Experimental Verification of a Realistic Input Model for Polyhedral Terrains

Esther Moet

Department of Information and Computing Sciences, Utrecht University

Technical Report UU-CS-2007-052

www.cs.uu.nl

ISSN: 0924-3275

Experimental Verification of a Realistic Input Model for Polyhedral Terrains

Esther Moet

December 19, 2007

Abstract

A realistic input model for polyhedral terrains was introduced in [16]. With this model, improved upper and lower bounds on the combinatorial complexity of visibility and distance structures on these terrains were obtained. In this paper, we perform experiments with real-world data to determine how realistic the assumptions of that input model are, by examining what the values of the model parameters are in practice. We consider the results of these experiments as evidence that the assumptions of the model are probably realistic: they are all satisfied by the terrains that we generated from GIS data. Moreover, we evaluate the complexity of the visibility map in these terrains by counting vertices of the visibility maps for different viewpoints.

1 Introduction

For many geometric problems, the theoretically proven worst-case computation and combinatorial complexity bounds are higher than the running time and storage that are observed in practical applications. Realistic input models are a way to explain this discrepancy between theory and practice. By describing certain criteria which the input has to satisfy, it is possible to show better bounds; these improved bounds are usually closer, or even equal, to the real-world complexities.

Polyhedral terrains are used extensively in GIS, computer graphics, and computational geometry, since they provide a way to approximate parts of the surface of the earth. In principle, they are triangulations in which each vertex has a height coordinate associated with it. Since for every x- and y-coordinate, there is only one z-coordinate such that the point (x, y, z) belongs to the terrain, they are sometimes considered to be 2.5-dimensional.

Many well-known two- and three-dimensional data structures can also be defined for the special case that a polyhedral terrain is the input, where the metric is the distance travelled over the (triangles of the) terrain. For example, one can consider the Voronoi diagram of a set of points on a terrain, or the shortest path map on a terrain for a single source, or even for multiple sources. For GIS applications, the visibility map has been studied extensively. It is the three-dimensional variant of the two-dimensional visibility polygon or diagram. Since a polyhedral terrain can be considered as a hybrid between 2D and 3D, we expect the complexities of algorithms and data structures for terrains to be worse than in 2D but better than in 3D. However, for the visibility map for instance, the worst-case complexity bound is the same for a terrain as it is for general 3D scenes, namely $\Theta(n^2)$, where n is the number of triangles of the terrain. In spite of that, the worst-case scenario hardly ever occurs in practical situations, and thus the complexity that is observed in applications rarely is (close to) quadratic. Therefore, the discrepancy between theory and practice also exists for polyhedral terrains.

In [16], a realistic input model for polyhedral terrains was introduced. It put forward several assumptions that allowed for improved, and supposedly more realistic, complexity bounds for three different structures on terrains. The goal of this paper is to check to what extent these assumptions, and the bounds they lead to, are indeed realistic.

We perform several experiments to determine how realistic the assumptions of the input model from [16] are. The values of the four input model parameters α, β, c, and d are evaluated for different test scenes of varying size (i.e., number n of triangles), representing landscapes in the US. We observe that these experimental values do not depend on n, which is what is assumed in the input model. This is evidence that the assumptions are indeed realistic. In our experiments, we also determine the complexity of the visibility map for different viewpoints in the test scenes to observe what this complexity is in practice.

The structure of this paper is as follows. In Section 2, we give details on the setup of our experiments. Then we repeat the input model assumptions. In Section 4, for each of the input model parameter, we discuss the method of computation and present the results we have obtained. Finally, in Section 5, we present the complexity of the visibility map that we observed in practice.

2 Experimental Setup

Verifying whether an input model is realistic starts with *reality*, that is, input data from practice. In this case, polyhedral terrains are supposed to model landscapes, or in other words, parts of the earth. The real-life data comes from Geographical Information Systems (GIS). In this section, we give more details of the raw data that we used, how we acquired it, and how we preprocessed it to use in our experiments.

2.1 DEM data

In *remote sensing*, the elevation of the surface is measured at sample points that lie on a regular grid, which yields a (two-dimensional) matrix with height values. Such a matrix is usually called a Digital Elevation Model (DEM). The DEM-data that we use in our experiments was downloaded for free from `http://www.mapmart.com/`, the GIS website maintained by IntraSearch Inc. The data was originally collected by the U.S. Geological Survey (USGS). The areas that we analyze all lie in either Colorado or Nevada; the data set consists of 16 scenes and contains both urban and mountainous landscapes.

The downloaded data is set in the Spatial Data Transfer Standard format (SDTS). The cellsize in all DEMs is 30 meter. We convert these SDTS DEMs to the more intuitive and compact ARC/info ASCII Grid format; several utilities that provide this procedure can be downloaded from the web. The syntax of the header of this file format is shown in Table 1; every line starts with a (textual) identifier, followed by a numerical value corresponding to that identifier. After this header, the actual grid data is given, in **nrow** rows of **ncol** numerical (height) values. The height value is missing for some coordinates; there is an indicator for these grid points, which usually is the value -9999.

`ncols` ncol	(Number of columns in the grid)
`nrows` nrow	(Number of rows in the grid)
`xllcorner` x	(Lower left x-coordinate of the grid)
`yllcorner` y	(Lower left y-coordinate of the grid)
`cellsize` size	(Grid cell size)
`NODATA_value` NODATA	(Value indicating an empty grid cell)

Table 1: The header of the ARC/info ASCII Grid file format.

Each DEM consists of around 160,000 height values, corresponding to a grid of approximately 360 rows and 470 columns. If we let the rows correspond with x-coordinates and the columns with y-coordinates, then the height values are the corresponding z-values. While reading the input data we determine the range of z-coordinates, to be able to scale all 3D-coordinates to smaller values that fall into the range of screen-coordinates. This so-called z-range also gives an indication of how mountainous the scene is. The characteristics of all sixteen DEM files are given in Table 2.

Name	State	#rows	#cols	#points	z-range
Antelope Peak	NV	362	465	168330	[1887,3184] (1297)
Arvada	CO	357	462	164934	[1572,1729] (157)
Aspen	CO	370	470	166494	[7415,12218] (4803)
Castle Peak	CO	367	470	165022	[2024, 3440] (1416)
Coffin Mountain	NV	359	466	163697	[1577, 2486] (909)
Commerce City	CO	357	462	164927	[1536, 1667] (131)
Crooked Creek	CO	368	469	165892	[2484, 3672] (1188)
Eagle	CO	366	469	165269	[1959, 3196] (1237)
Golden	CO	358	468	165016	[5389, 7070] (1681)
Grouse Mountain	CO	366	469	165576	[2268, 3977] (1709)
Lava Creek	CO	365	468	164966	[2119,3502] (1384)
Mount of the Holy Cross	CO	366	468	165857	[2660, 4266] (1606)
Mount Jackson	CO	366	468	165821	[2870,4152] (1293)
Red Cliff	CO	365	467	165569	[2611, 3651] (1040)
Red Creek	CO	367	469	165857	[2207, 3664] (1457)
Suicide Mountain	CO	367	470	165601	[1962, 3418] (1456)

Table 2: DEM characteristics for the 16 scenes which are used in the experiments.

2.2 TIN generation

To obtain a continuous input data set from the discrete input, it is customary in GIS to construct a Triangular Irregular Network (TIN) that represents such a DEM as good as possible; see Figure 1. The elevations at (some of the) grid points give the vertices of the TIN, and the triangles give interpolated height values at all other x- and y-coordinates. A TIN is the same thing as what is called a polyhedral terrain in computational geometry. In this section, we describe how we obtain polyhedral terrains from the input data. We evaluate the input model parameters for these generated terrains in Section 4.

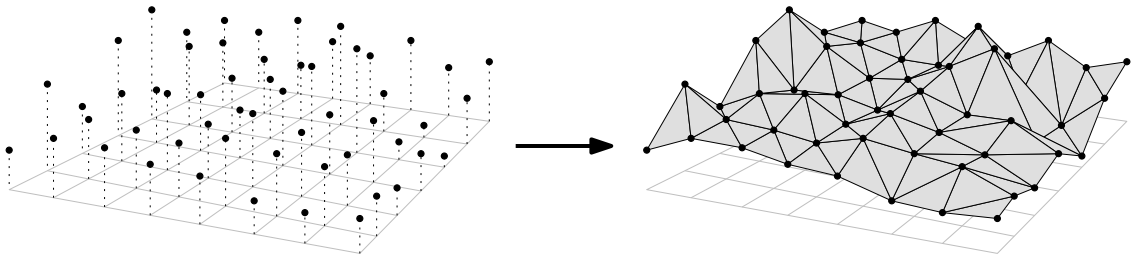

Figure 1: A DEM (left) and a TIN that represents this DEM (right).

Besides the fact that elevation data is often missing at a number of grid points, it is common that not all sample points of the DEM are inserted as vertices into the TIN, as is the case in Figure 1. TINs generally are denser in areas where the elevation is more variable, and sparser in flat areas. Thus, TINs help to reduce storage space and computation time while they still preserve the most important characteristics of the input. Obviously, by selecting a subset of the DEM grid points, some data loss occurs, but this is usually acceptable since DEMs are already approximations of the real world.

2.2.1 Related work

There are many different approaches to construct a triangulated irregular network (TIN) from a digital elevation model (DEM). Two groups of methods can be distinguished: namely, *surface simplification* and *surface refinement*.

Algorithms in the first group start with the original surface and simplify it until the number of vertices is small enough, while the error of approximation is still acceptable. This type of method inherently also provides a multi-resolution hierarchy of approximating surfaces. Most succesfull simplification methods are based on *iterative edge contraction* [11, 13], or the contraction of (not necessarily incident) vertex pairs [8]. Another simplification method is the so-called *drop heuristic* [12], in which DEM points are successively removed, based on significance, until a certain error level is reached. More details on the various polygonal surface simplification algorithms can be found in two surveys [9, 14].

Surface refinement algorithms start with a very coarse surface and then increase the resolution until the desired level of approximation is obtained. These refinement methods are sometimes called *iterative vertex insertion methods*, and vary based on the selection criterium that determines which DEM grid points are inserted as vertices of the TIN. As the name suggest, iterative vertex insertion methods work iteratively, which means that while a subset of the vertices is being selected, a triangulation (of the convex hull) of the selected vertices so far is maintained. The most common choice is the Delaunay triangulation because of several desirable properties, such as maximizing the minimum angle. One of the first and quite succesfull methods was presented by Fowler and Little [6]. They describe a global procedure that identifies features of the surface such as peaks and ridges, and selects the vertices on these features for insertion into the TIN. The very important point (VIP) method is another well-known vertex selection method. It is a local procedure that determines the significance of a point relative to its 8-connected neighborhood [2]. De Berg et al. use this method in their work on realistic input models for geometric algorithms [1].

2.2.2 Our method

We use a different, intuitive approach called the *greedy insertion method*, which is a surface refinement algorithm, and which has been described in a number of different fields, such as GIS [10], computer graphics [7], computer-aided design (CAD) [4], and computational geometry [5]. The vertex that is inserted in each step, is the grid point that defines the maximum approximation error, i.e., the DEM element that has the greatest deviation from the current approximating terrain. Originally, one would iterate until a given level of error was reached, but since we want to obtain a TIN with a specific number of triangles, we iterate until that given size is reached.

