Interactive manipulation of sand on a TIN terrain model for virtual reality

Václav Purchart*
Center of Computer Graphics and Data Visualization,
Department of Computer Science and Engineering
University of West Bohemia
Plzen / Czech Republic

Abstract

This paper describes a new approach in modeling of sand surface. The terrain model is based on a triangulated irregular network while existing solutions have been based on a regular grid. Memory consumption is significantly reduced without a regular grid at the expense of more complicated algorithms. Our solution allows deforming terrain by a set of virtual tools and erosion simulation. All these simulations run in real time. We want to use haptic devices with our application in the future.

Keywords: Virtual Reality, Terrain Erosion Modeling, VR Tool, TIN Terrain Model

1 Introduction

Our project has the following goals. We would like to induce the same visual and touch feeling as if the user directly touches the sand in real world with the respect of the sand humidity and the thrust.

At present, we have started to work on the haptic part of this project (together with the Masaryk University in Brno, Czech Republic), however, this paper contains only the geometric, physical simulation and visualization part of our solution.

Our method aims to handle computer modeling of a terrain formed by some granular material such as sand, salt, etc. Our goal was to propose a terrain model in which it will be possible to make direct deformations using virtual tools. The terrain model responds on these and other changes by a material movement – it fills up holes and pits. Let us present a concept of our solution.

The whole terrain surface is represented by a triangulated irregular network (TIN), a triangulation. Triangle vertices describe the terrain height in the given position. The enforcement of terrain changes is done by setting vertex height or adding new vertices to the TIN. We also need to enforce some shape to the TIN because of virtual tools. It is done by constraining edges to the TIN. A tool stroke is simulated by continual enforcement of virtual tool samples.

Erosion modeling is based on a particle system principle. Material moves over the edges from one vertex to another to imitate the behavior of real sand. Depending on the characteristics of the modeled material (e.g. humidity of sand), the material is moving from the above-lying position into the low-lying positions. This causes continuous terrain smoothing.

Model based on an irregular network allows setting any necessary level of detail. The places where the terrain is flat can be described using a few triangles. By contrast it is not necessary to set a fixed level of detail for complex shapes. This approach has significantly reduced the consumption of memory needed for the terrain model. The disadvantage is that more complicated algorithms are needed than for a regular grid. The main novelty of our approach is the use of TIN instead of regular grid. Preliminary version of the algorithm has been published in [10].

2 Related work

In the area of terrain computer modeling many papers have been published. The following work deals with the modeling of granular materials, too. The existing work can be divided into two main streams – the first one focuses primarily on physical correctness. The second stream concerns interactive applications operating in real time.

In the publication [14] the terrain surface is represented as a regular grid. Grid cells represent the terrain height in each point. Working with this data structure is relatively simple and very fast. A major disadvantage, as described in [14], is the size of this data structure. The number of elements in the matrix must correspond to the smallest detail which we want to display. Memory consumption is rapidly growing with the resolution. This method does not work in real time and focuses more on creating the best possible model visualization.

Other publication dealing with the modeling of sand is [8]. It is an interactive application which simulates terrain erosion and interactive manipulation with virtual tools. The terrain is represented as a regular grid. The application contains a particle system allowing to simulate sand pouring.

By contrast, the paper [2] focused more on the use of haptical devices. These devices have force feedback and

*lipop@students.zcu.cz

induce a feeling that the user directly touches the material in the real world. Simulated feedback contains two forces: the first is the penetration resistance of the material to the tool and the second is the friction that is caused by tool movement.

3 The proposed method

In this section we describe in detail the proposed method – the manipulation of sand on a TIN terrain model. The described solution has been created within the bachelor thesis of V. Purchart [9], J. Kadlec [7] and master thesis of J. Sedmihradský [11]. The method is further improved by the author of this paper.

The model is based on a triangulated irregular network, a triangulation which represents the terrain surface. It is formed by a list of vertices coordinates, a list of triangle vertices and a list of pointers to the triangle neighbors. Each vertex *X* has real coordinates *x*, *y*, *z*, where *x* and *y* is the location in the terrain model and *z* is the height in the given location. It follows that the terrain shape must be a function of two variables $z = f(x, y)$. The model does not allow modeling of overlapping and tunnels. For sandy terrain this model is sufficient.

3.1 Geometric part

At first, we define how the model is represented exactly. Then we describe the basic operations such as adding / removing the vertex, constraining the edge, getting and setting a height in any location and finding a position of the triangle in the TIN.

