THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

*PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,*
*Volume 16, Number 4/2015, pp. 531–538*

# ASPECTS OF GENERATING 3D SURFACES WITH APPLICATIONS IN DRIVING SIMULATORS

Lucian ILEA[1], Ligia MUNTEANU[2], Dan DUMITRIU[2], Mircea DUDESCU[1], Cornel BRIŞAN[1], Veturia CHIROIU[2]

[1] Technical University of Cluj-Napoca, Dept. of Mechatronics and System Dynamics,
Cluj-Napoca 400020, Romania
[2] Institute of Solid Mechanics of the Romanian Academy, 15 Ctin Mille, P.O. Box 1–863, 010141 Bucharest, Romania
E-mail: veturiachiroiu@yahoo.com

Driving simulators are complex technical entities for reproducing the phenomena of real driving. One of their essential components is the visual simulator. For the purpose of creating the visual simulator it is of utmost importance to generate the 3D surfaces such that they simulate both the road and the environment. In this paper, the problem of terrain generation has been encountered. The generation of such surfaces, suitable with real ones, is a natural issue. In the last few decades much research was done and many attempts were made in order to generate random surfaces, as real looking as possible, theoretically justified and easy to model. The purpose of our research is to propose an alternative method for generation of 3D surfaces for different roads by using two uncommon algorithms which were initially designed for other purposes: the Douglas-Peucker algorithm and the so called onion peeling algorithm.

*Key words*: driving simulator, 3D surfaces for roads, Douglas-Peucker algorithm**,** onion peeling algorithm.

## 1. INTRODUCTION

To describe the shape of the road's surfaces, the data about three dimensions of every point are necessary. The contour lines have to be constructed from various relief parameters such as elevation, slope, slope aspect, roughness and so on, and can be used for sight analysis. This paper presents an alternative technique to generate 3D road surfaces. This technique works in two steps: the first step is to obtain the *rough* surface, followed by the second step where the surface is *smoothening*. The first step is completed using one of the methods: the Brownian motion technique [1, 2] and the genetic algorithms [3]. To complete the second step, the Douglas-Peucker algorithm can be successfully used. The Douglas-Peucker algorithm simplifies the rough surface. After our knowledge, such procedure hasn't been enough developed and studied yet. In the Douglas-Peucker algorithm the surface is generated by modifying and slightly moving an initial planar curve. The initial form of the algorithm was independently suggested in 1972 by Urs Ramer [4] and 1973 by David Douglas and Thomas Peucker [5] and several others in the following decade [6–9].

The *onion peeling algorithm* has the objective to find a triangulation over a set of planar points. It works on a set of planar points and consists in determining of some consecutive convex hulls, followed by the triangulation of the planar region situated between two convex hulls. In this paper we pay greater attention towards the layer division which stems from the iterative construction of convex hulls. Each layer is obtained by applying a method to conceive the planar convex hull for the set of points that were not already assigned to a previous layer.

For the second step, we intend to use in addition a method based on *quadtrees* [10–12] for the sake of comparisons. The Douglas-Peucker algorithm gives results that are quite different from those obtained by using *quadtrees*, but both methods can have specific advantages. The virtual roads generation is a process that involves determining a set of height values over a two dimensional grid. Some researchers consider that this process cannot be entirely mathematically and suggest hybrid cooperation between the algorithmic

approaches and the human interaction. The user has the possibility to influence the result by choosing between some patterns or even by manual corrections. This paper is not aimed at the user's contribution, which may be interesting for game developers or artists, but on the computational aspects: several computer programs have been designed and their outcomes have been compared and analyzed.

We are aware of many applications of the proposed alternative algorithm, such as: propagation of chemical events, study of the Earth's atmosphere, network protocols [13], fingerprints verification [14]. In this paper we focus only on the building of virtual roads close to the real ones, for the purpose of their use in driving simulators.