For each scene, we create thirteen TINs with increasing *resolution*, i.e., a growing number of triangles n. We let n range over [500,20000]. For completeness, we also create the TIN with all DEM grid points; we call this TIN the *full terrain*. The data structure that we use to store the TIN is a two-dimensional Delaunay triangulation, which is implemented by `CGAL` [3]. We adapt this data structure in a straightforward way so that every vertex of the 2D triangulation stores a height value. In this way, the projection of the TIN is always a Delaunay triangulation. When we insert a new (3D) grid point into the terrain, we insert the xy-projection of this point into the (2D) triangulation, while storing the z-coordinate in the resulting vertex, after which the `CGAL` implementation ensures that the (2D) Delaunay property is automatically restored.

The first step in the TIN generation algorithm is the initialization of the TIN. At the boundary of the DEM, input data may be missing for some grid points. These grid cells are flagged by the `NODATA`-indicator. Since we want to create a rectangular TIN, which is common in GIS, we find a (maximal) subgrid such that the four corner points all do contain elevation data. We insert these four corner points into an initially empty Delaunay triangulation.

For a given value n, we now want to create a TIN that consists of n triangles, by inserting the *most deviating* grid points iteratively. Which grid point deviates the most from the 'current' terrain is determined in the following way. For each grid point (x, y, z), we locate the triangle that lies above resp. below it. For this triangle, we compute the supporting plane, which we use to compute the interpolated height value z' for (x, y). We compute the deviation $|z' - z|$, and find the maximum deviation over all grid points. This point is inserted as a new vertex into the Delaunay triangulation. This process is repeated until the number of triangles of the TIN is at least n (it can be slightly more). In Figure 2, four resulting TINs for the scene Antelope Peak (NV) are shown, for the values $n = 1000, 4000, 12000, 20000$. In Appendix A, the TINs with the

same values of n are displayed for the remaining fifteen scenes.

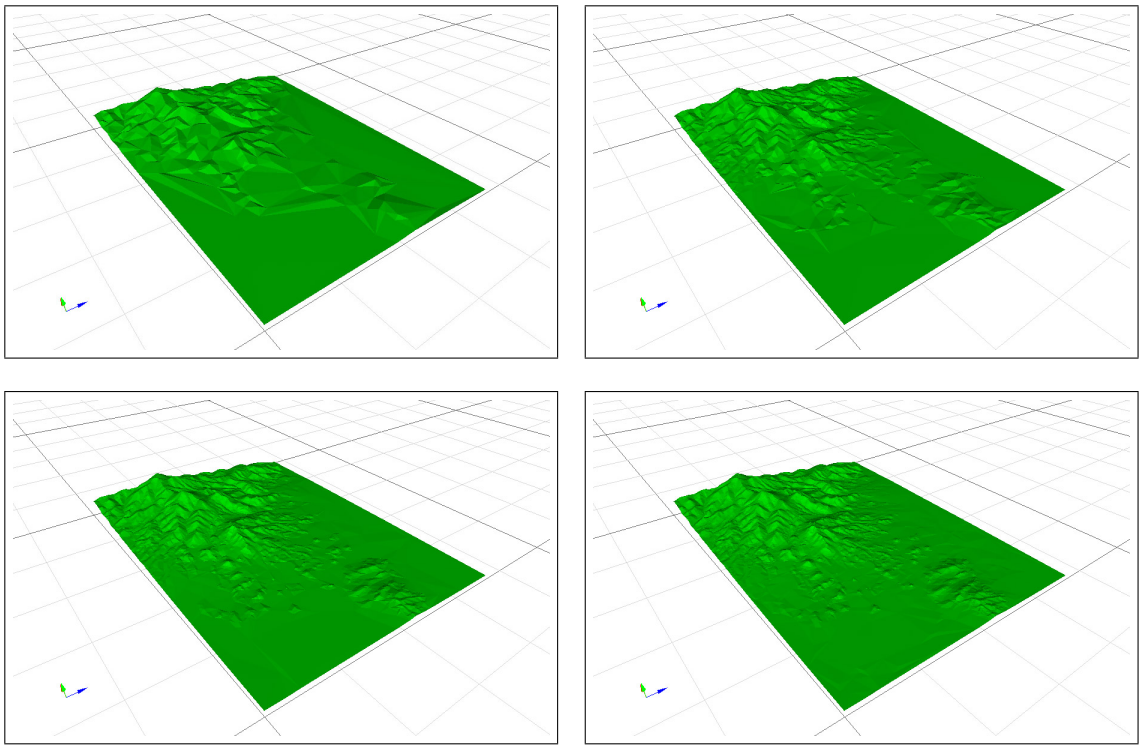

Figure 2: Antelope Peak (NV).

2.2.3 Running time analysis

Let N be the number of grid points, and let n be the number of triangles in the resulting TIN. Filling a two-dimensional grid with the DEM values and scaling all coordinates takes $O(N)$ time. The initialization takes $O(N)$ time in the worst case, but will on average take $O(1)$ time. We do not know how efficiently the point location method of the `CGAL`-Delaunay triangulation data structure is implemented, but in the worst case it takes $O(n_i)$ time, if the (intermediate) TIN has n_i triangles. Thus, the time complexity for inserting vertices until the TIN has n triangles is at most $\sum_{4 \leq n_i \leq n} O(n_i N)$. This yields an overall time complexity of $O(n^2 N)$. The whole TIN generation algorithm, and in particular the point location method, can be implemented in a much smarter, and thus more efficient, way. A typical running time for an implementation of the greedy vertex insertion algorithm is $O(Nn)$ or $O(N \log n + n^2)$, whereas the optimized algorithm by Garland and Heckbert typically takes $O((N + n) \log n)$ time [7]. However, since in this context TIN generation is only a preprocessing step, we did not attempt to optimize our implementation.

3 The Model

We use the same definitions and notation as in [16]. Let $\mathcal{T}$ be a polyhedral terrain: a piecewise-linear continuous function defined over the triangles of a triangulation in the xy-plane. A terrain $\mathcal{T}$ comprises a set T of n triangles, a set E of n_e edges, and a set V of n_v vertices, where n_e and n_v are $O(n)$.

The four assumptions of the input model are given below; assumptions 1 to 3 refer to the xy-projections of T and E, whereas the last assumption refers to the original triangles in $\mathbb{R}^3$. The restriction that we put on the values α, c, d, and β in the assumptions below is that they all are positive constants, i.e., constants strictly greater than zero.

1. **Fatness**: the minimum angle of any triangle in T is at least α.
2. **Aspect ratio**: the smallest rectangle that contains T has side lengths 1 and c.
3. **Edge length ratio**: the longest edge in E is at most d times as long as the shortest one.
4. **Dihedral angle**: the dihedral angle of the supporting plane of any triangle in T with the xy-plane is at most β, where $\beta < \frac{\pi}{2}$.

In [16], the fatness assumption is replaced by low-density. Fatness is a stronger restriction than low-density, i.e., the number of terrains that satisfy low-density is larger than those that have only fat triangles. In theoretical respect, we prefer to work with low-density, since this is a more general assumption and thus yields stronger results. However, in this empirical context, we study fatness. Terrains with fat triangles also obey the low-density property. In particular, when you show that for terrains in practice, fatness is a realistic assumption, this implies that low-density also is realistic.

4 Evaluation of the Model Parameters

In this section, we determine the values of all four model parameters α, β, c, and d for all sixteen input scenes, and evaluate whether or not the assumption that they are all constants is realistic.

4.1 Aspect ratio

The parameter c can be considered as the aspect ratio of the terrain domain. We treat this parameter separately and briefly, since it differs significantly in nature from the other ones. That is to say, the aspect ratio parameter refers to the terrain as a single entity, whereas the other three model parameters view the terrain as a set of triangles.

The aspect ratio of the terrain is equal to the aspect ratio of the input domain. This ratio depends on the number of rows and columns of the grid that contains the DEM data, and is therefore independent of the number of triangles n, which is chosen later on. Of course, this ratio can change when we select a subgrid of the original DEM to construct the TIN from. Still, for all our scenes, the parameter c is a constant with respect to n. In our experiments, all values of c lie in the range $[1.287, 1.315]$. All aspect ratios of our sixteen scenes are given in Table 3. First, we give the aspect ratio of the full DEM, and then the aspect ratio of the selected subgrid out of which the TINs are generated. Since this parameter is equal for all values of n, the second assumption of the input model from [16] is satisfied by all our terrains from practice.

For the other three parameters α, β and d, the number of triangles can be of influence, so we treat them in more detail in the sections below.

4.2 Fatness

In this section, we determine the value of the fatness of triangulations in practice.

We refer to the xy-projection of the terrain, i.e., to a two-dimensional triangulation. For every triangle, we compute the angle at each of the three vertices. We take the two (normalized) vectors, pointing out of the vertex along the two incident edges, and take the arccosine of their dot product. The minimum of the three angles gives the fatness of the triangle. Finally, we compute the minimum over all triangles; this is the fatness of the triangulation. The input model states that this quantity should be bounded from below by a constant $\alpha > 0$.

4.2.1 Results

In Figure 3, the graphs with the results of our experiments are displayed. We present four graphs; in each graph we group the results of four scenes. Recall that for each scene, we have thirteen terrains, with a growing number of triangles n. Thus, we can set out the fatness values for these terrains against n. Since n is much larger for the full terrain, the values for these terrains are not

Scene	State	DEM a.r.	subgrid a.r.
Antelope Peak	NV	1.285	1.292
Arvada	CO	1.295	1.296
Aspen	CO	1.271	1.287
Castle Peak	CO	1.281	1.298
Coffin Mountain	NV	1.299	1.308
Commerce City	CO	1.295	1.297
Crooked Creek	CO	1.275	1.289
Eagle	CO	1.282	1.298
Golden	CO	1.304	1.315
Grouse Mountain	CO	1.282	1.296
Lava Creek	CO	1.283	1.297
Mount of the Holy Cross	CO	1.279	1.292
Mount Jackson	CO	1.279	1.293
Red Cliff	CO	1.280	1.291
Red Creek	CO	1.279	1.294
Suicide Mountain	CO	1.281	1.297

Table 3: The aspect ratio of the 16 scenes.

included in these graphs. The fatness of each of these full terrains lies in the range $[2.796, 18.294]$ with an average of 15.358.

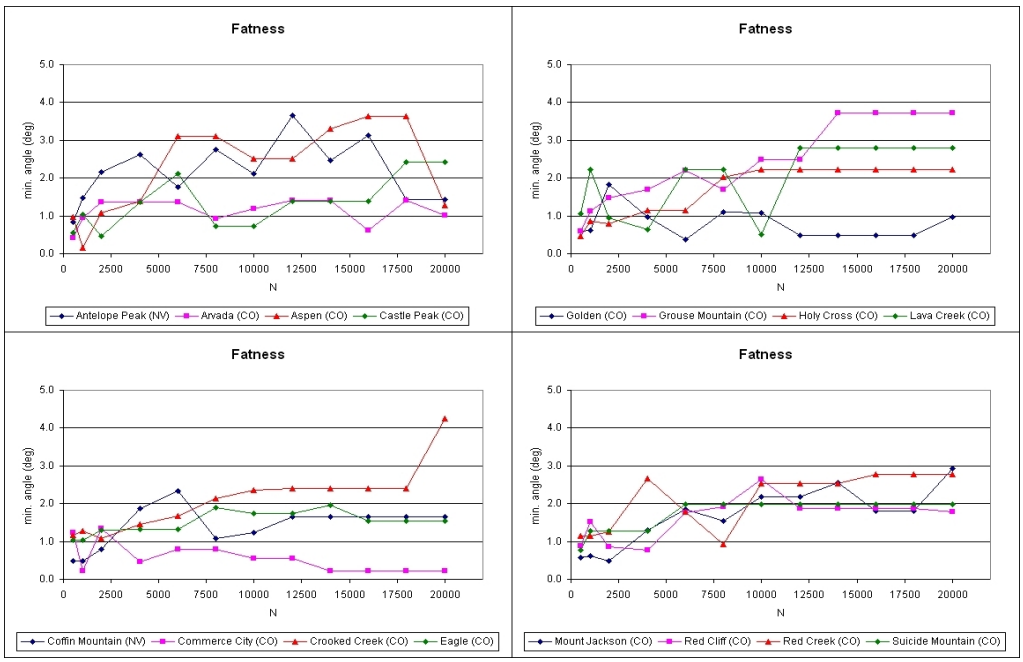

Figure 3: Fatness as a function of n.

