Optimization of Procedurally Generated Landscapes[*]

Aleksandr Mezhenin[0000-0002-7150-9811] and Anastasia Shevchenko[0000-0003-2200-3264]

ITMO University, Kronverksky Ave. 49, St. Petersburg, Russia
`mejenin@mail.ru, chertpaderi@gmail.com`

Abstract. This paper discusses the optimization of procedurally generated polygonal landscapes. The Level of Detail method is considered and its shortcomings are listed. The proposed approach to solving the optimization problem based on the Ramer-Douglas-Pecker algorithms for the three-dimensional case, Delaunay triangulation, and the Hausdorff metric is presented. To achieve better results in number of triangles of the optimized mesh and maintaining landscape detail than some LOD implementations the follows is proposed: from a height map, in a certain way, points are selected that most accurately convey the curvature and terrain features. Other points are deleted. The basis of the obtained points, an irregular triangulated grid is constructed. The proposed method for analyzing the similarity of polygonal models of an arbitrary topological type can serve as a basis for the implementation of the corresponding algorithms. The use of the weighted average when calculating the normal vectors, according to the authors, increases the accuracy of the subsequent calculation of the Hausdoff metric. Issues of assessing the quality of optimization are considered. A mathematical model is proposed. A prototype of the optimizer for the polygonal mesh was developed and tested.

Keywords: Procedural Landscape Generation, Polygonal Mesh Optimization, Hausdorff Metric, LOD.

1 Introduction

Procedural landscape generation (PGL) is the automatic creation of a three-dimensional landscape model without human intervention or with its minimal contribution. This approach is often used in the field of computer games, in various simulators or training programs, as well as in the film industry.

One of the most common methods of procedural landscape creation is the generation of a height map using various stochastic or fractal algorithms [1, 2]. The height map in this case is a black and white image in which each pixel contains information about the height of the future point of the landscape polygonal mesh. There are vari-

[*] Publication financially supported by RFBR grant №20-01-00547

ous programs that allow you to create such maps, for example, World Machine or L3DT.

Procedural landscape generation is used in many industries, in particular, in the gaming industry, in cinema, in training programs, simulators, etc. In some cases, the main task of PGL is to achieve the maximum realism of landscapes, in some - quickly and variably create a virtual environment.

When the terrain is generated for real-time rendering, as is the case in games or, for example, navigation applications, there is a need to optimize the 3D model, because otherwise you may encounter a significant delay between rendering frames. And if in computer games this situation is more unpleasant than critical, then in navigation applications that display terrain data received from the satellite, such a problem can cause extremely negative consequences. In computer games, this delay can make the process of the playing uncomfortable, in the worst situations – impossible. However, in navigation applications render delays can cause extremely negative consequences. In this situation, 3D models contain information about real terrain, which means that correct navigation requires a timely and most reliable display of the terrain data received from the satellite. At the stage of landscape generation, it can only be optimized at the level of the polygonal mesh, that is, to reduce the number of polygons. At the same time, it is highly desirable to keep the landscape recognizable, not to distort the silhouette, and in special cases, to preserve distinctive details. All of the above indicates that there is a problem associated with optimizing the landscape model and maintaining sufficient detail.

2 Existing Optimization Methods

In modern computer graphics systems, especially those operating in real time, several types of algorithms are used to reduce computational costs. One of these algorithmic approaches is to display an object with a different level of detail in terms of visual resolution. When creating each subsequent level of simplification, small model details that do not actually affect the image structure are combined and replaced with larger ones. This technology is known as Level of Detail (LOD) [3].

We used a simple version of algorithm LOD, which reduces the number of points with a certain step evenly over the entire plane. There are a lot of more complicated methods of LOD [4], which modify the polygonal mesh partially, depending on the detail of the terrain fragment or the distance between fragment and camera. At this stage, we are considering only a fragment of such a complexly modified mesh to understand how our method works with different types of landscapes. Obviously, if the level is too low, the landscape model may lose a significant amount of detail, but if you increase it, there is a risk of going beyond the limit in terms of the number of model polygons. Also, the quality of the result of this method depends strongly on the "roughness" of the landscape, that is, on how detailed it is.