## 2. SURFACES GENERATION USING THE DOUGLAS-PEUCKER ALGORITHM

As mentioned above, we propose an alternative technique for generation of 3D virtual road surfaces. The procedure has two steps: the *rough* surface followed by its *smoothening*. However, the results of the first step may be too far from our ability to manage random data and the second step has to simplify the surface. In the context of 3D surfaces, the simplification means the reducing of the number of points needed to represent the surface. The diagram shown in Fig. 1 illustrates the entire process:

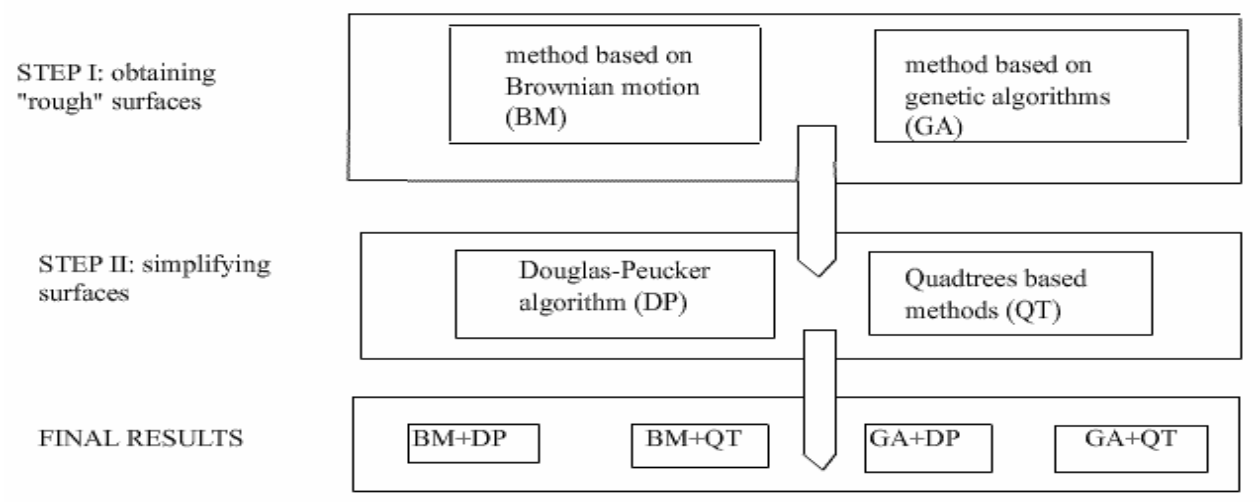


Fig. 1 – The steps and results in generating smooth surfaces.

The surface generating begins with creating a random planar curve. This curve is slightly modified and moved, and after drawing several curves, a surface is obtained. The Brownian motion procedure is based on the random motion of particles suspended in a fluid. In 1D, a function $x(t)$ representing the position of a point at the moment $t$, is obtained under two restrictions: the increment $x(t+\Delta t)-x(t)$ must have Gaussian distribution and the mean square of the increment must have a variance proportional to the time differences [1, 2]. The algorithm gives the broken lines used to generate the 3D surfaces (Fig. 2).

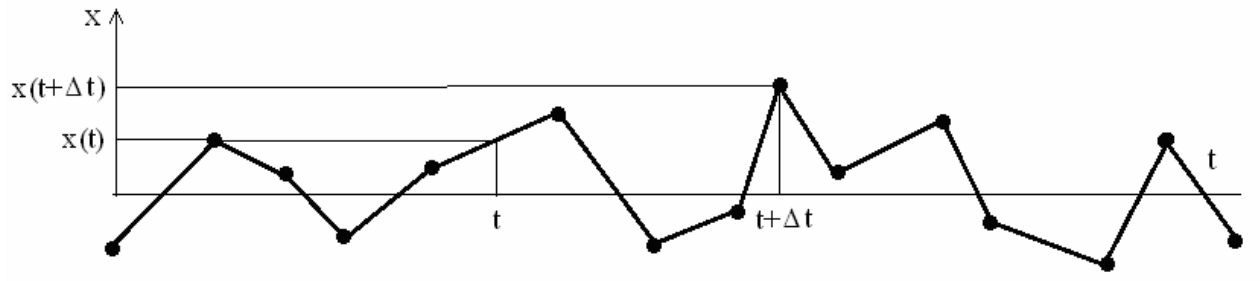


Fig. 2 – Broken line obtained using Brownian motion.

A C++ program creates a matrix $H$ and writes it in a file. This file is imported in MATLAB to generate images similar to those displayed in Fig. 3. Every new line in the matrix is obtained by applying the same algorithm.