4.2.2 Conclusion

The fatness of a triangulation of the vertices of a entirely filled regular grid is 45 degrees; all triangles in such a triangulation are right isosceles triangles with side lenghts 1, 1, and $\sqrt{2}$. In practice some input data will be missing, and thus this optimal 45 degrees will generally not be attained.

The input assumption states that the minimum angle over all triangles should never be arbitrarily close to zero. It seems this will not the case. In the majority of the graphs the fatness remains the same when n grows, or it even increases. For five scenes, the main tendency of the graph is not increasing: Antelope Peak (NV), Arvada (CO), Aspen (CO), Commerce City (CO), and Red Cliff (CO). But even for these scenes, the fatness for the maximum n (i.e., the full terrains) is still bounded away from zero by a constant $\alpha > 0$, namely 18.294, 15.101, 18.225, 15.101, and 2.796 respectively.

We observe that, for our input set, the smallest angle in a TIN does not become arbitrarily small if n becomes very large. This means that the first assumption of the input model of [16] is satisfied in our terrains from practice, and thus we call it realistic for this input set.

4.3 Edge length ratio

Next, we evaluate which values are realistic for the ratio between the length of the longest and the shortest edge of a triangulation.

Computing the edge length ratio of a triangulation is very straightforward; you iterate over all edges, and maintain the maximum and the minimum length that you encounter. Finally, you divide these two quantities. The input model states that this ratio should be bounded from above by a constant $d > 0$.

4.3.1 Results

In Figure 4, the graphs with the results of our experiments are displayed. As in the fatness graphs, the full terrains have not been included here. The edge length ratios for these full terrains lie in the range $[3.171, 8.000]$ with an average of 4.363.

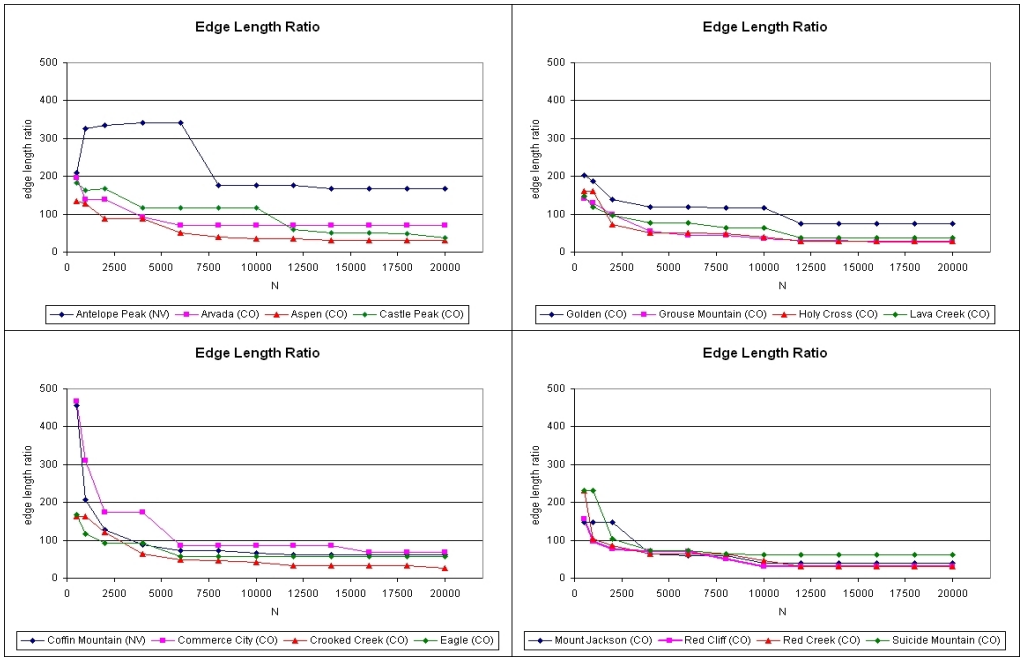

Figure 4: Edge length ratio as a function of n.

4.3.2 Conclusion

In a triangulation of the vertices of a entirely filled regular grid, the maximum edge length is $\sqrt{2}$ and the minimum is 1, which yields an edge length ratio of $\sqrt{2}$. This optimal value is the same

for *all* scenes created from an entirely filled DEM input.

If we set out the edge length ratio d against n, we see that, generally, d decreases if n increases. For most scenes, the rate at which d decreases initially is high, but becomes smaller for larger n. Note that even though the edge length ratio itself is not a constant with respect to n, it is (asymptotically) bounded from above by a constant d, which is exactly what the third input model assumption states. Note that grid data may be missing from a DEM and thus the edge length ratio will most likely not attain its optimal value of $\sqrt{2}$. Hence, we observe that the third input model assumption is satisfied by our terrains as well, and we call it realistic for this input set.

4.4 Dihedral angle

Finally, we determine whether or not the fourth assumption of the input model of [16] is satisfied by terrains from practice.

For every triangle of the TIN, we compute the supporting plane; this is the plane of which we need to compute the dihedral angle it makes with the xy-plane. Then we compute a unit normal vector for this plane. To compute the dihedral angle, we take the arccosine of the dot product of this normal vector with the vector $(0, 0, 1)$, which is the unit normal vector of the xy-plane. Finally, we take the maximum over all dihedral angles. The input model states that this quantity should be bounded from above by a constant β which is strictly smaller than $\frac{\pi}{2}$.

4.4.1 Results

In Figure 5, the graphs with the results of our experiments are displayed. The maximum dihedral angles for the full terrains lie in the range $[31.815, 80.067]$ with an average of 62.304.

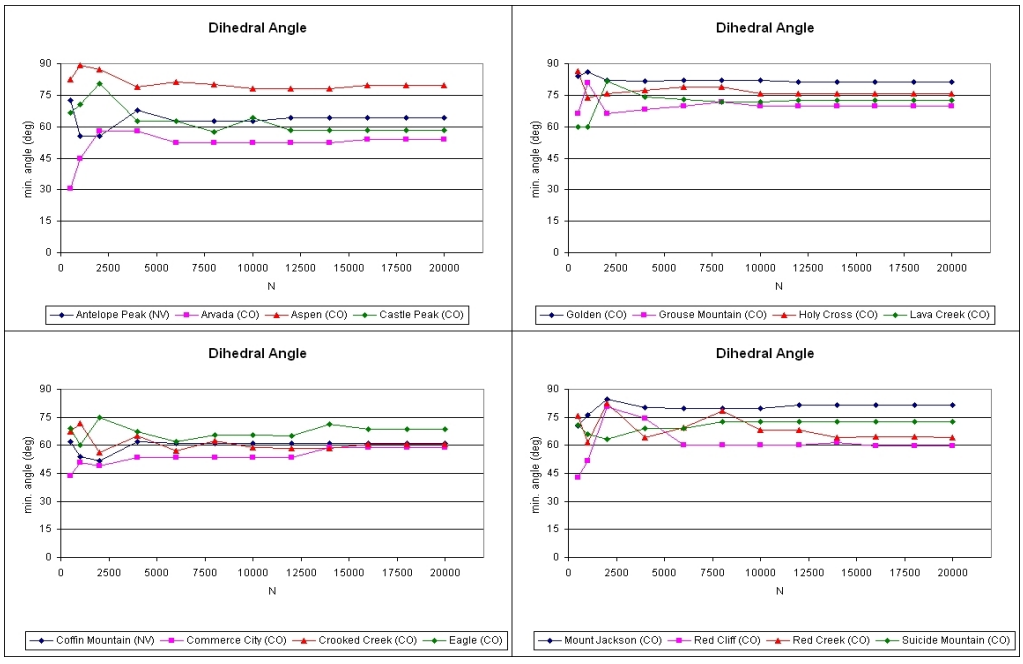

Figure 5: Dihedral angle as a function of n.

4.4.2 Conclusion

The graphs are very clear; for our input set, the dihedral angle does not become arbitrarily close to $\frac{\pi}{2}$ when n becomes very large, i.e., the triangles of the TIN will not become arbitrarily steep. In particular, the maximum value of the dihedral angle depends on the maximum height difference

of adjacent DEM grid points. It could be larger for smaller n, since the global maximum and the global minimum of the DEM could be incident vertices in a very coarse TIN (they probably even will when the greedy vertex insertion method is used). But even then, in our terrains, there is a natural upper bound on the value of the dihedral angle. Therefore, we conclude that also the fourth assumption of the input model is satisfied by our terrains, and we call it realistic for this input set.

5 Complexity of the Visibility Map

Besides verifying whether the input model is realistic, we also check whether the results that are obtained with this model in [16] conform with reality. In particular, we determine what the typical complexity is of the visibility map (VM) in terrains in practice.

The visibility map of a viewpoint on a terrain is the subdivision of that terrain into maximal connected components such that throughout each cell the viewpoint p is either visible or invisible. It has worst-case complexity $\Theta(n^2)$[15].

To analyze the complexity of the VM in practice, we compute the complexity of the transparent VM for our sixteen scenes, and for three different viewpoints. We pick these viewpoints such that they differ significantly in location: one is located very far away from the terrain, one close to the boundary of the terrain, and one right in the middle of the terrain. The first viewpoint should yield a VM similar to one of a viewpoint at infinity; the height at which the viewpoint is located is almost irrelevant in this case. We place the second viewpoint at the height that is halfway in the range of z-values of the vertices of the terrain. The third viewpoint is placed a little bit higher, at two-thirds in the range of z-values. We chose these three viewpoints in this way in an attempt to achieve that any other possible viewpoint will have a VM that resembles the VM of one of these three viewpoints. This is not the case for viewpoints that lie far above or below the terrain, but for such viewpoints, basically the entire terrain is visible, and thus the complexity is $O(n)$, which makes it unnecessary to study these cases any further.

Now, we first describe how we determine the complexity of the VM, and then we present the results that we obtained.

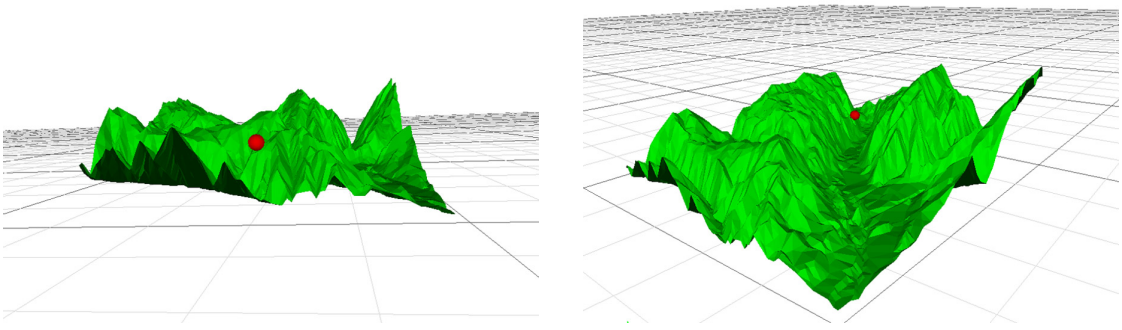

Figure 6: The second (left) and third (right) viewpoint for the scene Aspen (CO) with $n = 10000$.