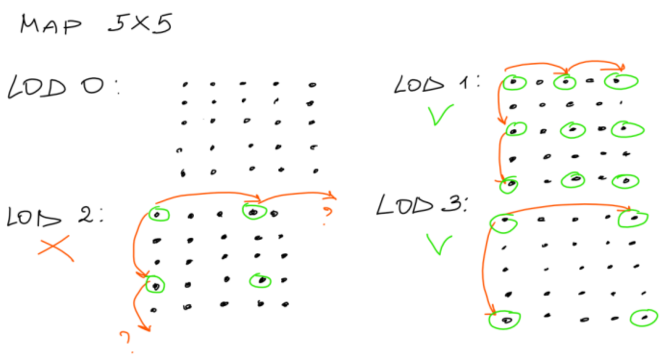


Fig. 1. An example of the work of the considered LOD method.

For citations of references, we prefer the use of square brackets and consecutive numbers. Citations using labels or the author/year convention are also acceptable. The following bibliography provides a sample reference list with entries for journal articles [4], an LNCS chapter [5], a book [6], proceedings without editors [7], as well as a URL [8]. Consider two different landscapes that differ from each other in the degree of smoothness (Fig. 1). For each landscape, we perform a series of actions:

1. Optimize the landscape using the LOD method, having examined and saved all the levels of detail possible for a given polygonal mesh;
2. For each optimized grid, we calculate the Hausdorff distance [9, 10] relative to the original high-poly model. An image representing inputs with caption (flattened and detailed) (see Fig. 2).

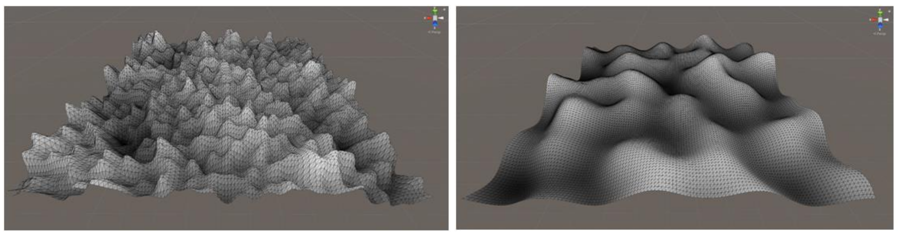
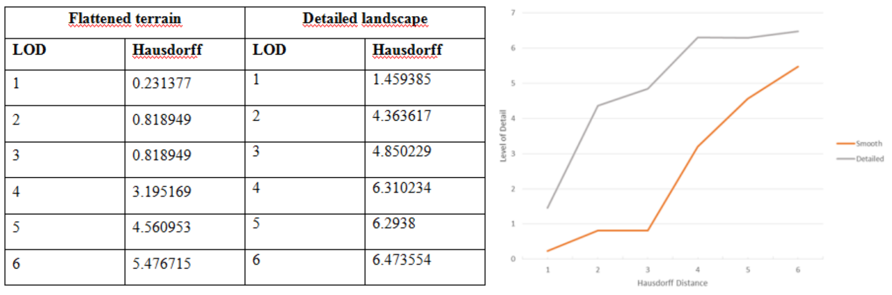


Fig. 2. An example of the work of the considered LOD method.

The graph and table of measurements of the dependence of the Hausdorff distance on the level of detail are presented in Figure 3.



Flattened terrain		Detailed landscape	
LOD	**Hausdorff**	**LOD**	**Hausdorff**
1	0.231377	1	1.459385
2	0.818949	2	4.363617
3	0.818949	3	4.850229
4	3.195169	4	6.310234
5	4.560953	5	6.2938
6	5.476715	6	6.473554

Fig. 3. An example of the work of the considered LOD method.

Most iterative algorithms for simplifying polygonal models can be divided into three categories [10]:

- Thinning peaks;
- Coagulation of the ribs;
- Thinning faces.

Clusters are a three-dimensional grid, i.e., the coordinates of the vertices are actually rounded with a given accuracy, and when several vertices coincide, after rounding, the vertices are replaced by one. All references of the set K to collapsed vertices are replaced by references to a new vertex. The advantage is high speed, the disadvantage is not very high quality of the resulting models.