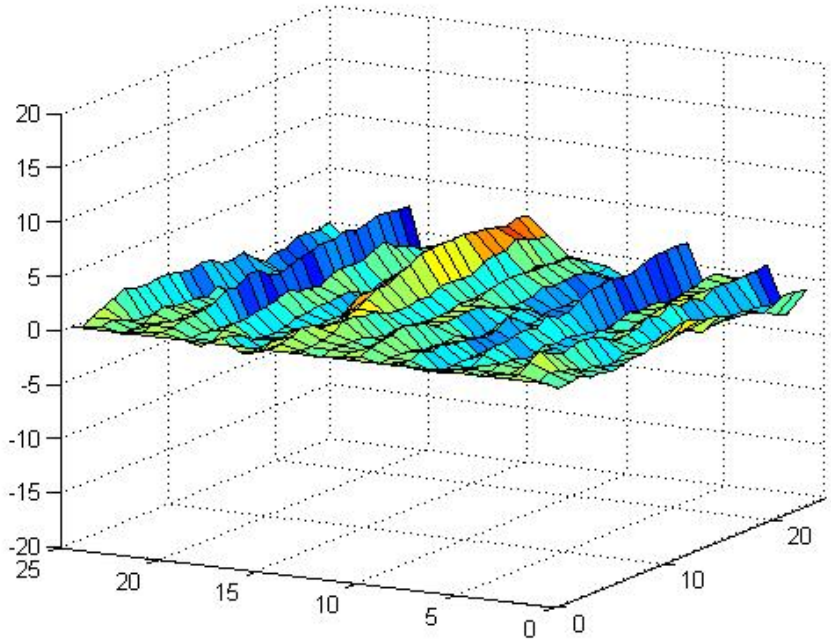


Fig. 3 – Rough surface obtained using the Brownian motion algorithm.

The $H$ matrix has the following form

$$H = \begin{pmatrix} x_1(t_1) & x_1(t_2) & \dots & x_1(t_m) \\ x_2(t_1) & x_2(t_2) & \dots & x_2(t_m) \\ \dots & \dots & \dots & \dots \\ x_n(t_1) & x_n(t_2) & & x_n(t_m) \end{pmatrix}, \tag{1}$$

where $x(t)$ represents the position of a point at the moment $t$.

Genetic algorithms provide another way to obtain planar curves and, ultimately, rough surfaces. The usage of the genetic algorithms in surfaces generation is based on the elementary procedure of deforming a planar line segment and transforming it into a planar broken line. Each breaking point will be referred to as a gene and the structure formed by all the genes as a chromosome. Given a starting point *A*, an ending point *B*, $n$ other points from *A* to *B* have to be placed so that each segment formed by the last two points forms a random angle with the O*x* axis. Starting in *A*, the random direction changes may lead to a final point *B*', different from *B*. Considering the fact that *B* should be the final point, the *AB*' segment has to be scaled and rotated, together with all the other intermediate points so that *B*' equals *B*. The fitness score for an individual broken line is computed as a sum of certain values calculated for all the belonging genes. The values associated with the genes depend on the angle of the direction change and on the smoothness computed into a small enough vicinity [10].

Replacing the old generation with the new one will need specifying how crossover and mutation take place. The previously described technique leads to results shown in Figure 4. There are many situations when genetic algorithms are preferable to Brownian motion based algorithms, mainly because the outcomes can be changed and partially directed and this can be easily done by a proper way to calculate the fitness function.

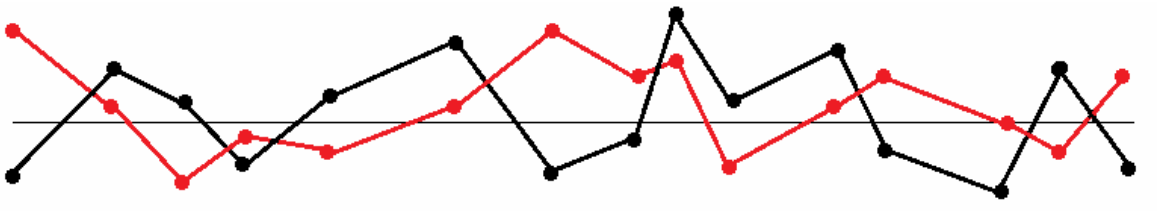

Fig. 4 – Broken line obtained by genetic algorithm (RED line) compared to the broken line obtained using Brownian motion (BLACK line).

