DYNAMIC TERRAIN MODIFICATION USING A CORRECTION ALGORITHM

Roy Latham
President
Computer Graphics Systems Development Corporation

&

Donald Burns
Partner
Andes Computer Engineering

Abstract

The problem is to adjust the elevation in a terrain database for visual simulation so that the terrain surface 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 is to define a correction function $C(x,y)$ over the whole database 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 elevation to match the target.

Mission Rehearsal Requirements

Mission rehearsal is an application of simulator technology in which military aircrews practice a mission using the most recent information available for the battle space. New information may include aerial photographs and, our concern here, precise three-dimensional locations of targets and of other features relevant to the mission success. The new data may not agree with the previous data from which the terrain surface for the simulation data was derived. If uncorrected, the key features might be shown floating above or hidden below the terrain surface.

A traditional remedy has been to move the targets vertically to place them on the surface. This has the disadvantage that rehearsal simulation is not accurate with respect to the important target locations. The alternative is to adjust the terrain surface so it conforms to the target positions. The adjustments should be made so that there are no unnatural discontinuities in the terrain, water surfaces remain flat, and all non-target features remain on the surfaces.

Mission rehearsal is a time-critical application, and that complicates the terrain correction problem. Remaking an entire simulator database may take many hours; for mission rehearsal, a real-time solution is sought so that the simulation may be used without delay. Large simulator databases are divided into smaller rectangular or triangular *area blocks*. Real time software retrieves area blocks from mass storage as they come into view during the simulation. Real-time modification of the terrain can be implemented for each area block when it is retrieved. Modifying the area blocks upon retrieval avoids processing the portions of the database not used during the mission and allows overlapping the database modification time with mission rehearsal execution time. Upcoming blocks can be modified while previously-modified blocks are being displayed.

Making modifications block-by-block requires that the modifications be coordinated or decoupled across block boundaries. For example, an elevated target near a block boundary may require a gradual hill be created, with portions of the hill lying in the adjacent area block as well. Another case is that in which there are targets on either side of the area block boundary and the corrections must match at the boundaries. Even though targets are compact, a large area of terrain may be affected to ensure a smooth transition of the terrain contours to fit the targets. Theoretically, large correction to the elevation of a single target might cause the creation of a mountain or plateau that extends over many area blocks. More likely, a target in block A might cause changes in the adjacent block B, but block B could not be modified until further changes to a more distant block C were taken into account in B along with the changes due to A. The chain of interacting area blocks

Presented at the IMAGE 2006 Conference
Scottsdale, Arizona July 2006

could continue, encompassing a large fraction of the whole database and requiring substantial computation time to resolve.

Approach

The problem of interacting modifications could be solved by setting a limit to the scope of changes. We might require that changes in the terrain elevation be limited to a small fraction of the smallest dimension of an area block. However, an alternative approach allows arbitrarily large modifications in real time. Allowing large modifications avoids having to check if the bounding assumption is met, and it allows more general applications of the algorithm. Applications that are more general include the construction of a terrain database from specification of a relatively small number hill and valley elevation and the modeling of flexible surfaces for medical applications.

The more general algorithm relies upon a weaker assumption, that the number of specified target positions is small compared to the number of pre-determined elevation grid posts and features in the database. Contemporary terrain databases contain many millions of grids posts and features, so even there are thousands of target measurements the assumption remains valid.

The general algorithm defines a *correction function* globally at the start of the simulation. The correction function is the required change in terrain elevation at each point in the database. The correction function will ultimately be evaluated at every affected gridpost in each retrieved area block, but at the start of the simulation the function must only be defined for later evaluation. Because, by assumption, the number of targets is small relative to the size of the database, the definition process is negligible relative to the ordinary startup of the simulation. If the startup time were found to be significant, then the process of defining the correction function could be structured so that the area blocks needed first were in the part of the correction function defined first.

The role of the correction function is best understood relative to the concept of a *layered* database (Fig. 1). In a layered database, the components of the database are stored separately in categories and then assembled to make the final database. The categories of features may include aerial photographs used for surface textures, pre-determined terrain gridposts, surface objects such as trees and houses, a road network, lakes and rivers, light points, and perhaps other data unrelated to visual simulation, such as soil types. The layered approach facilitates porting the database among simulators having different capabilities, making modifications to the database, and generating compatible levels of detail. The final assembly of the layers may be done either prior to the simulation or in near-real-time during the simulation. Vehicle simulations involving large databases now commonly use the layered approach.

The correction function is effectively an additional layer in the database. The value of the correction function at each gridpost location is added to the evaluation of the gridpost. The terrain polygons are generated, and the surface objects, roads, lakes and rivers, and light points are added to the surface conventionally.