A triangulation *T(P)* of a set *P* of *N* points in the Euclidean plane is a maximum set of edges *E* such that:

- no two edges in *E* intersect at a point not in *P*,
- the edges in *E* divide the convex hull of *P* into triangles.

The edges of each triangle in the TIN are required to be similarly long. Triangles should be as much as possible close to equiangular triangles. Delaunay triangulation provides such triangles. Material movement looks more realistic on such a triangulation.

The Delaunay triangulation in $\Re^2$ of the set of vertices *P* is a triangulation *DT(P)* in which each triangle defined by the vertices $A, B, C \in P$ meets *The Delaunay empty circle criterion* for each vertex *D*, where $D \in P \wedge D \notin \{A, B, C\}$ (Equation 1). Vertex *D* is not in the circle defined by the triangle *ABC*, when $d < 0$ (the criterion is met).

$$d = \begin{vmatrix} A_x - D_x & A_y - D_y & (A_x - D_x)^2 + (A_y - D_y)^2 \\ B_x - D_x & B_y - D_y & (B_x - D_x)^2 + (B_y - D_y)^2 \\ C_x - D_x & C_y - D_y & (C_x - D_x)^2 + (C_y - D_y)^2 \end{vmatrix} \quad (1)$$

There are many Delaunay triangulation algorithms. Our choice is the incremental insertion algorithm because we need to modify the completed TIN (a more detailed description see in [3]). When we add a new vertex to the TIN, the triangle which includes the new vertex must be found. We divide this triangle into new ones. We adjust the rest of the TIN by edge flipping to meet the triangulation criterion.

For further application it is necessary that the TIN allows direct enforcement of some edges. For edge constraining we use the Constrained Delaunay triangulation (*CDT*). This triangulation method is based on the basic Delaunay triangulation, but, moreover, there are "constrained edges". These edges are required in the TIN even if they do not meet the criterion of Delaunay empty circle (Equation 1). This allows enforcing direct changes into the model (see Figure 1).

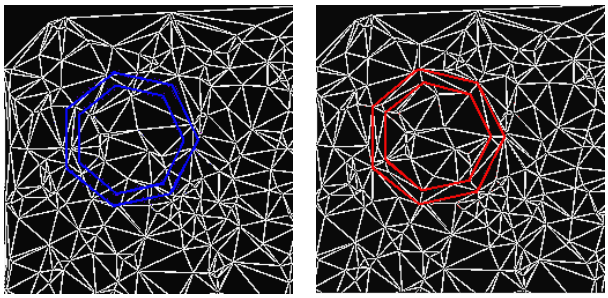

Figure 1: CDT in action. On the left we can see the proposed edges which should be constrained. On the right is the same TIN with constrained edges included.

Algorithms for the constraining of the set of edges are described in [1], [4] and [12]. Our algorithm is based on [12]. At first we construct the basic *DT* and than we constrain the required set of edges. Continuously we flip existing edges which intersect with the constrained edge. Finally we can insert the constrained edge to the TIN without an intersection with any existing edge.

Removal of vertices

Removal of vertices is necessary due to erosion, some parts of the terrain model may become flat and unnecessary vertices in such places are not needed any more (see Figure 2). They do not bring a substantial shape information but slow down the computation. Vertices removal is based on the algorithm [5]. At first, the vertex is removed from the TIN and then the resulting hole is retriangulated. So it will become again the Delaunay triangulation. After vertex removal we get a star-shape polygon in the worst case. By cutting ears of this polygon we eliminate the hole.

Unnecessary vertices can be relatively easy to detect. If the vertex is, due to its neighbor vertices, (approximately) in the plane, we do not loose any detail from the model by removing the vertex. At first we get the maximum angle between each join from the removed (center) vertex to each of its neighbor vertex (i.e. vertices which are connected by the edge) and to the horizontal line defined by the analyzed vertex. If the maximum angle is close to zero, the center vertex is redundant and can be removed from the TIN. It is impossible to check all vertices in the TIN for the criterion above, because

there can be a huge amount of vertices and the time needed for it would be prohibitive. We use a queue, which accumulates vertices that could be redundant, such as the vertices which have changed the height in the last iteration somehow. If the above criterion is met, the vertex is marked as redundant and before the completion of iteration it is removed from the TIN.