The process of simplifying the surfaces is the second important step in generating surfaces that are built upon a grid [1, 2]. This process can be considered the opposite of the first one, as it straightens the surfaces previously generated. This process is done not only to control the processing time, because "it is desirable to use approximations in place of excessively detailed models" [15] but also because it allows the manipulation of a number of additional parameters. Some simplifying methods are based on triangulations, but their applicability in MATLAB is limited, as MATLAB normally uses matrices. The following two methods, the Douglas-Peucker algorithm based method and the quadtrees based method are supposed to be more suitable for MATLAB, since they work with both one dimensional and two dimensional arrays.

The Douglas-Peucker algorithm is a technique used for reducing the number of points in a planar curve that is approximated by a series of points. It was found at the beginning of the seventies and it is used on a large scale for drawing maps. One of the targets of this research is to prove that the Douglas-Peucker algorithm can be applied in the case of the task of simplifying surfaces (in this case a surface is considered to be a number of broken lines one next to another). Briefly, to simplify a broken line determined by the planar points $P_1, P_2, ..., P_n$ one can check if the maximum distance from one of the points $P_2, ..., P_{n-1}$ to the line formed by $P_1$ and $P_n$ (called $d_{\max}$) is smaller than $\alpha |P_1P_n|$, where α is a positive real constant and $|P_1P_n|$ is the length of the segment. If the condition is met, the simplified line is determined only by its endpoints, $P_1$ and $P_n$, otherwise the algorithm is recursively applied for $P_1, P_2, ..., P_k$ and $P_k, P_{k+1}, ..., P_n$ where $P_k$ is the point placed at the maximum distance from the line $P_1P_n$.

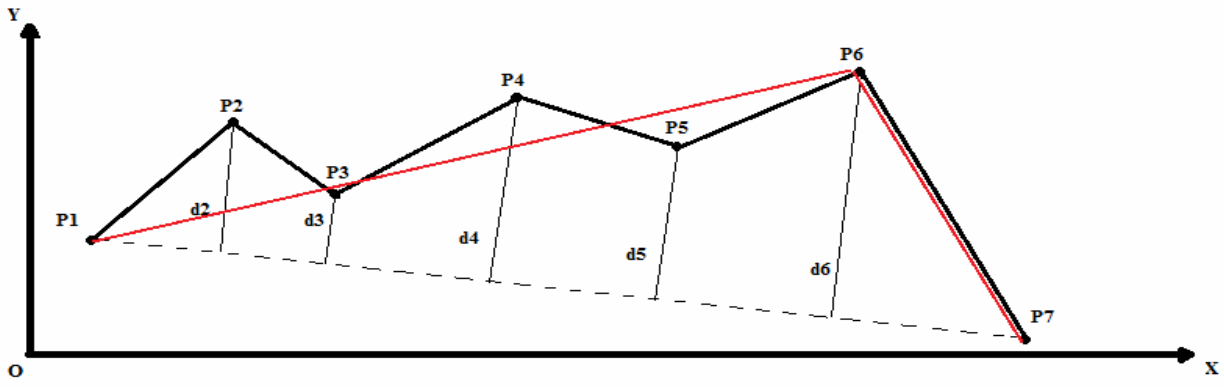


Fig. 5 – Example of simplified planar curve.

Figure 5 illustrates the situation where the Douglas-Peucker algorithm is used. The number of points is $n = 7$. Parameters $d_2, d_3, ..., d_6$ represent the distances from the points $P_2, P_3, ..., P_6$ to the line determined by points $P_1$ and $P_7$, respectively. We note $d_{\max} = \max(d_2, d_3, ..., d_6) = d_6$. The red broken line in Fig. 5 is obtained by running the Douglas-Peucker algorithm on the black broken line.

To determine the length of the planar AB segment the well-known Euclid's formula is used

$$|AB| = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}\,. \tag{2}$$

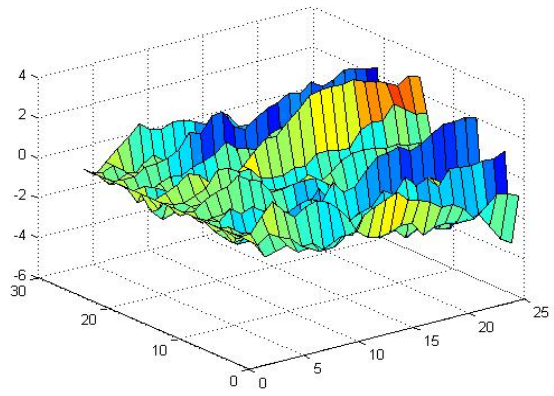

Fig. 6 – Original surface generated by Brownian motion technique.

