The Landscape Used For the Flight Program

C. Godsalve
email: seagods@hotmail.com

December 17, 2010

The basis of the landscape used in the "Flight" program is a triangular network. Triangular networks may be regular or irregular, the Triangular Irregular Network (TIN) for instance, is a regular workhorse for Geographic Information Systems (GIS)[1]. It just consists of a set of points or *nodes* which are connected together so that a given region of the plane is covered by a mesh of triangles. For such a network to be of practical use in GIS (for instance) there are certain rules about the triangulation. Pretty well, they are demonstrated in Fig.1 where we see three basic rules. In Fig.1 part C, what do we mean by

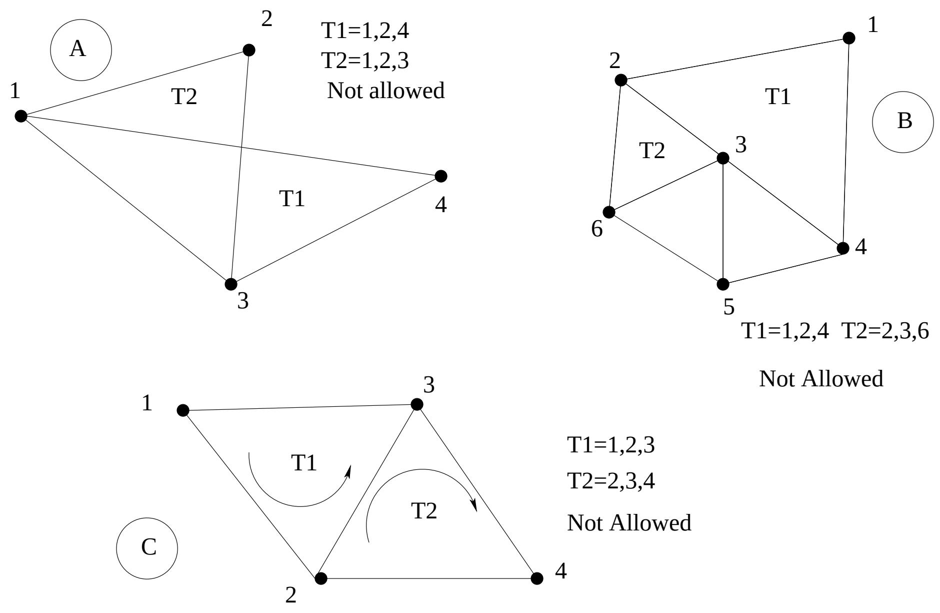


Figure 1: A; triangles cannot overlap: B; nodes cannot occur on edges: C; triangles must have the same orientation.

the "orientation" of a triangle? Naturally, each node is given a number, and each triangle has a number, and in any triangle three nodes must be listed *in order*.

In the simple case of a single valued surface defined as heights over each node in the plane, such as occurs in many landscapes, this order is defined by whether the listing of the nodes 1, 2 and 3 in each triangle appears as clockwise or anti-clockwise from a given side of that plane. The nodes listed in the triangles must *all* be either anticlockwise or clockwise. Quite often, the nodes are not regularly positioned in any way or order. In such cases, a Delaunay triangulation[2] is usually used to go from a bunch of nodes on a plane to a triangulation.

In this article, the "landscape" generated is defined by a square array of nodes and the network is regular. Such landscapes may be thought of as a special subset of possible TINs. To generate a square consisting of $N \times N$ lesser squares, we need a square array of $(N+1) \times (N+1)$ nodes, and at each node, the height above the plane in which the nodes exist, defines our "landscape". (Heights can, of course, be negative.) If we wish our landscape to consist of simple flat facets, we must make the facets triangular. The reason is simply that, in general, four points in a three dimensional space are not generally co-planar. However, three points in a three dimensional space define a plane (unless they lie in a straight line or coincide with each other).

So, for any square consisting of $N \times N$ small(est) squares, each small square can be split into two triangles. Unfortunately, without extra information on a sub-grid scale, this does not uniquely specify the landscape as seen in Fig.2. Suppose we denote H as (relatively) high, and L as relatively low, we cannot determine whether we have a square with a "valley" along one diagonal of the square or a "ridge" on the other diagonal.

We can split each square into two triangles along either diagonal as we please. For a realistic attempt at representing the landscape we can use the given height data, and with those cases which are undecidable, we may use data from the eight surrounding squares to decide the best choice to make as to which diagonal defines the two triangles (given assumptions on the nature of the landscape.)