The method works as follows. The space of $C(x,y)$ is triangularized independent of the underlying database. A function corresponding to the shape of the smooth fit is defined independently for each target vertex. Nominally, a two-dimensional Gaussian function is used for each target vertex with the peak equal to the desired correction and the width proportional to the height, so for the *i*th target vertex, we define $f_i(x,y) = a_i * \exp[- d^2/k^2 a_i^2]$ where $d^2 = (x-x_i)^2 + (y-y_i)^2$ and k is a slope constant. For each point within each triangle of C, the value of the correction to be made at the point is a weighted sum of the three functions associated with respective vertices of the triangle, i.e. $C(x,y) = w_1 f_1(x,y) + w_2 f_2(x,y) + w_3 f_3(x,y)$. The weights sum to one and are inversely proportional to the distances to their respective vertices. C is defined over the whole database, but it is only evaluated at selected points in the underlying terrain database. The correction is first computed at the vertices of the terrain polygons, and then at the midpoints of the sides of the polygons. If the midpoints all lie within a small tolerance of the plane of the corrected triangle, the triangle is ready for display. If any of the midpoint corrections are

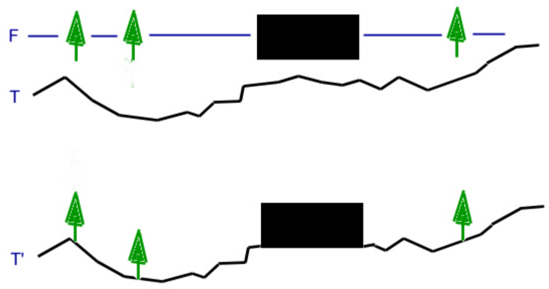


Fig. 1. In a popular database approach, a layer of features is placed using local terrain modifications

not within the tolerance, then the midpoints are connected to divide the original triangle into four new triangles. The procedure is iterated until the surface has been approximated smoothly.

An advantage of the correction function approach is that it lends itself well to application in real time. The correction function itself must be computed globally, but it is likely to be small in comparison to the terrain database, because the number of targets is likely to be small compared to the number of terrain polygons. The actual terrain modifications can be applied to each area block when it is retrieved. An additional advantage is that virtually any functional form can be associated with each vertex, so that variations in terrain roughness can be simulated. The method can be applied to terrain modifications made by bomb cratering or earthworks, as well as to target data.

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.

Traditionally, the approach has been to adjust the target data by changing the target elevations to match the terrain. However, for mission rehearsal applications this can lead to a less realistic situation in which the simulator does not correspond as well to the mission as it would if the target coordinates were preserved. Turn around times can be short for mission rehearsal, so it is better to apply the correction in real time or near-real-time rather than recompute the whole database.

Potentially even a small number of targets can affect large areas of the database. That is due to the need to modify surrounding terrain to smoothly integrate the targets into the terrain. In the extreme, one could imagine a single target point on a mountain peak requiring the whole area of the mountain to be adjusted to match.

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 targets 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 in Fig. 2. The triangularization is not unique. There are published triangularization algorithms, also known as meshing algorithms. [1,2]

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 ith 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 correct and unchangeable.

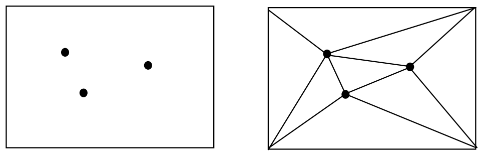


Fig. 2. As an example, three point targets appear in gaming area and result in the associated traingularization of the correction space.

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 an *influence function*

$f_i(x,y) = a_i * \exp[- d^2/k^2 a_i^2]$

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.

There is some theoretical justification for picking the two-dimensional Gaussian function, above, as the influence function. The slope at the apex is horizontal, which will match terrain flattened to fit buildings. The slope distance from the center also approaches horizontal, allowing the hill to disappear without discontinuity.

A different functional form may used for influence function, and each of the functions may be different if desired. Hills and valleys may be made more abrupt by raising the power to even numbers greater than two. More abrupt functions might be used in areas associated with rough terrain.

Interpolation

The value of the correction function within each triangle of the triangulated target database is determined as the weighted sum of the influence functions defined for the targets at the three vertices.

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 adjoining triangle matches exactly, and cracking is thereby prevented.

Given a triangle A,B,C and a point P within this triangle (Fig. 3), weights, w_1, w_2, w_3 are be computed as follows.

I_1 is a point which defines the intersection between lines <u>AP</u> and <u>BC</u>.
I_2 defines the intersection between lines <u>BP</u> and <u>CA</u>.
I_3 defines the intersection between lines <u>CP</u> and <u>AB</u>

w_1 = 1 - (length(A,P) / length(A,I1))
w_2 = 1 - (length(B,P) / length(B,I2))
w_3 = 1 - (length(C,P) / length(C,I3))

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 it 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

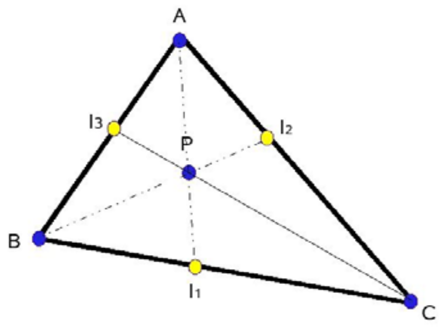


Fig.3 Interpolation to a point within a triangle uses the distance to the vertices

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 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

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

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 (Fig. 4). If none fail, the triangle is left alone.