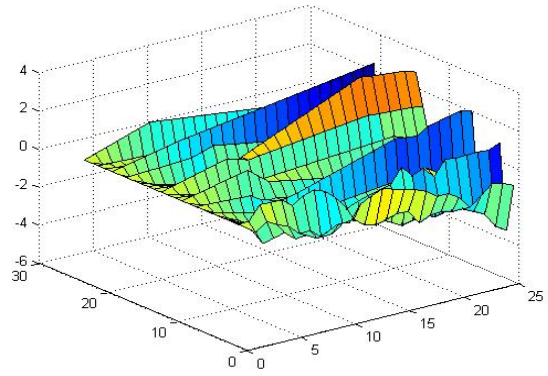

Fig. 7 – Same surface after applying the Douglas-Peucker algorithm.

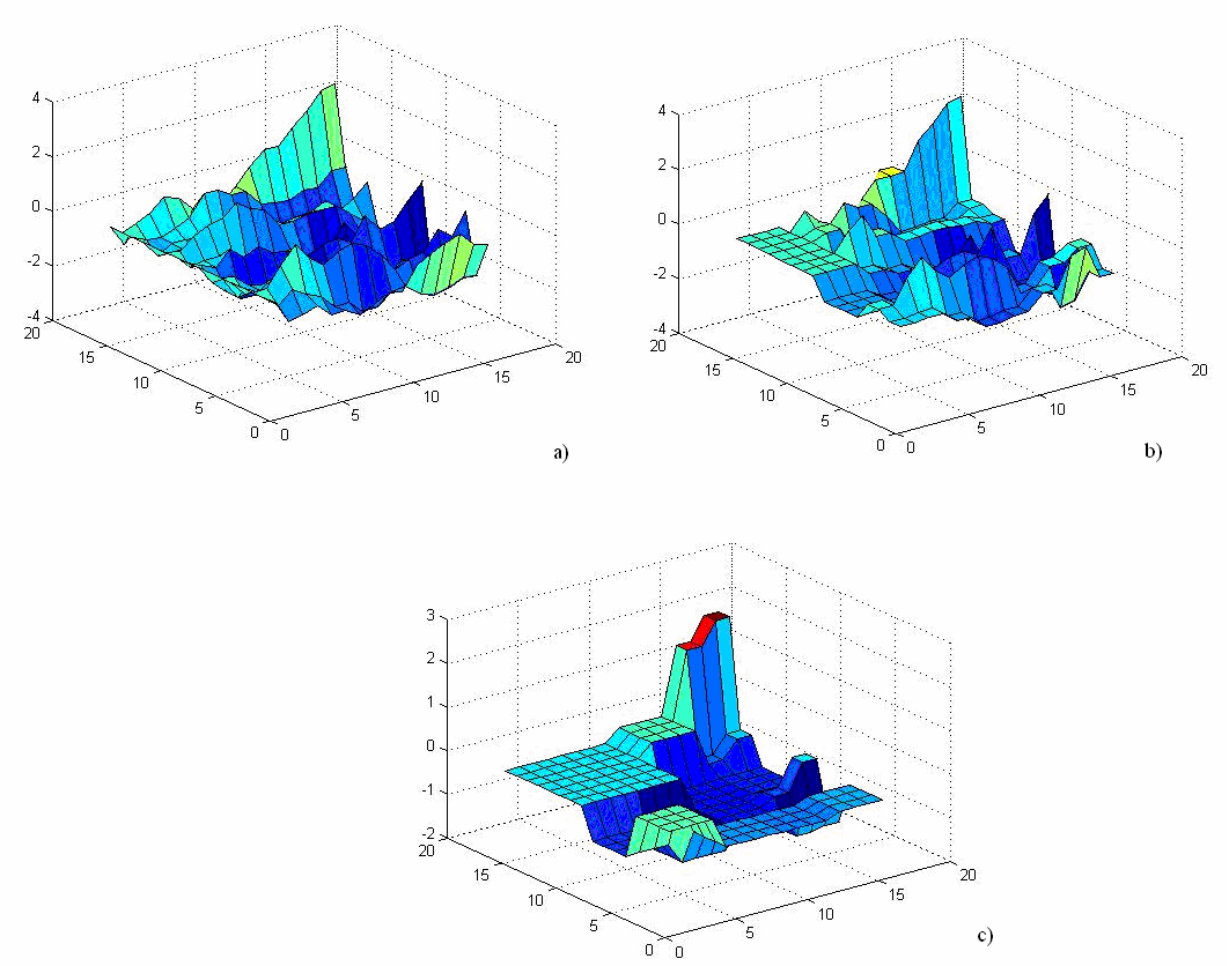


Fig. 8 – a) Original surface; b, c) simplified surfaces using quadtrees with different criteria in accordance to the experimental data.