5.1 Computation

We compute the complexity of the VM by counting its vertices. A vertex of the visibility map of a point p directly corresponds to a line through p that is a common tangent of two terrain edges, see [16]. When we assume that the triangles of the terrain do not obstruct visibility, the VM becomes the *transparent visibility map*, which generally has higher complexity than the so-called *occluded visibility map*. We count the vertices of the transparent VM.

At the start of the algorithm, we initialize the VM complexity for the viewpoint p to be the number of vertices n_v, so that we don't have to count all incident edge pairs or coplanar edges.

Our implementation iterates over all unique pairs of edges, and determines for each such pair whether or not they contribute a vertex, i.e., whether they have a common tangent. For two edges e_1 and e_2, we first construct the planes P_1 and P_2, that are defined by p and e_1 and p and e_2, respectively. The intersection of these two planes generally is a line l — if e_1 and e_2 are coplanar, P_1 and P_2 are equal. Note that P_1 and P_2 cannot be disjoint, since p is a common point. This line l does not necessarily intersect the two edges; it intersects both e_1 and e_2 if and only if they induce a transparent VM vertex.

Now we determine whether e_1 intersects P_1 and e_2 intersects P_2; if not, we finish for this edge pair and do not increase the complexity. If they do intersect, we switch to 2D and determine whether the 2D projections of e_1 and e_2 are consecutively intersected by the projection of l, i.e., whether they intersect l on the same side of p. Of course, they do not define a vertex of the VM of p if p has to view in different directions to see them. If this test also evaluates to true, then we increase the complexity by one and move on to the next edge pair. For a terrain with n triangles and $n_e = O(n)$ edges, we check $O(n^2)$ edge pairs, and thus the algorithm described above takes $O(n^2)$ time.

5.2 Results

In this section, we present the results of this vertex counting algorithm for the three different viewpoints. In Figure 7, the complexities of the VMs in all sixteens scenes, and for the three different viewpoints, are displayed as functions of n. They are all quite similar, and seem to grow just a little bit faster than linearly. Since the algorithm takes quadratic time and the full terrains have roughly 160,000 vertices, we have not included these terrains in the computations. Recall that in [16], it was shown that the complexity of the visibility map for a terrain that satisfies the input model has complexity $\Theta(n\sqrt{n})$ in the worst case.

To get a better idea of the growth rate of these complexities, we present graphs of the *approximated derivative* of the complexity for the first viewpoint. The other two viewpoints give similar results. This approximated derivative was computed discretely. Let c_{n_i} be the computed complexity for the number of triangles n_i. We approximate the derivate at this point with the central difference quotient:

$$\frac{1}{2}\frac{(c_{n_{i+1}} - c_{n_i})}{(n_{i+1} - n_i)} + \frac{1}{2}\frac{(c_{n_i} - c_{n_{i-1}})}{(n_i - n_{i-1})}.$$

We approximate the second derivative in the same way, from the values of the first derivative. The results are shown in Figures 8 and 9.

All graphs in Figure 8 show an increasing growth rate, which means that in this case the VM complexity grows more than linearly with n. The growth rate for the scene Commerce City (CO) increases the fastest. Its graph of the approximated second derivative even starts growing again at the end. This scene has the following parameters: for n =16000, 18000, and 20000, it has fatness 0.213, edge length ratio 68 and dihedral angle 58.63. In other words, it satisfies the input model and we call it a realistic terrain. In this way, we obtain evidence that the visibility map in a realistic terrain can indeed have superlinear complexity, which was theoretically shown in [16]. It has complexity $\Omega(n)$ by definition, and these graphs give the idea that it may be unlikely to find a visibility map with a complexity of $\omega(n\sqrt{n})$ in a realistic terrain.

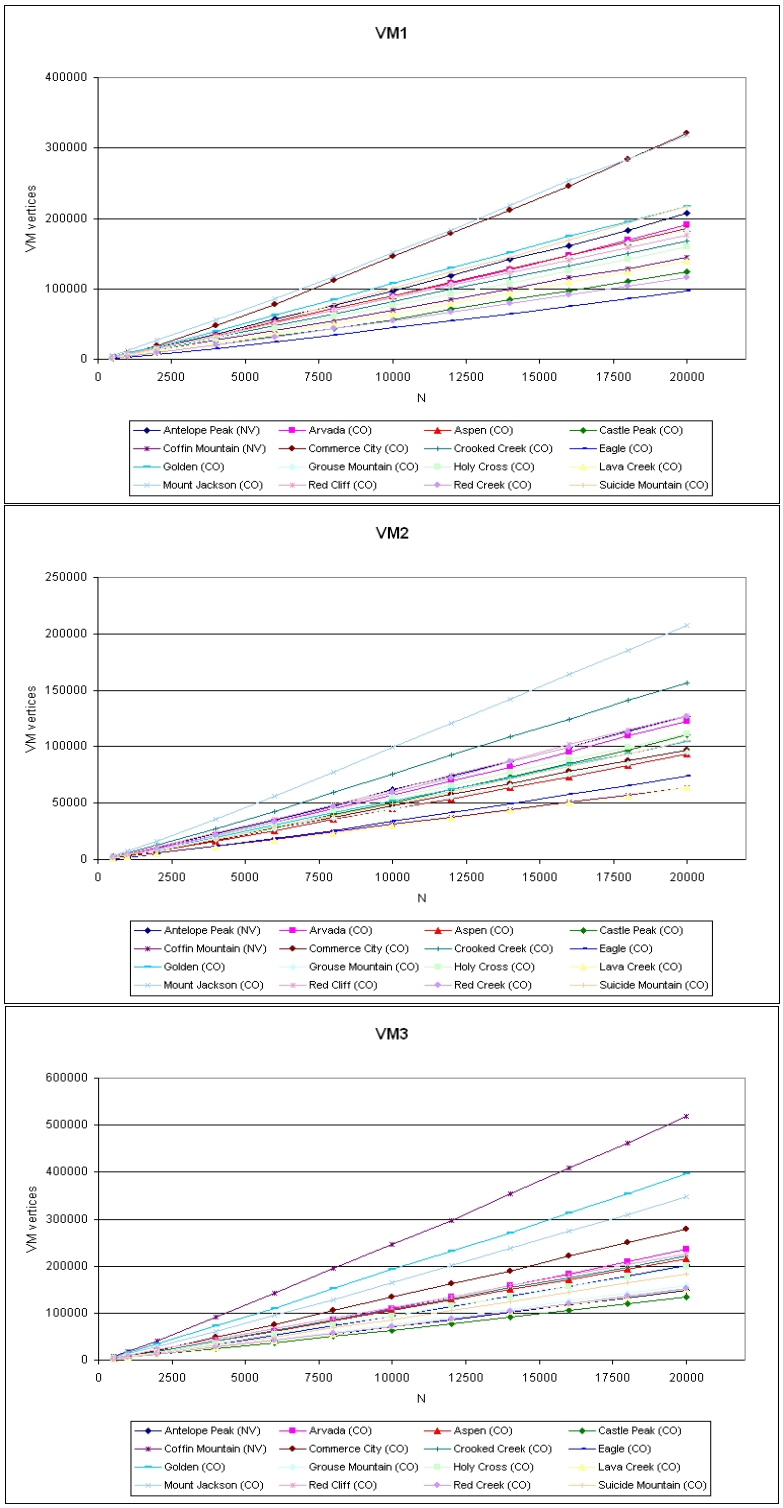

Figure 7: Complexities of the three visibility maps as functions of n.

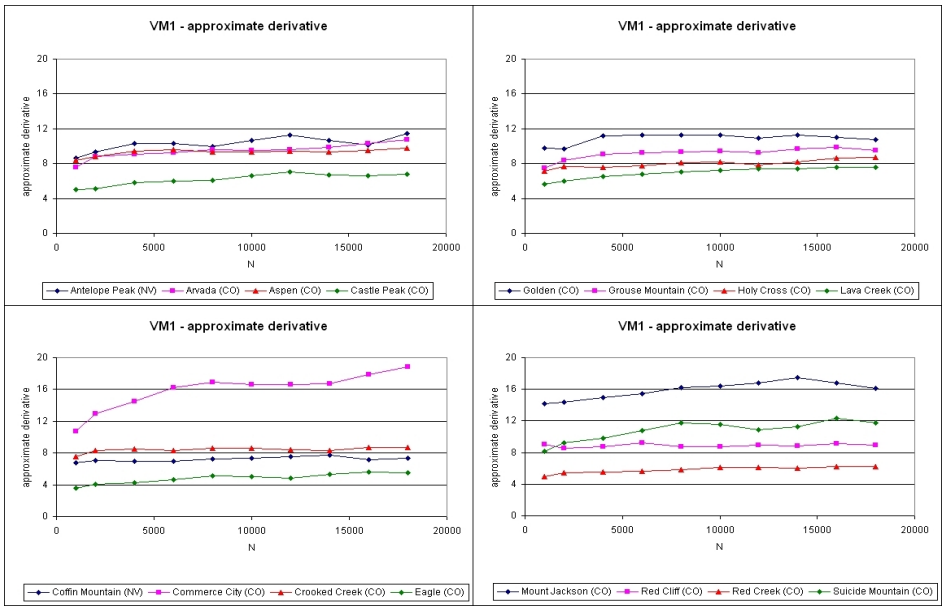

Figure 8: Approximated first derivative of the VM complexity (viewpoint 1) as a function of n.

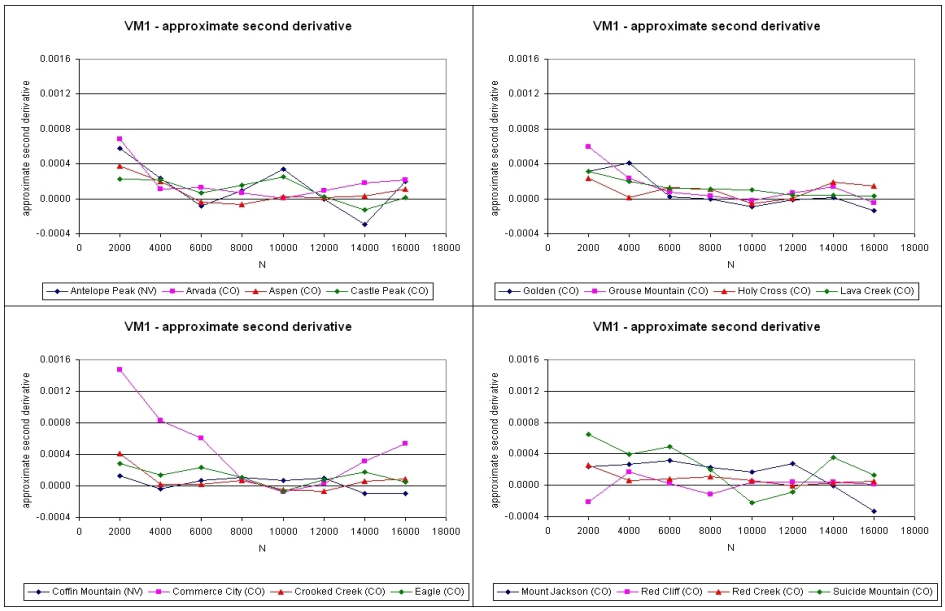

Figure 9: Approximated second derivative of the VM complexity (viewpoint 1) as a function of n.