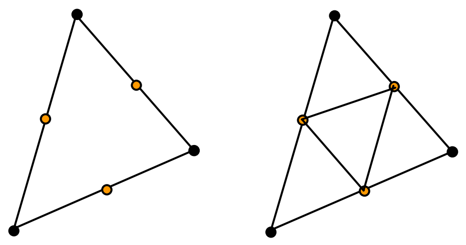


Fig. 4. Triangles are subdivided recursively to achieve the required fit.

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.

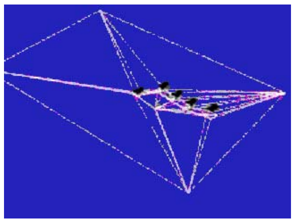
One way to do this is to construct a square region of possible influence around each target. If the target is a point, then compute k for the point and define the target region as the square bounded by $(x_t - 6k) < x < (x_t + 6k)$ and $(y_t - 6k) < y < (y_t + 6k)$. If the target is a lineal or areal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

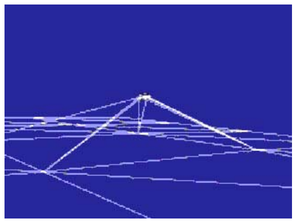
The influence bounds can be computed and kept with a list of the targets. When an area block is read in, the area block boundaries can be tested against the target list and the targets found to potentially influence the block can be marked for use within the block. If nothing in the block can be modified, 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.

Implementation

The dynamic terrain correction algorithms have been implemented experimentally in Open Scene Graph, an open-sourced real-time software system for simulation. The target locations are read in SHAPEfile format [3] and the unmodified database in OpenFlight format. Steps in the creation of the correction function are shown in wireframe in Fig. 5. The corrections are made artificially large for illustration.

a. 

b. 

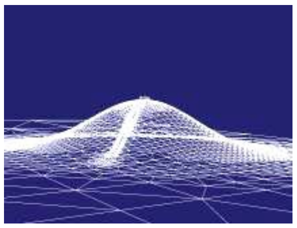
c. 

Fig. 5. Steps in creating the correction function include triangulating the database (a), applying the raw corrections to the target positions (b), and smoothing the corrections with the influence functions (c).

The rendered database with mismatched targets is shown in Fig. 6a, and with corrected terrain in Fig. 6b.

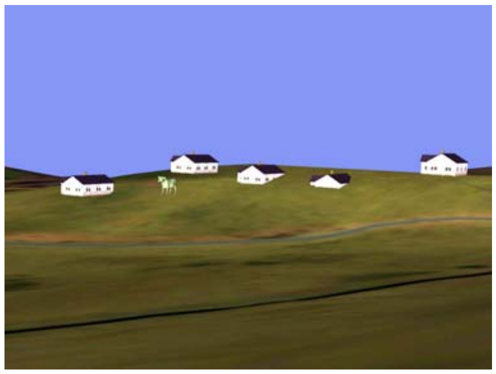


Fig. 6a. Example of targets mismatched to terrain.

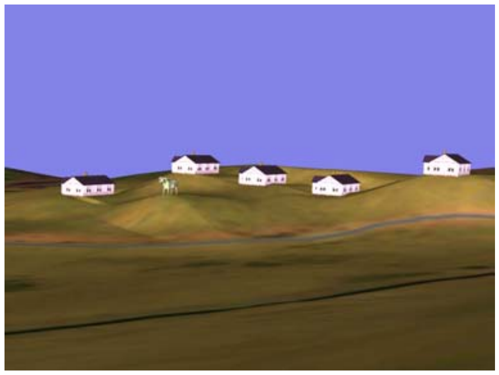


Fig. 6b. The terrain corrected dynamically using the algorithms described.

Conclusions

The requirements of terrain modification for mission rehearsal can be met by using layered approach to database construction in which correction function is included as an additional layer. The correction function embodies all the changes that must be made to the terrain elevations to make the terrain surface conform to the most current measured data. The correction function is defined for the whole database at the start of the mission, but it is not applied until required area are retrieved. This approach minimizes the delay before the mission rehearsal can begin.

Acknowledgement

The work described in this paper was supported by the U.S. Air Force under Small Business Innovation Research Phase I and Phase II contracts, topic AF05-064. The opinions expressed herein are entirely those of the authors.

References

1. A well developed software set is given at http://www.cs.cmu.edu/~quake/triangle.html

2. A survey of literature on the subject is given at http://www.andrew.cmu.edu/user/sowen/mesh.html

3. *Shapefile Technical Description*, Environmental Systems Research Institute, Inc. , 1998.