most cases) and it is easier to insert/delete elements, but they are slower and have sequential access so it is harder to get at specific elements. Thus, when trying to find a certain edge in the edge buffer to decide whether to draw it or not, the array implementation is faster.
    * 1. **Which implementation(s) did we do? Why?**

As said earlier, I implemented the edge buffer as a 2D array for ease of use and performance.

* + 1. **What did we implement? 🡨Include detailed descriptions**

See section 2.1, everything was implemented and detailed descriptions are provided there and in the paper [1] for slope steepness formula.

* + 1. **What didn't we implement? Why not?**

Everything I wanted to implement was implemented. Again, slope aspect and mean curvature were not implemented since they did not produce the effect I was looking for.

1. **Results**
   1. **How did we measure success?**

I deemed a successful result as a result that included all of the following:

1. A complete edge buffer structure with all of the necessary components (V, T1, T2, Acmodel, Acuser, A\_SlopeS, F, B, Fa, and Ba).
2. Silhouette, boundary, and creases (dihedral angle) of a terrain model able to be rendered as a line drawing. The model could then be rotated and the camera could be moved. (Assignment 3)
3. Slope steepness fully implemented to show the slopes of a terrain model.
4. Gooch shading could be used with the line drawings (assignment 1).
5. Attribute based texture mapping could be used with the line drawings (assignment 2)
6. A reasonably large terrain model could be rendered (large amount of triangles) to show slope steepness as best as possible (the higher the resolution, the better the result).
7. A fully functional user interface that included all necessary functions. This includes all user interface options from assignments 1, 2 and 3 (including the bonuses) and options for slope steepness (color, style, z-axis direction and angle min/max (GA from 0 to 90 degrees)).
   1. **What experiments did we execute?**

* I printed out the edge buffer for (1) to make sure it was correct.
* I completed assignments 1, 2, and 3 including all of the bonuses for (2), (4), (5), (6), and part of (7).
* For (3) and the rest of (7) I made sure that the slope steepness effect looked the same as on the paper [1] (as shown in section 1.2.2). I then made sure that everything functioned the way I wanted (z-axis rotation for slope steepness, color, style, etc.). Lastly, I tested the result on a high polygon terrain model to see the results.
  1. **Provide quantitative results.**

Pictures are provided below to show results (GA between 27.5 and 57.3 degrees). The slope steepness direction is the z-axis unless otherwise specified.