6 Conclusions

We have studied 208 polyhedral terrains representing sixteen areas in the US. They are generated in such a way that they are similar to typical terrains that are used in GIS. We have observed that these particular terrains satisfy the input model of [16], in the sense that the parameters α, c, d, and β are all constants with respect to the number of triangles of the terrain, n.

The terrains that we used, and also the majority of the terrains that occur in applications, originate from a regular grid, that is, the xy-placements of their vertices are restricted. This implies that some input model parameters are naturally bounded from below resp. above. The

main artifact is that the xy-projection of an edge of the terrain is never shorter than the DEM cell size. This immediately gives a lower bound on the fatness. The maximum length of a projected edge is the diagonal of the input domain, which implies that the edge length ratio is bounded from above by a constant (depending on N). The dihedral angle is also bounded from above, by a scene-specific value, as we discussed in Section 4.4.

Complexity bounds that are obtained with a realistic input model always depend on the value of the input model parameters. Therefore, it is not only desirable to have parameters that are constants with respect to n, but also to have relatively *small* parameters when they are used as upper bounds in input model assumptions and relatively *large* parameters when they are used as lower bounds. In our experiments, the parameter α is a lower bound. The smallest α that we encountered was 0.155 (Aspen (CO); $n = 1000$), . The parameter c is an upper bound, and the largest c that we found was 1.315 (all TINs for Golden (CO)). The parameter d is an upper bound, and the largest d that we found was 467.7 (Commerce City (CO); $n = 500$)). Finally, the parameter β is also an upper bound, and the largest β in our experiments was 89.11 (Aspen (CO); $n = 1000$).

In Section 5, we counted vertices of the transparent visibility map. Its complexity is an upper bound on the complexity of the occluded visibility map. Of course, in GIS people will only be interested in the latter. Thus, it might be interesting to evaluate the complexity of the occluded visibility map in practice to see whether this VM also can attain superlinear complexity in realistic terrains.

We have studied terrains of various resolutions, that is, terrains whose vertices are all extracted from the same DEM, but that have different approximation errors. Another way to verify the input model would be to keep the approximation error the same, or to keep the resolution the same (e.g., one out of every 1000 grid points is a vertex) and to expand and shrink the DEM domain. We expect however, that these methods would give similar results to those of ours, since we implicitly also compare 16 terrains with equal resolution.

We have created TINs for values of n in the range of 500 to 20,000. This is not as much as n will be in typical GIS applications, and of course the amount of data will keep growing in the future. We did not have a supercomputer to our disposal, and our computations quickly became more demanding with growing n. Our results do give some evidence that the input model from [16] is realistic for terrains from practice, but it still would be useful to obtain further such evidence by evaluating larger terrains.

References

[1] M. de Berg, A. F. van der Stappen, J. Vleugels, and M. J. Katz. Realistic input models for geometric algorithms. *Algorithmica*, 34 (1), pp. 81–97, 2002.

[2] Z. Chen and J. A. Guevara. System selection of Very Important Points (VIP) from digital terrain models for constructing Triangular Irregular Networks. In *Proc. 8th Internat. Sympos. Comput.-Assist. Cartog*, pp. 50–56, 1988.

[3] Computational Geometry Algorithms Library. `http://www.cgal.org`.

[4] P.-O. Fjällström. Evaluation of a Delaunay-based method for surface approximation. *Compu. Aided Des.*, 25 (11), pp. 711–719, 1993.

[5] P.-O. Fjällström. Polyhedral approximation of bivariate functions. In *Proc. 3rd Can. Conf. on Comput. Geom.*, 1991.

[6] R. J. Fowler and J. J. Little. Automatic extraction of irregular network digital terrain models. In *Proc. SIGGRAPH'79*, pp. 199–207, 1979.

[7] M. Garland and P. Heckbert. Fast Polygonal approximation of terrains and height fields. Tech. Rep. CMU-CS-95-181, Carnegie Mellon Univ., 1995.

[8] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. *Proc. SIGGRAPH'97*, pp. 209–216, 1997.

[9] P. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms. Tech. Rep. CMU-CS-95-194, Carnegie Mellon Univ. 1995.

[10] M. Heller. Triangulation algorithms for adaptive terrain modeling. In *Proc. 4th Int. Symp. on Spatial Data Handling*, pp 163–174, 1990.

[11] H. Hoppe. Progressive meshes. In *Proc. of ACM SIGGRAPH'96*, pp. 99-108, 1996.

[12] J. Lee. A drop heuristic conversion method for extracting irregular networks from digital elevation models. In *Proc. GIS/LIS'89*, pp. 30–39, 1989.

[13] P. Lindstrom and G. Turk. Fast and memory efficient polygonal simplification. In *Proc. IEEE Vis.'98*, pp. 279-286, 1998.

[14] D. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Comput. Graph. and Appl.*, 21 (3), pp. 24–35, 2001.

[15] M. McKenna. Worst-case optimal hidden-surface removal. *ACM Trans. Graph.*, 6:19–28, 1987.

[16] E. Moet, M. van Kreveld, and A.F. van der Stappen. On realistic terrains. In *Proc. 22nd Ann. ACM Symp. on Comput. Geom.*, pp. 177–186, 2006.

A The Terrains

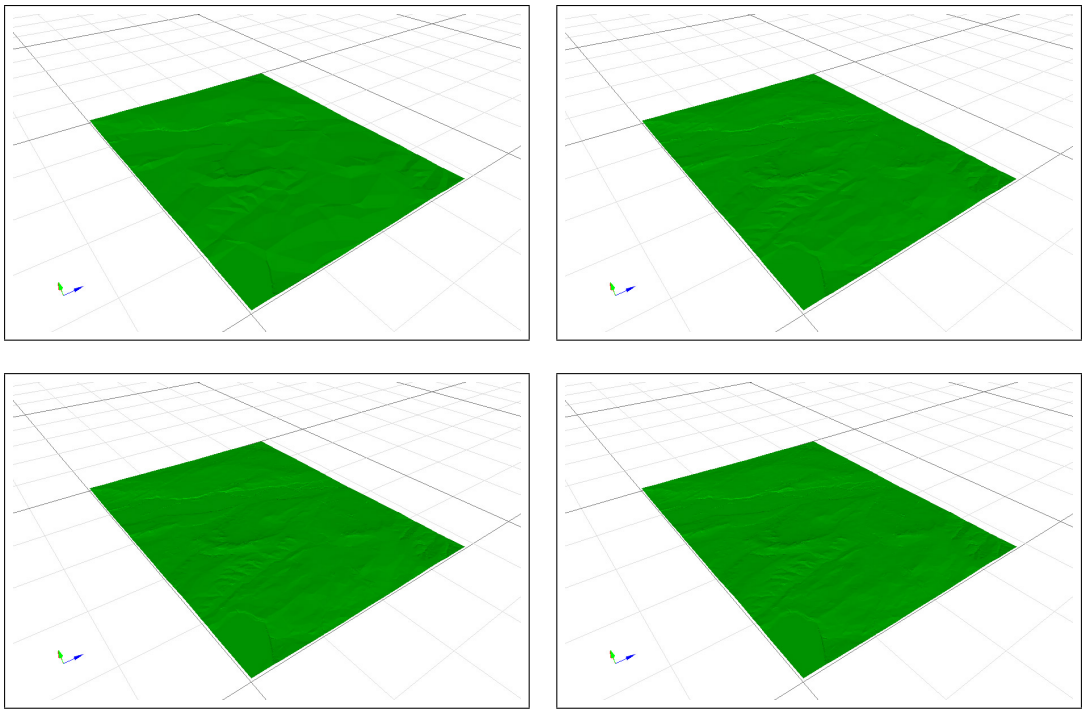

Figure 10: Arvada (CO).

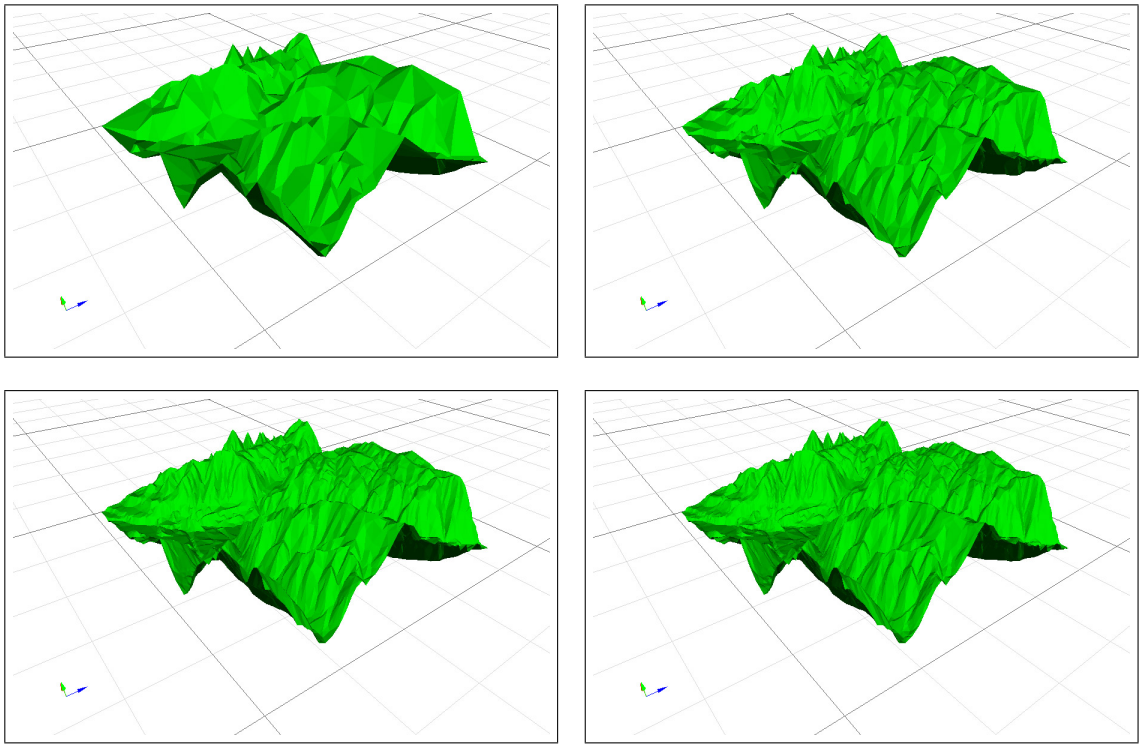

Figure 11: Aspen (CO).