The equation determined by two planar points, $A(x_A, y_A)$ and $B(x_B, y_B)$ is

$$\frac{x - x_A}{x_B - x_A} = \frac{y - y_A}{y_B - y_A}. \tag{3}$$

or $ax+by+c=0$, where $a=y_B-y_A$, $b=x_B-x_A$ and $c=x_By_A-x_Ay_B$.

The distance from the point $P(x_P,y_P)$) to the line $AB$ is

$$d(P,AB)=\frac{|ax_P+by_P+c|}{\sqrt{a^2+b^2}}. \tag{4}$$

This is applied for each of the lines of the matrix $H$ which contains the heights of the surface that are to be simplified. In Figs. 6 and 7, the same surface before and after applying the simplification algorithm are presented.

In order to compare the results of Douglas-Peucker algorithm, the quadtrees method is next analyzed. Quadtrees has a dynamic data structure, designed to store information about an $2^n\times 2^n$ surface [13]. If the whole surface contains elements of the same height, then the corresponding quadtree is composed of only one vertex, having no descendants and containing the numeric value of the height as specific information. If the surface contains different heights, then the surface is divided into four $2^{n-1}\times 2^{n-1}$ smaller surfaces. Quadtrees is used to simplify surfaces by testing if the difference between the highest point and the lowest point in a region is small enough [10]. Quadtrees method has some properties which make them suitable for some operations such as: changing altitudes, geometrical transformations (especially rotations, changing sizes, reflections, etc) in accordance to experimental data of real roads. Different results can be obtained by using different criteria and parameters for establishing the necessity of the recursive dividing process or stopping it, respectively.

We must specify, even if we not insist on the real roads specimens, that all virtual road construction are based on the real roads and check the minimum set of data which require virtual roads to be very close to the real roads from which we started.

## 3. SURFACES GENERATION USING THE ONION PEELING ALGORITHM

The onion peeling algorithm is a method which can is used to obtain a triangulation of a set of planar points. Let $P$ to be the set of planar points. The convex hull of $P$ is the planar convex polygon, having the minimum area and its vertices belonging to $P$. To determine the convex hull is a well-known issue and several methods were developed over the course of many years [16]. Every new point is compared with the last two points belonging to the stack. If the new point is situated to the left, then it is pushed to the stack, otherwise the top of the stack is popped (Figs. 9 and 10).

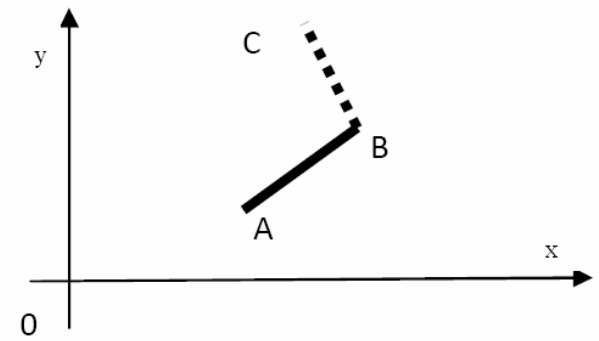


Fig. 9 – Planar point *C* is situated to the left of the line *AB*.

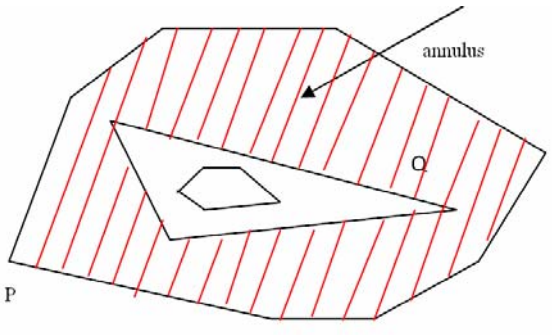


Fig. 10 – Two nested convex hulls, *P* and *Q*, and the annulus between them.