More precisely, the topology of the model is transmitted by the vertex thinning algorithm, which is executed in several passes. At each stage, the vertices located from the averaged plane of neighboring vertices are removed at a distance less than the specified one. One of the problems that occurs during the implementation of the algorithm is the removal of faces belonging to deleted vertices. In this case, a hole may be formed, which is filled with a triangulation operation. The algorithm gives good results, but triangulation operations require additional time. Algorithms based on folding of edges can be considered as a special case of removing vertices, but they do not require additional triangulation operations. Therefore, their implementation gives good results in terms of quality and speed. Coagulation of an edge is a merger of two vertices forming an edge into one. In this case, in the general case, the removal of two triangular cells occurs.

The sequence of coagulation of the edges is determined by some measure of error, which reflects the local geometric deviation of the cells. The methods for calculating this error determine the difference between the algorithms of this class.

The following error measures are known that are used to select a strategy when folding an edge:

- Determining the average distance between new triangles and sample points in the original model.
- Finding the tolerance value as a convex combination of spheres located at each vertex of simplification. The choice of faces is based on the smallest length, and

the new vertex is positioned so that the new surface is guaranteed to lie within this tolerance.
- Support for communication between points of the original model and the corresponding neighborhoods on the simplified model.

Due to the fact that both in nature and in various virtual environments, the landscape is more "rough" than smoothed, it is necessary to find an approach to optimizing 3D landscape models better than LOD.

3 Proposed Approach

As noted earlier, the generation of landscapes using elevation maps leads to the creation of a regular grid, which can still be highly polygonal, and the LOD method cannot always maintain the desired detail or reduce a sufficient number of polygons. Therefore, it is proposed to modify this approach and generate terrain from a height map of an irregular triangulated optimized grid using a pair of Ramer-Douglas-Pecker [8] and Delaunay triangulation algorithms. The work uses an algorithm for lines. There are several publications about the 3D version, they were tested, but the results were unsatisfactory.

We adapted an original Ramer-Douglas-Pecker algorithm for a two-dimensional array of points. Each point has two status variables called **rowStatus** and **columnStatus**. In the first step of the adapted Ramer-Douglas-Pecker algorithm the rows of the points array are considered. If the point is not important its **rowStatus** is set to FALSE. In the second step algorithm processes the columns of the array in the same way, if point is not important in the column its **columnStatus** is set to FALSE. In the final step points with both positive statuses are kept. This new array of points will compose an optimized terrain mesh.

To achieve better results in number of triangles of the optimized mesh and maintaining landscape detail than some LOD implementations the follows is proposed:

- From a height map, in a certain way, points are selected that most accurately convey the curvature and terrain features. Other points are deleted;
- Then, on the basis of the obtained points, an irregular triangulated grid is constructed [3].

In this approach, in step 1, two algorithms will be used: Diamond Square [4] and Perlin Noise [5]. The generation results with their help are very different from each other and generate fundamentally different landscapes, which will allow taking into account different situations in the developed approach. Using DiamondSquare produces rougher mountain surfaces, while Perlin Noise can generate smooth, hilly terrain (see Fig. 4).

Step two is the most important. It is the algorithm of this stage that will determine the quality of preservation of landscape detail. In search of a suitable solution, it was found that similar tasks of reducing landscape data are solved in the fields of topography and geodesy. So, in [11], various methods for selecting points from raster surface data obtained from a satellite are presented. Due to the relative simplicity of implementation and versatility, the Ramer-Douglas-Pecker algorithm for three-dimensional data was chosen for the approach being developed [13]. In the simplest case, this algorithm has one configurable parameter - ε, with which you can adjust the number of deleted points.

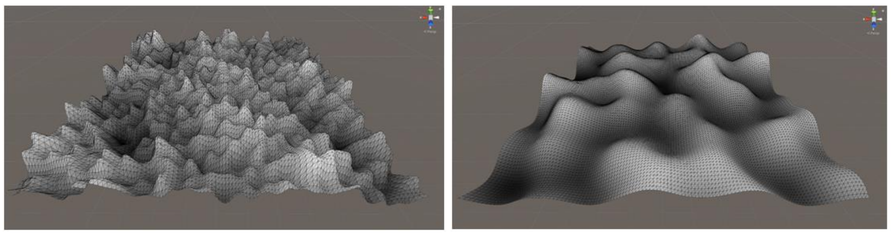


Fig. 4. DiamondSquare algorithm example. An example of the Perlin Noise algorithm.