In the program *fractalsurf* the two sub-triangles of the grid squares are defined as in Fig.3. That is, we do not have sub-grid information, and ignore any niceties which may be used to infer cases where things cannot be decided from the data at the corners of a given square. We have called the two ways to split a square "even" and "odd" and used a chequerboard pattern to "break things up". (Making the squares all odd or all even would tend to introduce more artifacts than otherwise, another choice would be to choose odd and even at random.)

So, we wish to generate a landscape. To generate a "skyline" or 1D landscape we can use random midpoint displacement[3]. Here we may start with two values at the end of a line segment. We may take the mean, and then generate a pseudo random number from a Gaussian distribution of some variance and add it to that mean to get the value of the "skyline" at the midpoint. The end points, tegether with the new midpoint, makes two new

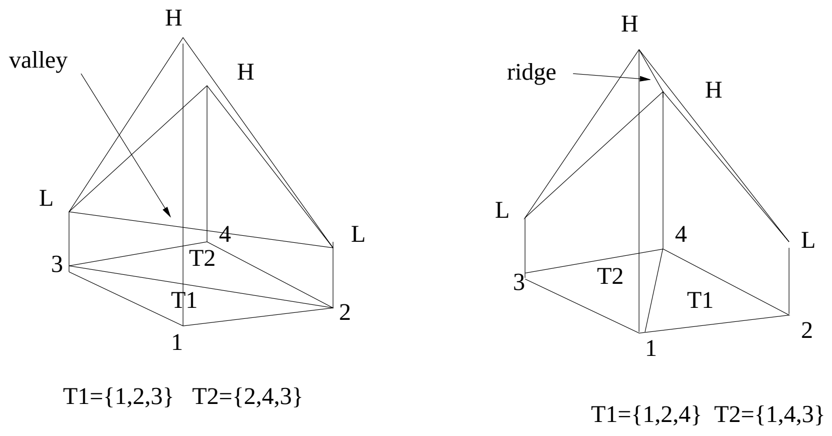
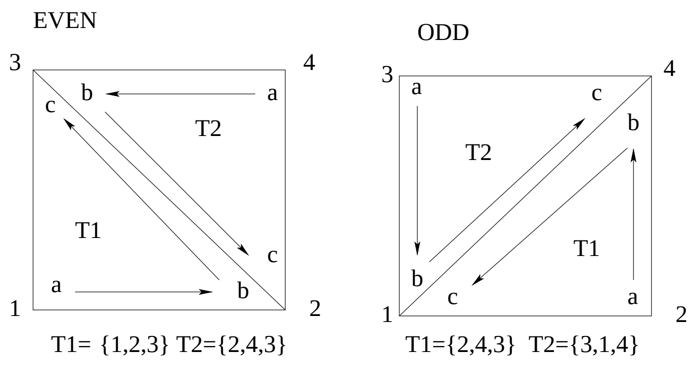


Figure 2: Without extra information we can't decide to put in a valley between nodes 2 and 4 or a ridge between nodes 1 and 3.

line segments. We may now take the two line segments and generate the midpoint values at two new midpoints. We halve the variance at this stage. Each time we end up with double the number of line segments, and we halve the variance every time. We might add a random number chosen in the same way to the set of values we are taking the midpoints of. This, and many other kinds of algorithms for generating fractal "skylines", "landscapes", and "clouds" are discussed extensively with many examples along with pseudo-code, are given in [3].

The "fractalsurf" program uses a variation of the random midpoint displacement in 2D. The triangles belong to a triangle class. As well as three node numbers, each triangle has three neighbours. If m=1, 2, or 3 are the nodes, then edge m is the edge *not* containing node m, and neighbour m is the neighbouring triangle which lies over edge m. Neighbour encoding was to facilitate geodesic propagation across the landscape at a later stage in code development. In the "Flight program", we generate a linked list of the triangles attached to each node in the mesh. This is used to generate the normals required at each node if Gouraud shading is to be used in conjunction with OpenGL lighting. It's easy enough to do this without the linked lists because of the regularity of the mesh, but it is included in case irregular meshes are used at a later stage in code development.



O	E	O	E
E	O	E	O
O	E	O	E
E	O	E	O

Figure 3: We assign which squares are odd or even in a regular chequerboard pattern without considering how to decide in cases such as Fig.2.

References

[1] K. Chang, *Introduction to Geographical Information Systems (4th Edition)* (McGraw Hill, New York, 2007).

[2] F. Peparata and M. Shamos, *Computational Geometry: An Introduction* (Springer Verlag, New York, 1985).

[3] M. F. Barnsley, R. L. Devaney, B. B. Mandelbrot, H. O. Peitgen, D. Saupe, and R. F. Voss, *The Science of Fractal Images* (Springer - Verlag, New York, Berlin, Heidelberg, London, Paris, Tokyo, 1988).