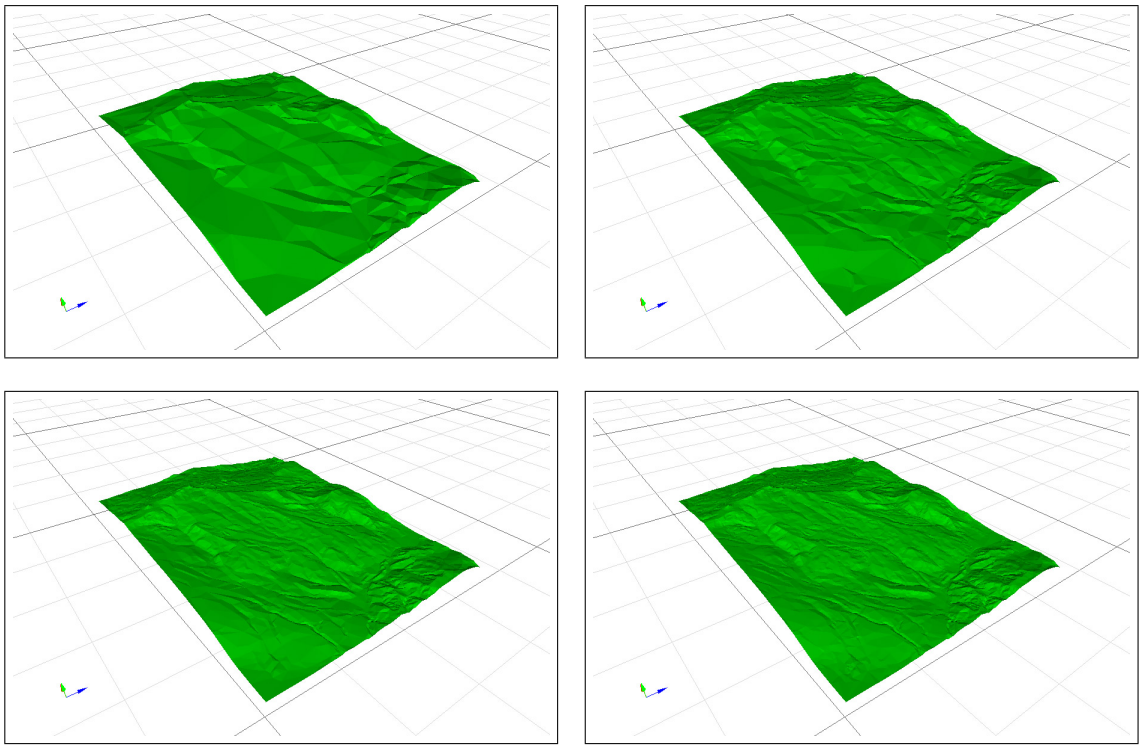

Figure 12: Castle Peak (CO).

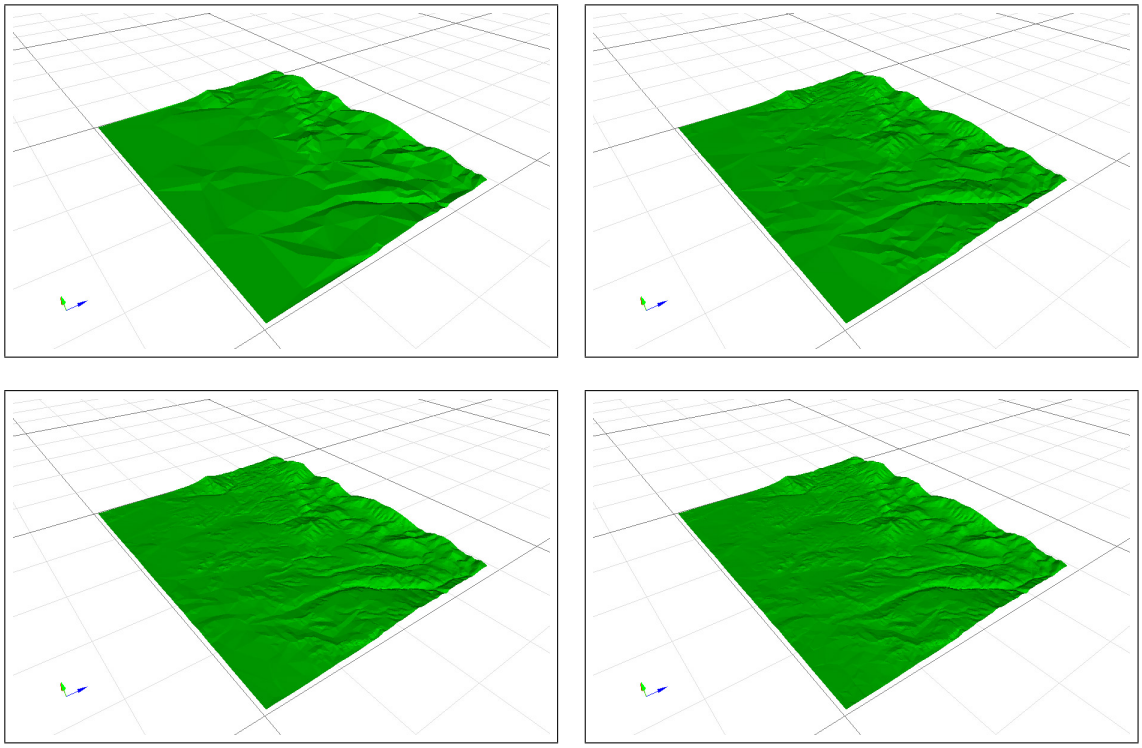

Figure 13: Coffin Mountain (NV).

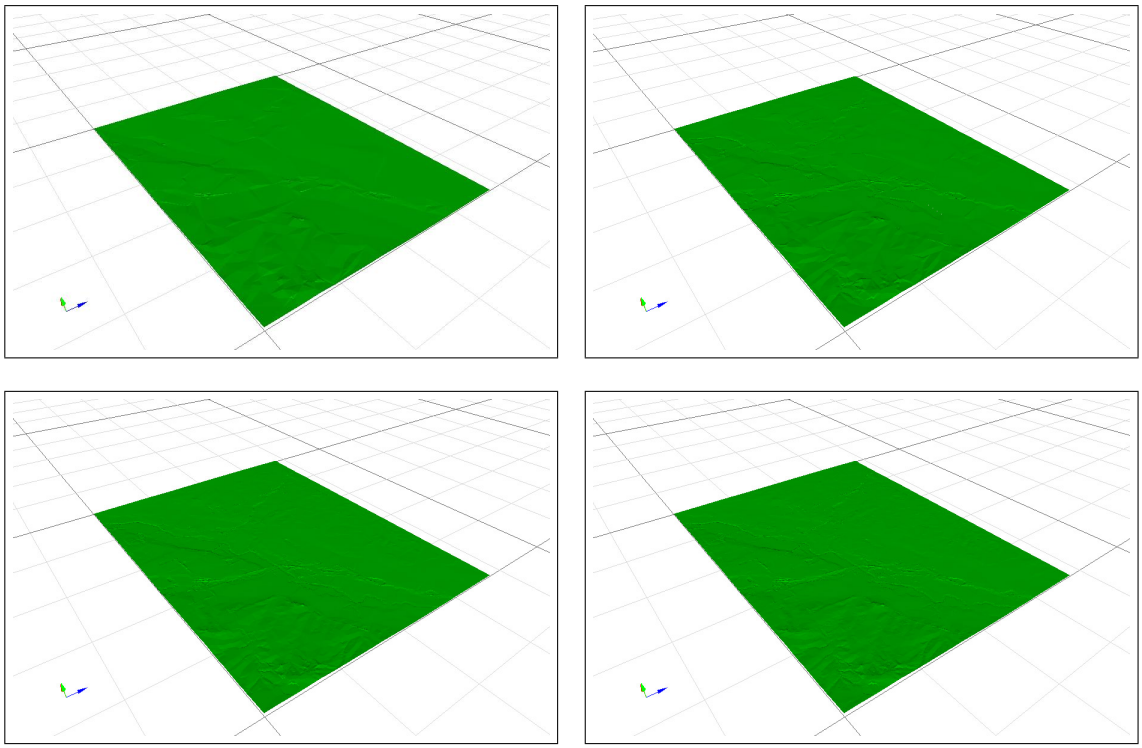

Figure 14: Commerce City (CO).

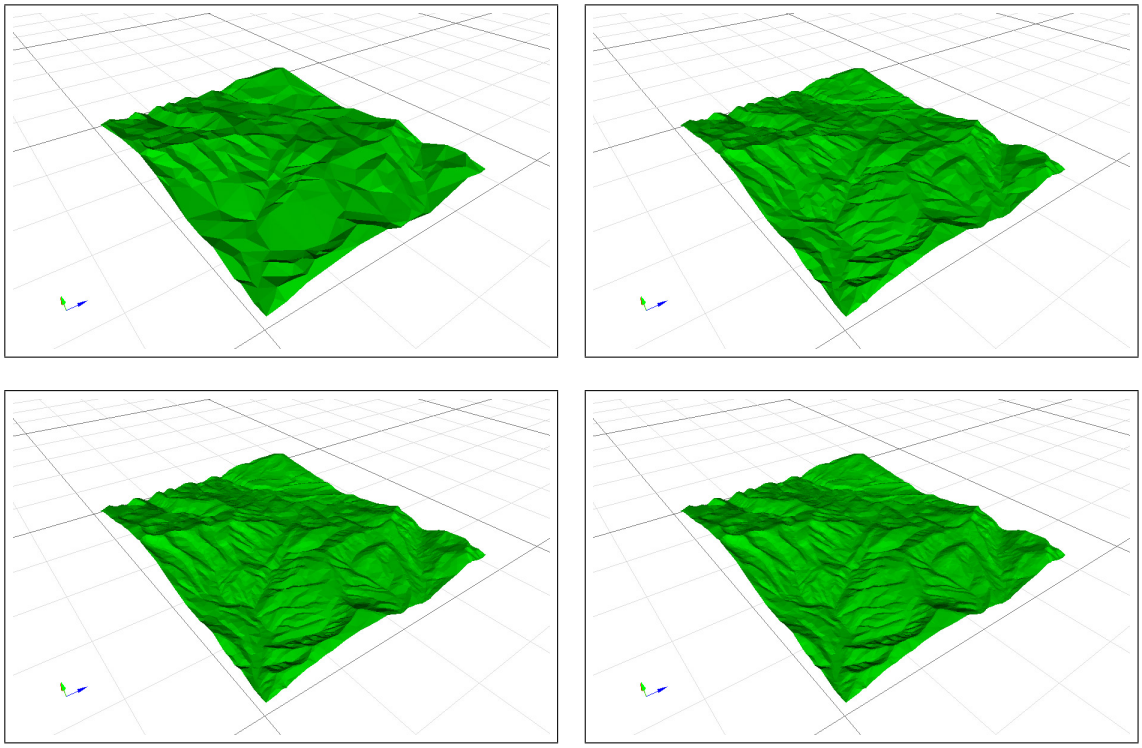

Figure 15: Crooked Creek (CO).

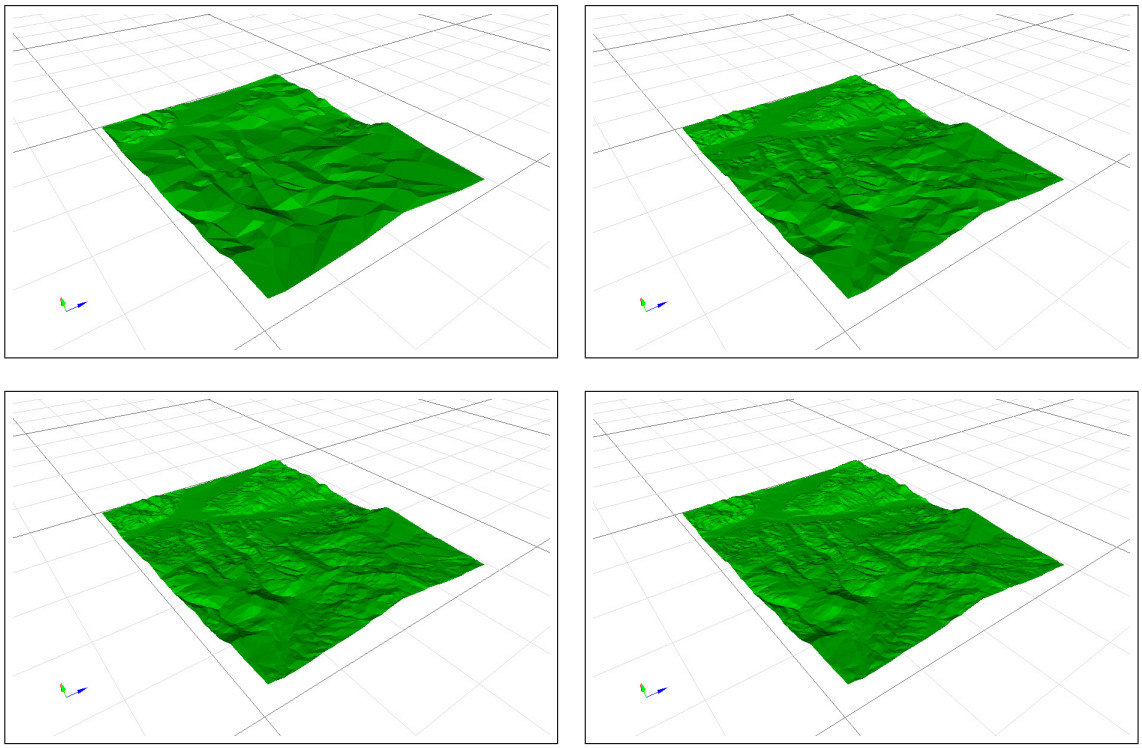

Figure 16: Eagle (CO).