Finally, at the third step of the proposed approach, the obtained control points will serve as the basis for constructing the Delaunay triangulation. There are a lot of ways to construct such a triangulation; many of them are described in [14]. In order not to spend too much time on complex implementations, it was decided at this stage to implement the Delaunay triangulation with a simple iterative algorithm.

4 Optimization Quality Assessment

The criteria used in the simplification process are highly differentiated and do not give the total value of the simplification error. In fact, many algorithms do not return measures of the approximation error obtained by simplifying the polygon mesh [15-18].

In most cases, the Root Mean Square Error (RMSE) is used to estimate the accuracy of 3D model reconstruction or to solve simplification problems,

$$RMSE = \sqrt{\frac{1}{N_{Rows}N_{Cols}} \sum_{i=1}^{N_{Rows}} \sum_{j=1}^{N_{Cols}} \left| f_{i,j} - d_{i,j} \right|^2} \tag{1}$$

This approach, according to the authors, does not allow obtaining reliable results. One of the possible solutions to this problem is to use the Hausdorff dimension, which will allow obtaining a quantitative estimate of the similarity of polygonal objects [6, 7].

For the sake of simplicity, let us imagine discrete 3D models represented by triangular meshes, since this is the most general way of representation of such data. The triangular mesh M will be a representation of the ensemble of points P in R3 (apices) and the ensemble of triangles T that describe the connection between the apices of P. Let us denote the two continuous surfaces S and S ', and

$$d(p,S') = \min_{p' \in S'} \left\| p - p' \right\|_2 \qquad (2)$$

where – is the Euclidian norm.

Therefore Hausdorff metrics between S and S': . It is important to understand the fact that the metrics is not symmetrical, h.e. Let us denote as the direct distance, as inverse distance. Then the symmetrical metrics:

$$d_2(S,S') = \max\left[d(S,S'), d(S',S)\right] \qquad (3)$$

Symmetric metrics ensures a more precise measurement of an error between two surfaces, since the calculation of a “one-way” error can lead to significantly underestimated distance values, as it was shown in Figure 5.

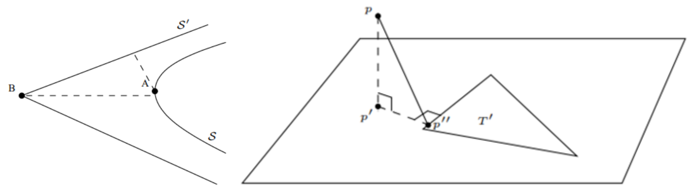


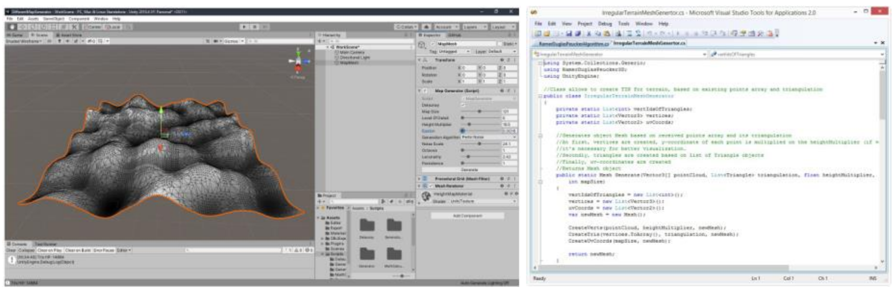
Fig. 5. Distance Comparison. Projection Construction.

One can see that $d(S,S')$ is smaller than $d(S',S)$, since $d(A,S') << d(S,B)$. Thus, a not very large one-way distance does not mean a small presentation. The calculation of the Hausdorff distance between two discrete surfaces $M(P,T)$ and is related to the preceding definitions. Let us focus on calculation of the Hausdorff direct distance, h.e. , since the symmetric distance can be calculated from the direct and inverse distances. The distance between any point p from M (p is assumed not to be from P) and M' can be calculated from the estimation of the distance minimum between p and all triangles $T \in T'$. When the orthographical projection p 'of p on the plane T' is inside the triangle, the distance between the point and the triangle is simply a distance from the point to the plane. When the projection remains outside T ', the distance between the point and the plane is the distance between p and the closest p '' from T', which should lie on one of the sides of T ' (Fig. 5).