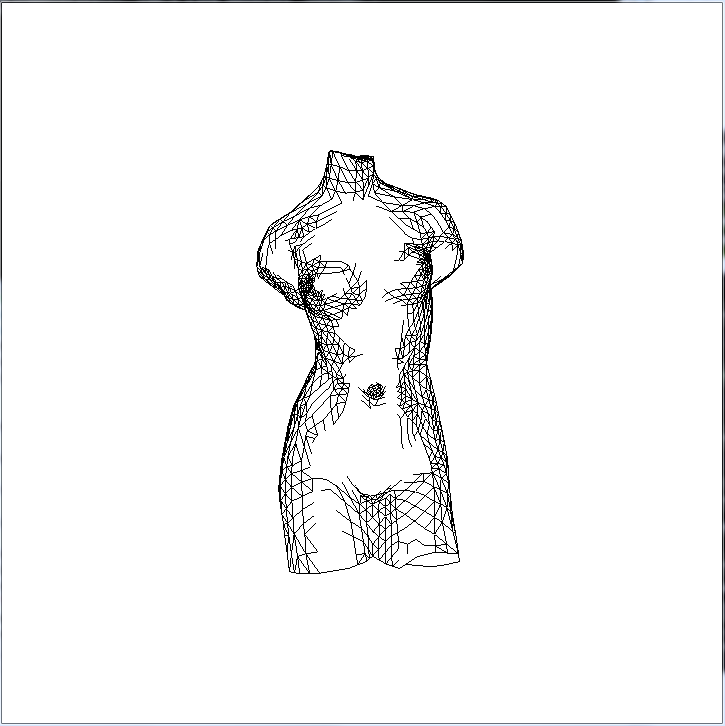
 

Figure 1: Venus Model Figure 2: Terrain Model

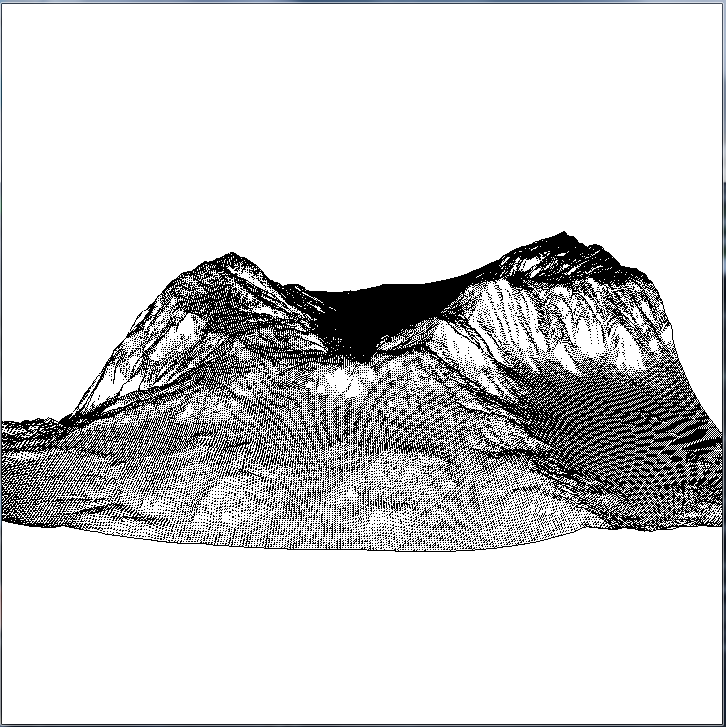
 

Figure 3: Terrain Model with Slope Steepness Figure 4: Terrain stylized with dotted lines

effect rotated 45 degrees around both the

x and y axis (looks like the sun is in the upper

right of the picture).

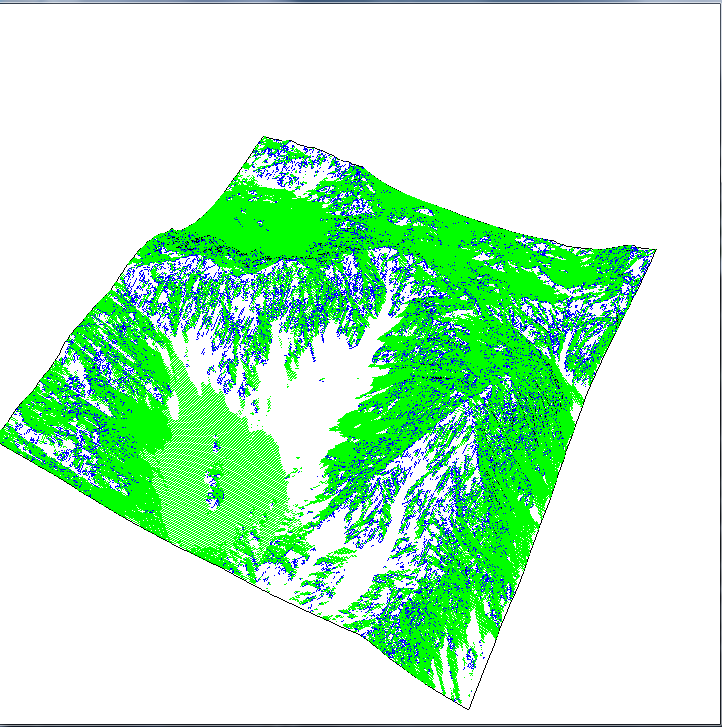
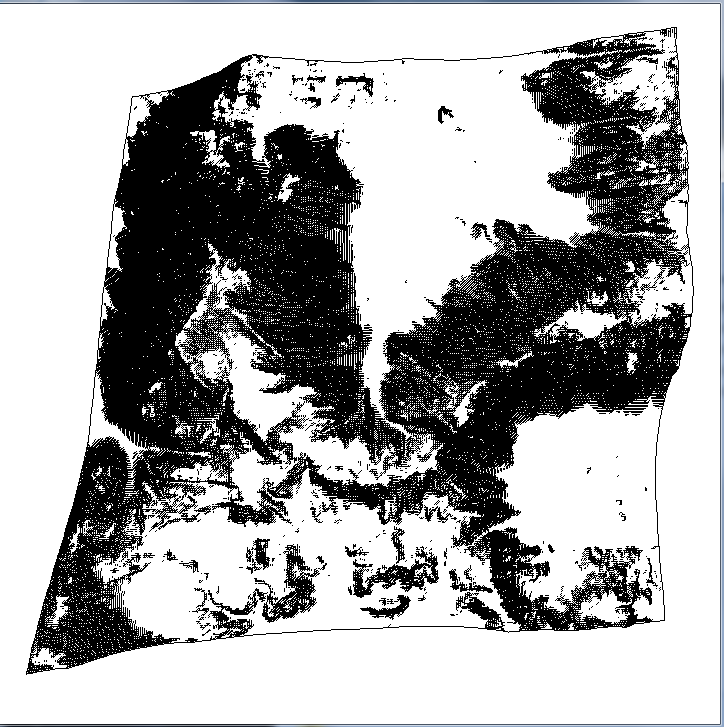
 

Figure 5: Terrain showing the difference between Figure 6: Overhead view of the terrain

slope steepness (green) and creases

(blue – between 20 and 40 degrees).

