The Dynamic Terrain Correction Algorithm

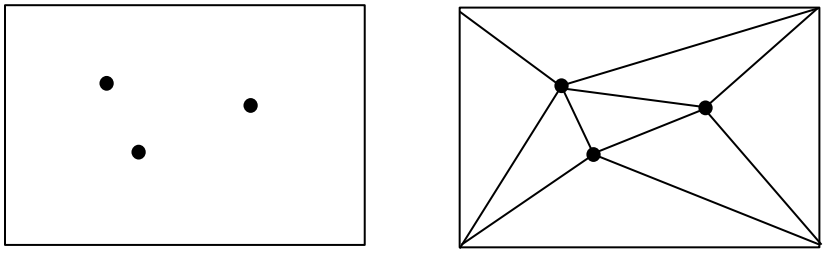
Rev. 1.0, November 24, 2005
Roy Latham

The problem is to adjust the elevation of a terrain database so that it will smoothly fit a set of newly described *targets*. Targets $t_1 \ldots t_n$ are geometric objects described by a set of polygons, with some of the polygons in contact with the terrain. Targets may be point features described by a single coordinate (x_i, y_i, z_i), lineal features described by a connected sequence of points, or areal features described by a set of polygons defining the area in contact with the terrain.

The approach adopted here is to define a correction function $C(x,y)$ such that when the correction function is added to every corresponding point of the terrain in the original database, the result will exactly conform to the targets and will vary naturally between targets. If there is a point target above the flat surface of the original database, for example, the correction function will provide a hill of the correct height to match the target.

Correction Space Meshing

The correction function is defined on the same x,y space as the original database. The first step in defining the correction function is to triangularize the correction function space with respect to the target coordinates. For example, if there are three point target in the space:



The corner vertices of the database area are included, so that the entire space is triangularized. The case of three point targets produced eight triangles in the triangularization shown. The triangularization is not unique.

There are published triangularization algorithms, also known as meshing algorithms. A well developed software set is given at http://www.cs.cmu.edu/~quake/triangle.html
A survey of literature on the subject is given at
http://www.andrew.cmu.edu/user/sowen/mesh.html

The correction function can now be defined with respect to each triangle in the meshed correction function space. The value of the correction function at each target vertex is defined by the target locations and the original terrain. The correction at the *i*th target

vertex is $C(x_i,y_i) = a_i = z_i - T(x_i,y_i)$, where $T(x,y)$ is the elevation of the original terrain at x,y. This asserts that the target elevation is always assumed to be correct and is unchangeable.

One possible correction function is that obtained with linear interpolation between the values at the vertices. The target vertices are all specified, and the vertices at the corners of the database area could be assumed to be zero. The linear function would work well if the corrections to the database were small. If the corrections are large, however, then the corrections could produce new long, unnatural, ridge lines and other abrupt changes.

Gaussian Hills and Valleys

To soften the impact of the changes we will instead create a hill or valley just in the neighborhood of each target point. For the *i*th target vertex, we define

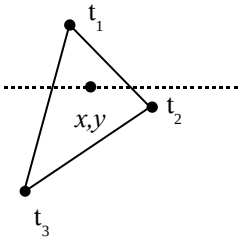
$f_i(x,y) = a_i * \exp[- d^2/k^2a_i^2]$ where $d^2 = (x-x_i)^2 + (y-y_i)^2$ and k is a slope constant, nominally 4. Increasing k makes the hills more gradual.

Each point within a triangle of the correction space has correction made of the weighted sum of three correction function values at that point,

$C(x,y) = w_1f_1(x,y) + w_2f_2(x,y) + w_3f_3(x,y)$ (x,y) within the triangle $[t_1,t_2,t_3]$

The weights must have certain properties. At the vertices, the weight of the corresponding influence function must one and the other two zero, otherwise the elevation of the target coordinate would not be preserved. Also, along the lines connecting each pair of vertices, the weight of the influence of the third vertex must be zero. That is required to ensure that correction along the adjacent edge of the adjopining triangle matches exactly, and cracking is thereby prevented.

The required properties are obtained by interpolating the weights linearly. Start by sorting the vertices in y order,



For $y > y_2$, compute the fractions of the distances down the left and right edges,

$w_{13} = (y_1 - y)/(y_1 - y_3)$, $w_{31} = 1 - w_{13}$

$w_{12} = (y_1 - y)/(y_1 - y_2)$, $w_{21} = 1 - w_{12}$

The fraction of the x-distance across the triangle is then computed from

$s_{13} = (y_1 - y)/(x_1 - x_3)$, $s_{12} = (y_1 - y)/(x_1 - x_2)$,
$x_{13} = (y_1 - y)/\ s_{13}$ $x_{12} = (y_1 - y)/\ s_{12}$

$u_1 = (x_{13} - x)/(x_{13} - x_{12})$ $u_2 = 1 - u_1$

Which yields the three weights,

$w_1 = w_{13}\ u_1$ $w_2 = w_{31}\ u_2$ $w_3 = w_{31}\ u_1$

for

$C(x,y) = w_1 f_1(x,y) + w_2 f_2(x,y) + w_3 f_3(x,y)$ (x,y) within the triangle $[t_1,t_2,t_3]$ and $y > y_2$