Although d(p,S') can be calculated for any point p, it is essential to perform sampling to calculate the maximum p϶S. Each T triangle is sampled, and the distance between each sample and M 'is estimated. Each triangle sampling is performed as follows: two sides of the triangle are considered as directions for the sample lattice. In accordance with the criterion of length, each side is selected with n points. By means of directions, it is easy to construct a 'correct' mesh of the triangle under consideration. According to this sampling, n (n + 1) / 2 samples are constricted in each triangle. An interesting property of this sample is that the triangle can be split into smaller ones that possess all the same areas, which leads to much simpler calculations of the integrals taken through surface. Representative illustration of a sample made on a triangle for n = 5. The sides adopted as main directions are in bold and the samples are specified with black dots.

5 Experimental Results

The optimizer prototype is implemented in Unity3d, in C # (Figure 6).



```
using System.Collections.Generic;
using RamerDouglasPeucker3D;
using UnityEngine;

//Class allows to create TIN for terrain, based on existing points array and triangulation
public class IrregularTerrainMeshGenerator
{
    private static List<int> vertIdsOfTriangles;
    private static List<Vector3> vertices;
    private static List<Vector2> uvCoords;

    //Generates object Mesh based on received points array and its triangulation
    //At first, vertices are created, y-coordinate of each point is multiplied on the heightMultiplier (if =
    //it's necessary for better visualization.
    //Secondly, triangles are created based on list of Triangle objects
    //Finally, uv-coordinates are created
    //Returns Mesh object
    public static Mesh Generate(Vector3[] pointCloud, List<Triangle> triangulation, float heightMultiplier,
        int mapSize)
    {
        vertIdsOfTriangles = new List<int>();
        vertices = new List<Vector3>();
        uvCoords = new List<Vector2>();
        var newMesh = new Mesh();

        CreateVerts(pointCloud, heightMultiplier, newMesh);
        CreateTris(vertices.ToArray(), triangulation, newMesh);
        CreateUvCoords(mapSize, newMesh);

        return newMesh;
    }
```

Fig. 6. Unity3d environment. Code snippet.

During testing, the following independent variables were set:

- The size of the landscape in points;
- Algorithm for generating height maps: Perlin Noise or Diamond Square;
- The parameter ε of the Ramer-Douglas-Pecker algorithm, on which the number of discarded points depends;
- The lod parameter of the Level of Detail optimization algorithm.

As the measured characteristics will be considered:

- The number of triangles / vertices of the optimized model. Measurement scale: absolute. This parameter is necessary for comparing optimized and non-optimized models, as well as for calculating the degree of optimization, which will be mentioned later.
- The Hausdorff distance between the high-poly model and the optimized one. Measurement scale: absolute. Hausdorff distance will show how the optimized model differs from the original. This metric will be calculated in the MeshLab package, where heat maps will also be built, which will make it possible to judge how well or poorly the detail is preserved during optimization.
- The degree of optimization as a percentage of the number of points of the high poly model: Measurement scale: absolute. This parameter will be calculated based on the number of vertices or triangles of the optimized and original model. This is necessary in order to be able to compare models optimized in different ways. So, with one degree of optimization, they can have different Hausdorff distances and different heat maps.

The results of the optimizer are presented in Fig. 7. As an example, it can be seen that the density of points is higher precisely in places of detailed landscape - that is, on hills and depressions.

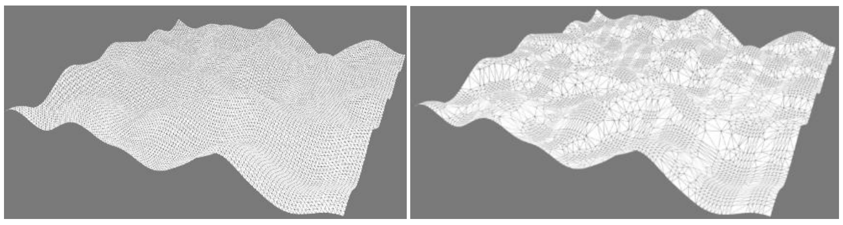


Fig. 7. Optimizer Results.