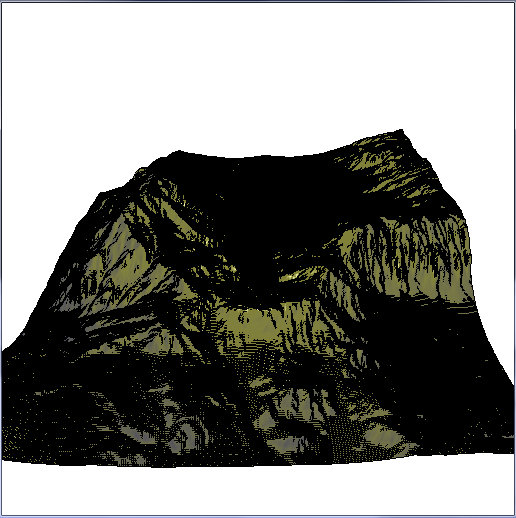
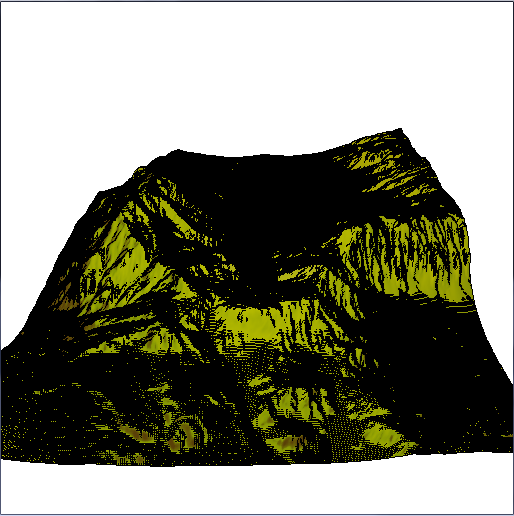
 

Figure 7: Terrain with gooch shading from Figure 8: Terrain with aerial perspective texture

assignment 1 and slope steepness. from assignment 2 and slope steepness.

* 1. **What do my results indicate?**

The results indicate slope on the terrain model using line drawings. The drawing can even be thought of as simulating a light source. In figure 1 it looks like the light source is directly at the eye. Rotating the slope steepness direction results in the light source being at a different location, as seen in figure 3. This means that the light direction is completely independent of the slope steepness direction when using shading or texture mapping (figures 7 and 8). Also, figure 5 shows that using creases from assignment 3 are not nearly as good at showing the mountains as slope steepness.

1. **Discussions**
   1. **Overall, is the approach we took promising?**

Yes, slope steepness works very well for what I was trying to achieve.

* 1. **What different approach or variant of this approach is better?**

Adding in the other approaches from the paper [1] (slope aspect, mean curvature, and line thickness and marking styles) could have made the line drawings even better.

* 1. **What follow-up work should be done next?**

Adding in the other approaches from the paper [1] and perhaps trying to find a method that produces the same result of slope steepness in similar time, but does not rely on the resolution of the mesh.

* 1. **What did we learn by doing this project?**
* How to render precise line drawings of objects using various techniques.
* How to combine line drawings with shading and texture mapping.
* More about C++ and OpenGL.

1. **Conclusions**

Slope steepness is a very good way of rendering line drawings that have curvature (hills, mountains, etc.). It is a very fast technique as well since it uses the edge-buffer which only uses OR and XOR operations and draws only existing edges on a mesh. However, this means that it does rely on the resolution of the mesh to produce a high quality image. There are also many other techniques that can be added to the edge buffer to make it produce better results such as slope aspect, mean curvature, and line thickness and marking style techniques.

1. **References**

**List of the most relevant papers/books useful for the project**

[1] Mario Costa Sousa, Kevin Foster, Brian Wyvill, Faramarz Samavati,

“Precise Ink Drawing of 3D Models,”

<http://ires.cpsc.ucalgary.ca/publications/2003/few-good-lines.pdf>

1. **Appendix**

**Any additional technical dtail complementing the main text**