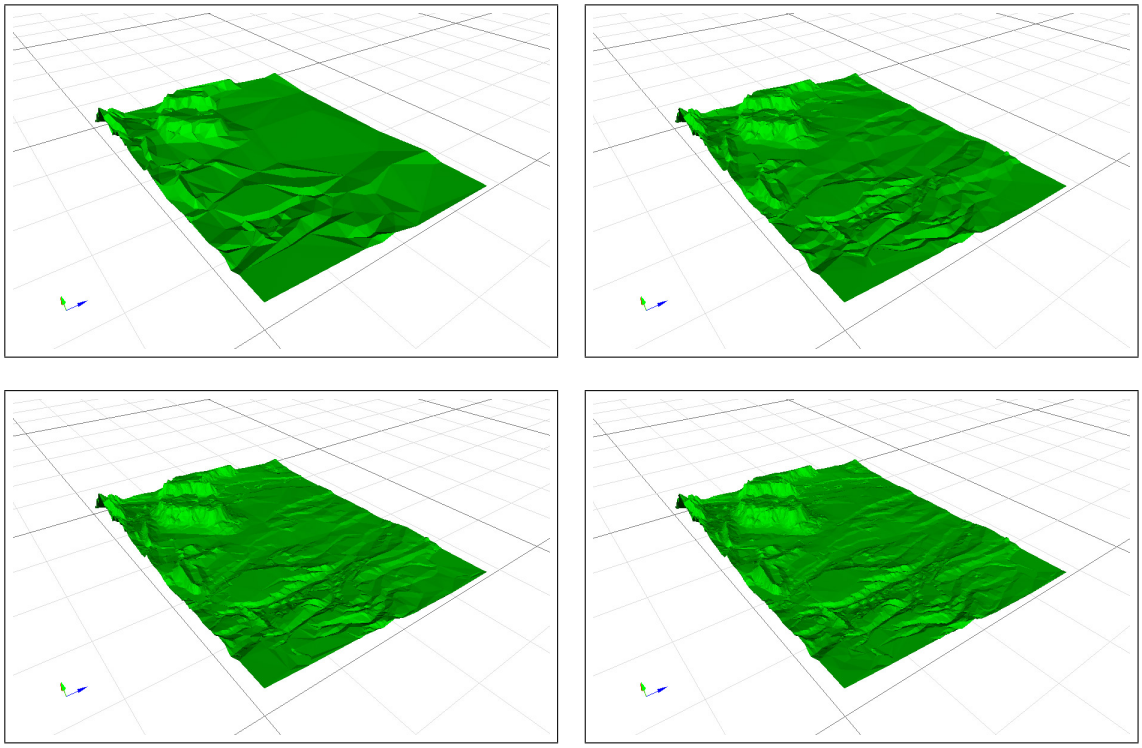

Figure 17: Golden (CO).

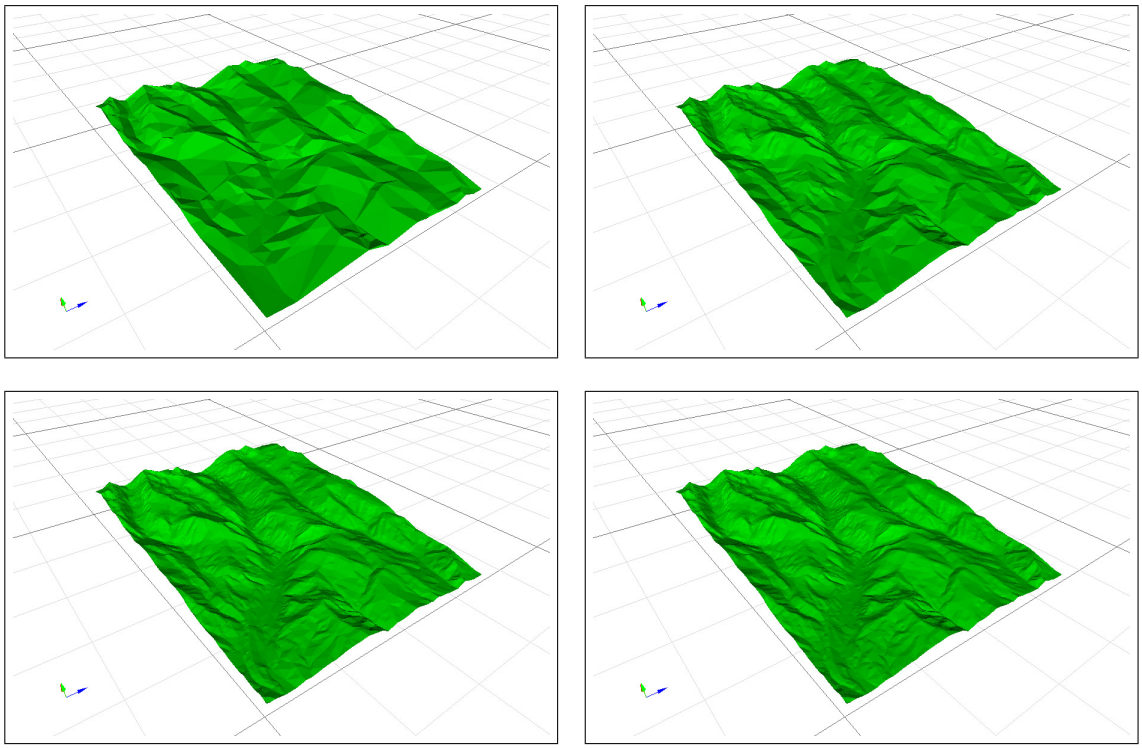

Figure 18: Grouse Mountain (CO).

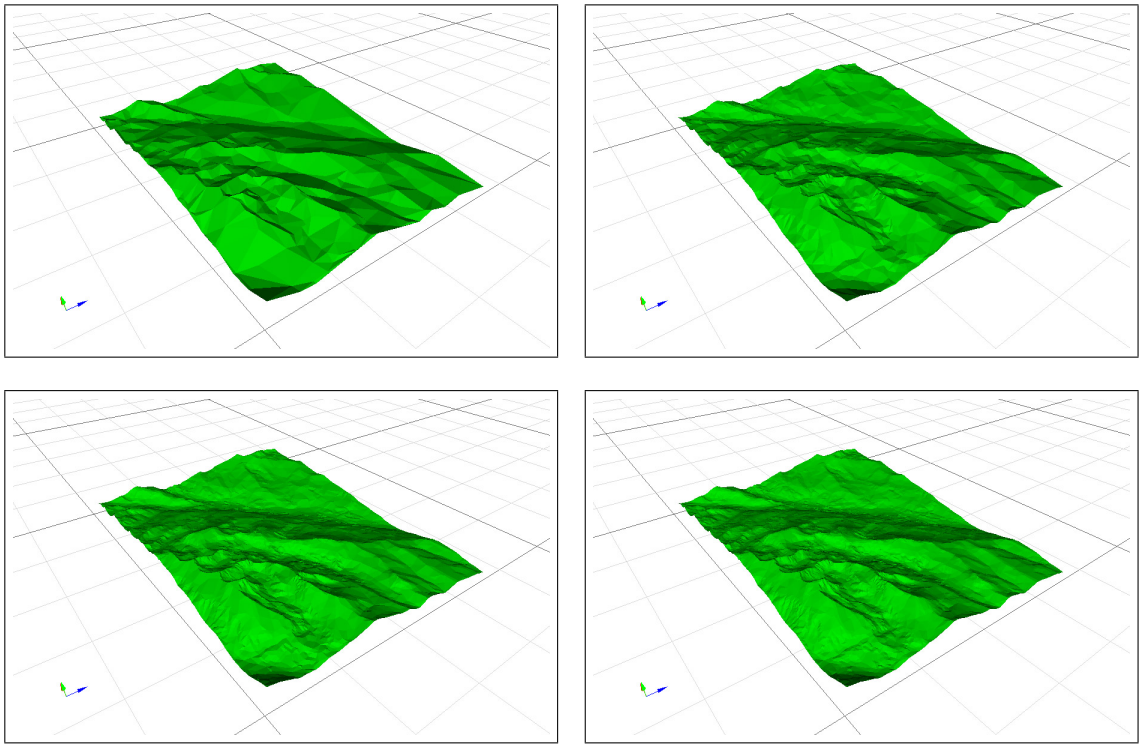

Figure 19: Mount of the Holy Cross (CO).

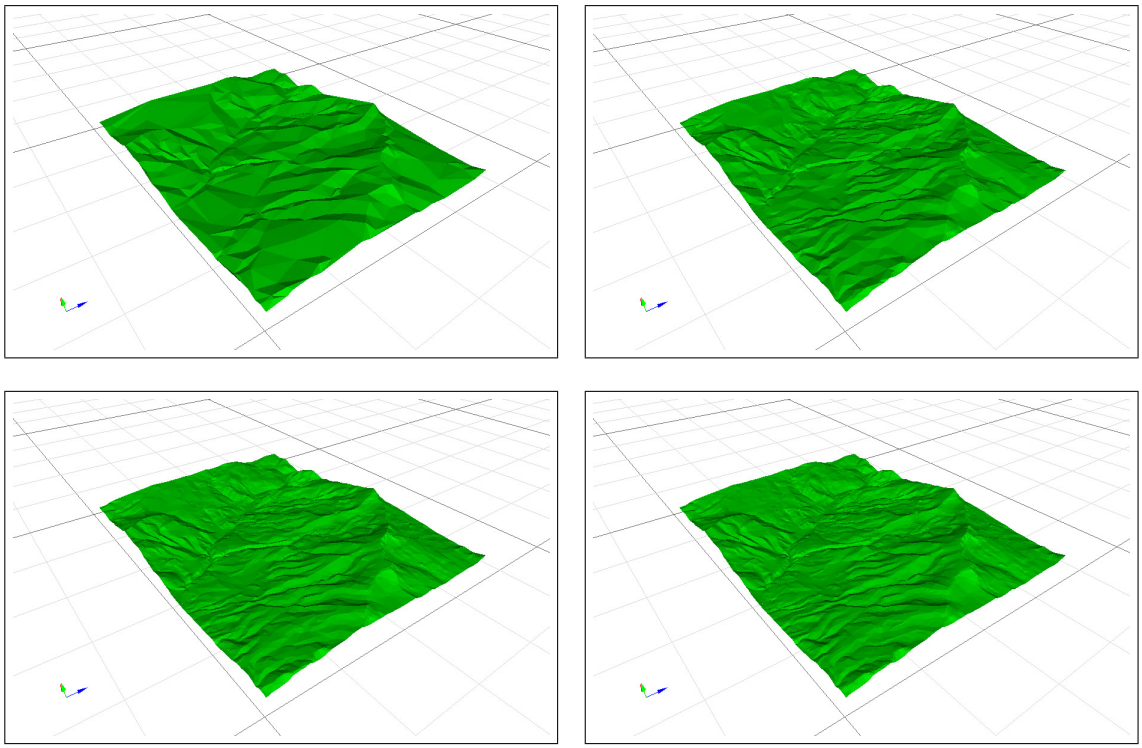

Figure 20: Lava Creek (CO).