The points that remain to the stack belong to the convex hull. To check whether the planar point $C(x_C,y_C)$ is situated to the left of the planar line $AB$ or not, $A(x_A,y_A)$, $B(x_B,y_B)$ one must check if the following inequality holds

$$(x_B-x_A)(y_C-y_A)-(x_C-x_A)(y_B-y_A)\geq 0. \tag{5}$$

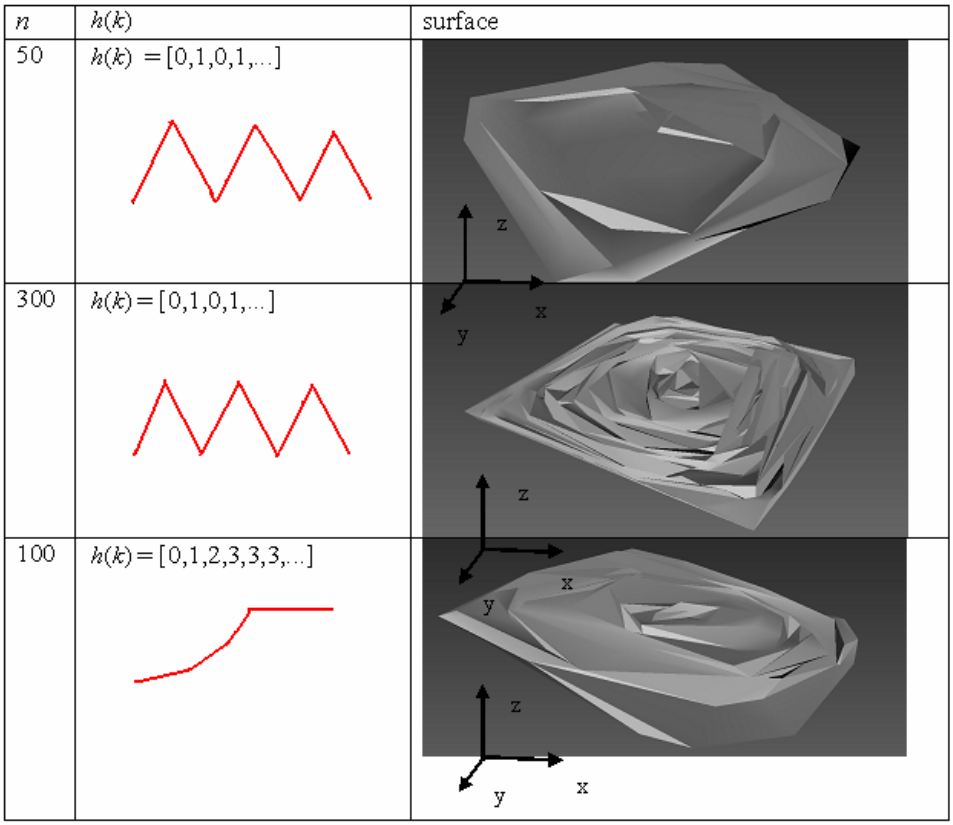


Fig. 11 – Results for different values of $n$ and function $h(k)$.

After the convex hull of the initial set of points $P$ is determined and the corresponding set of vertices $P'$ is established, the same method is applied on $P/P'$ and so forth, until all the points belong to one convex hull. The region between two nested convex hulls is called *the annulus*. About *annulus*, Touissant [18] discovered an ingenious algorithm to triangulate it. His algorithm has as input data two nested convex hulls, *Q* and *P*, *Q* being geometrically included in *P*.

Finally, the method to generate 3D road surfaces by using the onion peeling algorithm was tested by a C++ program. This program firstly generates a set of n random 3D points. All the points are situated in the inner part of a $[0,x]\times[0,y]\times[0,z]$ parallelepiped box in which the $z$ value is significantly less than $x$ and $y$ values. The onion peeling algorithm is then applied over the set of those $n$ points, which finally gives a triangulation of the set of *n* points. In the last step, by using different $h(k)$ functions, all the found triangles are pictured (Fig. 11). We must specify that the best virtual representation which approximates a sample of real road is obtained for $n=300$ and $h(k)=[0, 1, 0, 1, ...]$ in Fig. 11.

## 4. CONCLUSIONS

The first aim of this research is studying and proving the applicability of the Douglas-Peucker algorithm in the second step of the process of generating grid-based road surfaces. Even if its outcomes are different from those obtained with quadtrees method, it has several advantages: the number of mathematical operations is small and the complexity of data structures is a usual one. In addition to this, the Douglas-Peucker algorithm is not limited to $2^n\times 2^n$ working surface, thus making it much more versatile.