For the lower part of the triangle, $y \le y_2$

$w_{13} = (y_1 - y)/(y_1 - y_3)$, $w_{31} = 1 - w_{13}$

$w_{23} = (y_2 - y)/(y_2 - y_3)$, $w_{32} = 1 - w_{23}$

$s_{13} = (y_1 - y)/(x_1 - x_3)$, $s_{23} = (y_2 - y_3)/(x_2 - x_3)$,
$x_{13} = (y_1 - y)/\ s_{13}$ $x_{23} = (y_2 - y)/\ s_{23}$

$u_1 = (x_{13} - x)/(x_{13} - x_{23})$ $u_2 = 1 - u_1$

$w_1 = w_{13}\ u_1$ $w_2 = w_{23}\ u_2$ $w_3 = w_{32}\ u_2$

and

$C(x,y) = w_1 f_1(x,y) + w_2 f_2(x,y) + w_3 f_3(x,y)$ (x,y) within the triangle $[t_1,t_2,t_3]$ and $y \le y_2$

Special Cases

Above, we assumed that the three points were from different targets. If all three points are from the same target, the three influence functions should be set to one for that triangle. That will keep the terrain in flat facets so it will match the target, which we assume was modeled with polygons.

If two of the points are from the same target and the third from a different target, there is a potential for discontinuity near the target edge. To prevent cracking, use linear interpolation to obtain the correction function for points on the edge. Points in the interior of the triangle near the edge may have discontinuous values, but there won't be cracking. At worst, there would be a nearly-vertical wall near the edge of the target, matching the polygonal target to the approximately smooth terrain.

To complete the triangularization of the correction space we include the four corner points of the database. We could assign zero correction to those points, but in will provide

better continuity if a value is assigned that is reasonable for nearby corrections. We can select the three nearest target points near the corner vertex and assign the value of $f_1(x,y) + f_2(x,y) + f_3(x,y)$ to the correction. The sum of the three influence functions can then be used as the influence function for the corner point.

Creating New Vertices

After the correction function is defined it may be applied to each of the existing vertices in the original terrain. However, there may not be enough vertices in the terrain to accurately represent the smooth hills and valleys created by the influence functions. To fully modify the terrain, each of the original polygons, and any new polygons created by adding the targets, should be tested and subdivided if necessary.

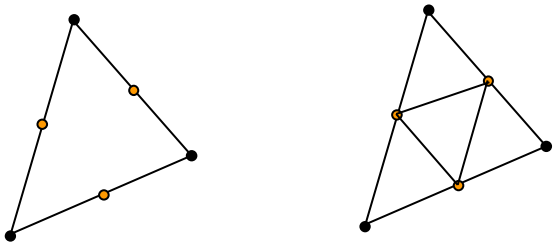
A test is performed on each edge of each triangle in the terrain database. The test is performed in the terrain database, **not** the triangularization of the correction function space. The test is to check if the existing straight edge is a good approximation to the underlying terrain, or whether a a vertex should be added to allow a closer fit to the intended terrain.

For each of the three edges of each terrain triangle, check if

$$| [C(x_i,y_i) - C(x_j,y_j)]/2 - C([x_i - x_j]/2, [y_i - y_j]/2) | < \varepsilon_T$$

where ε_T is a constant that determines the accuracy of the polygon fit, nominally 1.5 feet.

If any of the three edges fails the test, the terrain triangle must be subdivided into four triangles by connecting the midpoints of each edge. If none fail, the triangle is left alone.



If the triangle is subdivided, the process is then repeated on each of the four subtriangles. Eventually, the triangles will be small enough to adequately approximate the surface.

An Efficiency

The Gaussian curve used for the influence function falls to zero for large distances. If $d > 6k$, the value of the influence function can be taken to be zero. If there are only a relatively few targets in a large database, which generally is the case, then processing can be speeded up by excluding all the terrain that is outside of any influence function region.

One way to do this is to construct a square region of possible influence around each target. If the target is a point, then compute k for the point and define the target region as the square bounded by $(x_t - 6k) < x < (x_t + 6k)$ and $(y_t - 6k) < y < (y_t + 6k)$. If the target is a lineal or areal feature, then find the minimum x and y, the maximum x and y, and the maximum k for the set of vertices in the target. The boxed region potentially affected by the target is then $(x_{tmin} - 6k_{max}) < x < (x_{tmax} + 6\ k_{max})$ and $(y_{tmin} - 6\ k_{max}) < y < (y_{tmax} + 6\ k_{max})$.

The influence bounds can be computed and kept with a list of the targets. When an area block is read in, the area block boundaries can be tested against the target list and the targets found to potentially influence the block marked for use within the block. If nothing in the block can be modified, then all of the terrain processing can be skipped. If something in the block is potentially modified, then only the marked targets need be considered in the influence function calculations.

The influence regions can also be used for finer tests, such as on a cluster, object, or polygon basis.