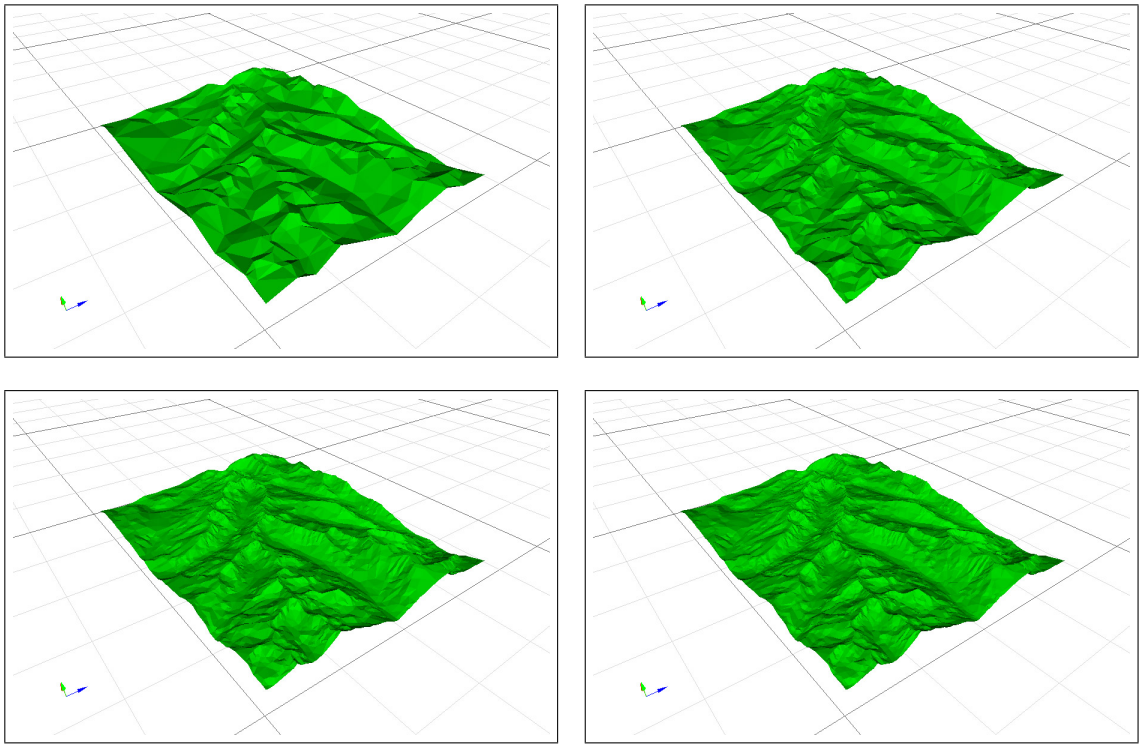

Figure 21: Mount Jackson (CO).

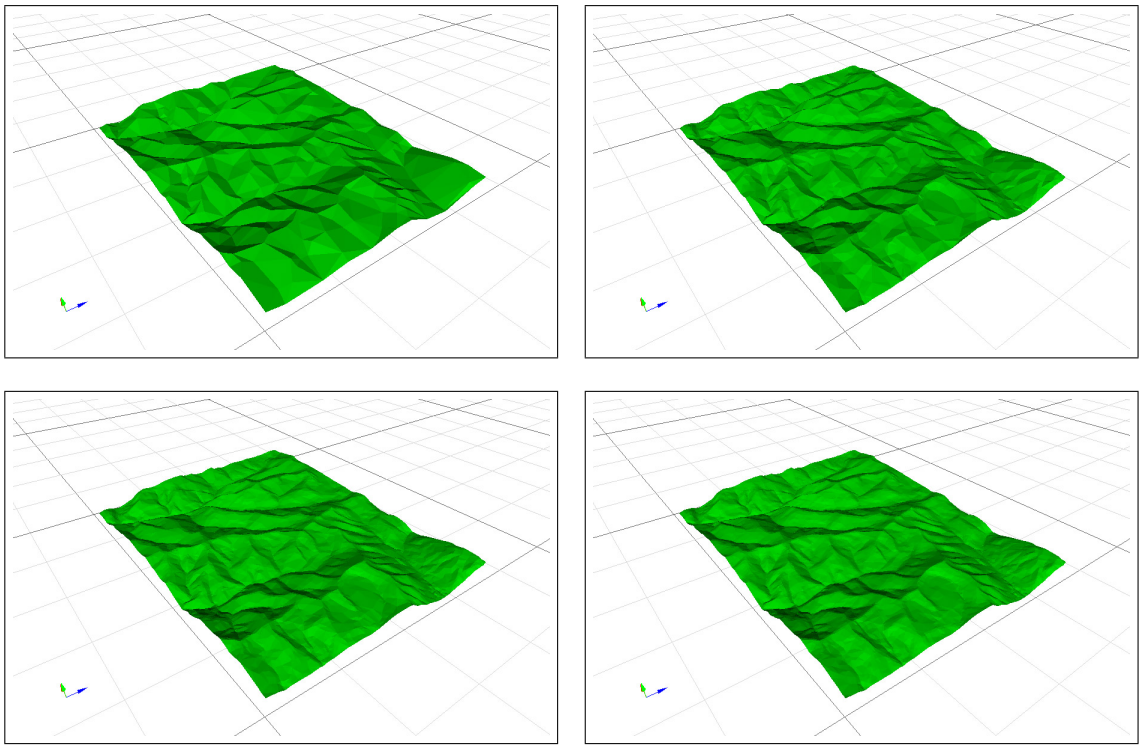

Figure 22: Red Cliff (CO).

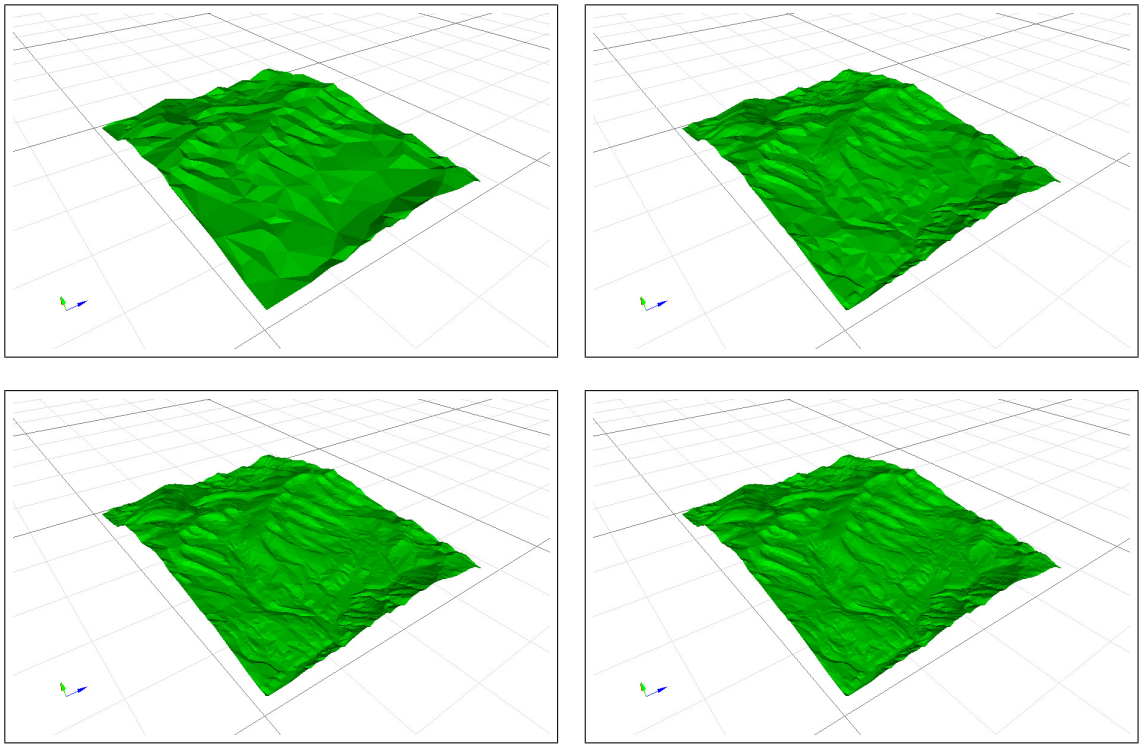

Figure 23: Red Creek (CO).

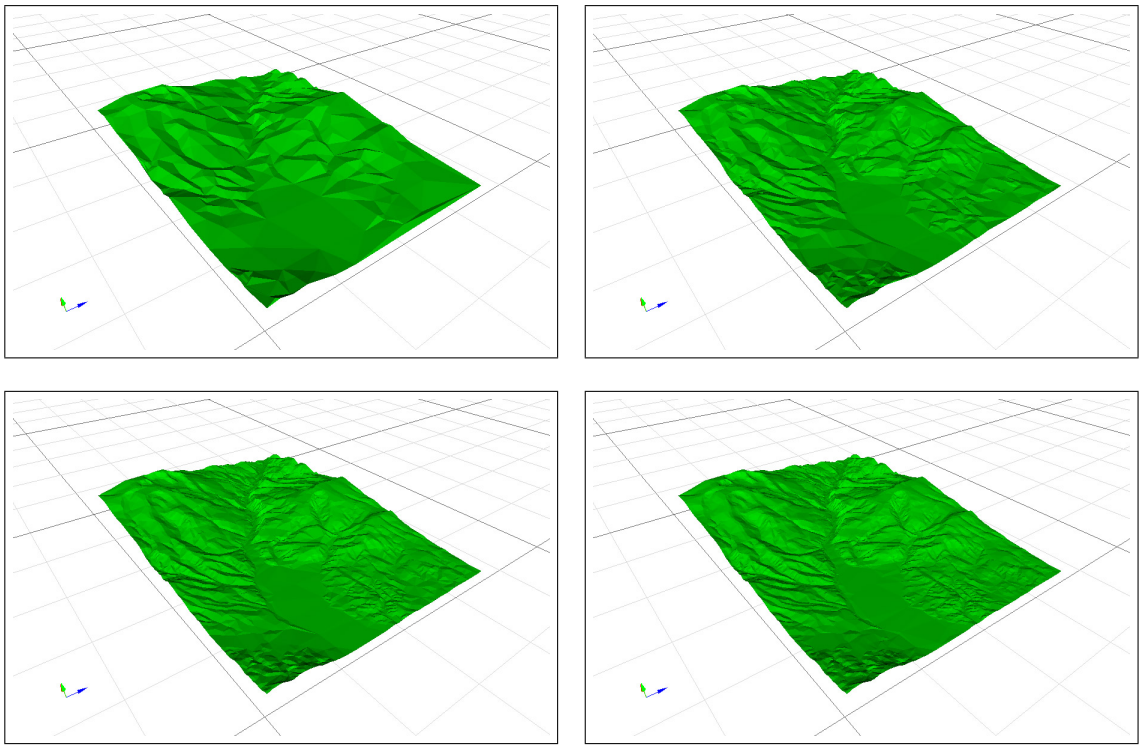

Figure 24: Suicide Mountain (CO).