6 Conclusion

Procedural landscape generation (PGL) is the automatic creation of a three-dimensional landscape model without human intervention or with its minimal contribution. This approach is often used in the field of computer games, in various simulators or training programs, as well as in the film industry. The proposed approach to procedural landscape generation will allow the generation of an irregular triangulated grid that will more accurately transmit a height map than a regular grid optimized by the LOD method. In addition to maintaining a certain level of detail, the developed method will generate a landscape with fewer triangles than a landscape optimized by the LOD method. The proposed method for analyzing the similarity of polygonal models of an arbitrary topological type can serve as a basis for the implementation of the corresponding algorithms. The use of the weighted average when calculating the normal vectors, according to the authors, increases the accuracy of the subsequent calculation of the Hausdoff metric. The proposed approaches can find application in the problems of assessing the quality of algorithms for reconstruction and recognition of 3D models, as well as in problems of multiscale representation of polygonal models.

References

1. Luis Oswaldo, Valencia-Rosado and Oleg Starostenko: Methods for Procedural Terrain Generation: A Review, (2019), https://link.springer.com/chapter/10.1007%2F978-3-030-21077-9_6.
2. Rose, T. J., & Bakaoukas, A. G.:Algorithms and Approaches for Procedural Terrain Generation - A Brief Review of Current Techniques, (2016), https://ieeexplore.ieee.org/document/7590336.
3. Terrain Level of Detail, https://graphics.pixar.com/library/LOD2002/4-terrain.pdf.
4. Diamon Square Algorithn, https://en.wikipedia.org/wiki/Diamond-square_algorithm.
5. Perlin noise, https://en.wikipedia.org/wiki/Perlin_noise, last accessed 2020/09/11.
6. Terrain Level of Detail, https://graphics.pixar.com/library/LOD2002/4-terrain.pdf.

7. Triangulated Irregular Network, https://en.wikipedia.org/wiki/Triangulated_irregular_network, last accessed 2020/09/11.
8. Ramer-Douglas-Peucker Algorithm, https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm, last accessed 2020/09/11.
9. Hausdorff Distance, https://en.wikipedia.org/wiki/Hausdorff_distance, last accessed 2020/09/11.
10. Aleksandr Mezhenin, Alena Zhigalova, Similarity analysis using Hausdorff metrics (2019), http://ceur-ws.org/Vol-2344/short5.pdf.
11. Mezhenin A.V. Issledovanie i razrabotka metodov i programmnyh sredstv dlya sozdaniya i otobrazheniya trekhmernyh virtual'nyh ob"ektov v internet// Avtoreferat dissertacii (2005) [in Russian].
12. Enrique R. Vivoni, Valeri Y. Ivanov, Rafael L. Bras, and Dara Entekhabi: Generation of Triangulated Irregular Networks Based on Hydrological Similarity (2004), https://ascelibrary.org/doi/abs/10.1061/(ASCE)1084-699 (2004) 9:4(288).
13. Fei, L., & He, J.A three-dimensional Douglas–Peucker algorithm and its application to automated generalization of DEMs (2009), [doi> 10.1080/13658810701703001].
14. Skvorcov A.V: Triangulyaciya Delone i ee primenenie (2002) [in Russian].
15. Garland, M., and Heckbert, P. S. Surface Simplification Using Quadric Error Metrics. In Computer Graphics (SIGGRAPH 97 Proceedings) (1997),pp. 209–218.
16. Hoppe, H. Progressive Meshes. In Computer Graphics (SIGGRAPH 96 Proceedings) (1996), pp. 99–108.
17. Eck, M., Derose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W. Multiresolution Analysis of Arbitrary Meshes. In Computer Graphics (SIGGRAPH 95 Proceedings) (1995), pp. 173–182.
18. Measuring the difference between two meshes, http://meshlabstuff.blogspot.com/2010/01/measuring-difference-between-two-meshes.html, last accessed 2020/09/11.