The second aim of this research is to find a way to adapt the onion peeling algorithm to the purpose of generating road surfaces. Different methods could be used to connect the different layers. The method based on triangulation allows to the results to be directed to a large number of patterns. We must specify, even if we not insist on the real roads specimens, that all virtual road construction are based on the real roads and

check the minimum set of data which require virtual roads to be very close to the real roads from which we started.

## ACKNOWLEDGEMENTS

This research was elaborated through the PN-II-PT-PCCA-2011-3.1-0190 Project nr. 149/2012 of the National Authority for Scientific Research (ANCS, UEFISCDI), Romania. The authors acknowledge the similar and equal contributions to this article.

## REFERENCES

1. SAUPE, D., *Algorithms for random fractals*, in: H.O. Peitgen and D. Saupe, eds., *The Science of Fractal Images*, New York, Springer Verlag, 1988, pp. 71-136.
2. SAUPE, D., *Point evaluation of multi-variable* random fractals, in: H. Juergen and D. Saupe (eds.), *Visualisierung in Mathematik und Naturissenschaft-Bremer Computergraphik Tage*, Heidelberg, Springer Verlag, 1988, pp. 71-136.
3. JOO ONG, T., SAUNDERS, R., KEYSEr, J., LEGGETT, J.J., *Terrain Generation Using Genetic Algorithms*, 2005.
4. RAMER, U., *An iterative procedure for the polygonal approximation of plane curves*, Computer Graphics and Image Processing, **1**, *3*, pp. 244-256, 1972.
5. DOUGLAS, D., PEUCKER, T., *Algorithms for the reduction of the number of points required to represent a digitized line or its caricature*, The Canadian Cartographer, **10**, *2*, pp. 112-122, 1973.
6. HERSHBERGER, J., SNOEYINK, J., *Speeding Up the Douglas–Peucker Line-Simplification Algorithm*, Proc. 5th Symp. on Data Handling, 1992, pp. 134-143.
7. DUDA, R.O., HART, P.E., *Pattern classification and scene analysis*, Wiley, New York, 1973.
8. VISVALINGAM, M., WHYATT, J.D., *Line Generalisation by Repeated Elimination of the Smallest Area*, CISRG Discussion Paper Series No 10, University of Hull, 1992.
9. O'ROURKE, J., *Computational Geometry in C*, Cambridge University Press, 1993.
10. PAJAROLA, R., *Overview of Quadtree-based Terrain Triangulation and Visualization*, Department of Information & Computer Science, University of California, Irvine, 2002.
11. SAMET, H., WEBBER, R.E., *Hierarchical data structures and algorithms for computer graphics. Part I. Fundamentals*, IEEE Computer Graphics and Applications, **8**, *3*, pp. 48-68, 1988.
12. SAMET, H., WEBBER, R.E., *Hierarchical data structures and algorithms for computer graphics*. *Part II. Applications*, IEEE Computer Graphics and Applications, **8**, *4*, pp. 59-67, 1988.
13. BRIQUET, C., *Introduction to Convex Hull Applications*, Cours d'Algorithmique Avancée (INFO036), Université de Liège, 2007.
14. POULOS, M., MAGKOS, E., CHRISSIKOPOULOS, V., ALEXANDRIS, N., *Secure fingerprint verification based on image processing segmentation using computational geometry algorithms*, Proceedings of the IASTED International Conference on *Signal Processing, Pattern Recognition, and Applications*, Rhodes, 2003.
15. HECKBERT, P.S., GARLAND, M., *Survey of Polygonal Surface Simplification Algorithms*, Report ARPA contract F19628-93-C-0171 and NSF Young Investigator award CCR-9357763., School of Computer Science, Carnegie Mellon University, Pittsburgh, 1997.
16. SKIENA, S.S., *The algorithm design manual*, Second edition, Springer, 2008.
17. GARLAND, M.,HECKBERT, P.S., *Surface Simplification Using Quadric Error Metrics*, Proceedings of the 24th Annual Conference on *Computer Graphics and Interactive Techniques*, pp. 209-216, 1997.
18. * * *, http://cgm.cs.mcgill.ca/~orm/rotcal.html.

*Received November 11, 2014*