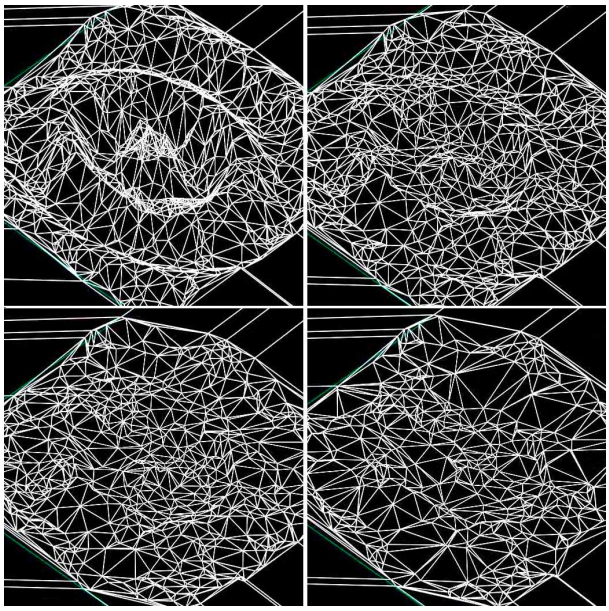

Figure 2: The automated simplification of the TIN. The top-left image shows the original TIN. The bottom-right image shows the simplified TIN which is more flat than the original due to erosion.

Point location

The most used geometric operation is a location of the triangle in which the given (target) point lies. Because of this the location of the triangle has to be effective. The TIN is permanently changing. Due to permanent changes, the use of complex search structures is not too efficient and is memory demanding. We used the algorithm of rectangular walk [13] which is suboptimal, but it does not need any special data structure, only information about neighboring triangles of each triangle.

3.2 Erosion modeling

In real life there are many types of erosion – such as water erosion, erosion caused by material temperature changes and others. The result of these processes is that the material is moved from higher-lying places to lower-lying places. The amount of the moved material depends on the height difference. We simulate this kind of erosion. The accurate physical models require more computation time. Therefore the simulation is not based on an accurate physical model because of interactivity of the whole process.

The set of vertices P of the triangulation *CDT(P)* and the set of edges E define a graph. The erosion simulation works iteratively. For each vertex $P_k \in P$ we find all vertices P_j, where the edge $E_i \in E$ from P_k exists. For each pair P_k, P_j we compute the angle. This angle is formed by a horizontal line going through P_k with an edge between P_k, P_j. If this angle is lower than the "critical angle" of the simulated material then the material transfer is realized from the vertex P_k to each neighbor vertex P_j, which lies lower than P_k (see Figure 3). For a dusty sand the critical angle is about 30°, see [11].

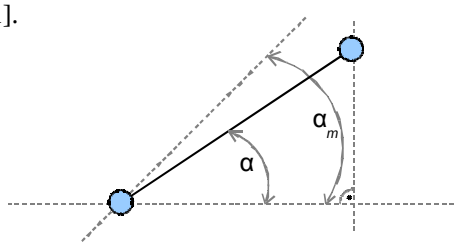

Figure 3: The material movement is stopped if an angle α is less than the critical angle α_m.

Browsing and eroding of all vertices in the TIN would be infeasible. Only the changed vertices are eroded where a material moving is possible. These vertices are saved to a special list L_e. A vertex must be added to the L_e if its height has been changed in the last iteration. The vertex height can be changed by a virtual tool or the erosion algorithm. The vertex can be removed from the L_e if its height had not been altered by erosion in a few last iterations.

Erosion methods

In the second phase of the algorithm we change the height of vertices (details in [11]). The height of the vertex is reduced and the height of some of its neighbors is increased. From the physical point of view, the essential erosion characteristic is the volume of the terrain. After the material movement the volume of the terrain should be the same as before. TIN has no "volume", but we can imagine a solid object which will be created by a terrain surface and some bottom plane.

3.3 VR deformations

Terrain deformations are done using a set of virtual tools [9]. These tools allow both simple contour constraining and pulling. VR tools are based on the *CDT* principles. We achieve a real sand deformation sensation by a managed constraining set of edges. Tool pulling is done as a periodical stamping of tool contour at fixed time intervals.

The basic VR tool consists of two parts – the set of outer edges (*O*) and the set of inner edges (*I*). The inner and outer edges cannot be at the same *(x, y)* position as we cannot represent vertical edges in our model. Therefore, the inner and outer edges are in a small mutual distance and are never completely identical in the projection, see Figure 4. By a proper constraining of both sets of edges in the TIN we achieve one tool stamping.

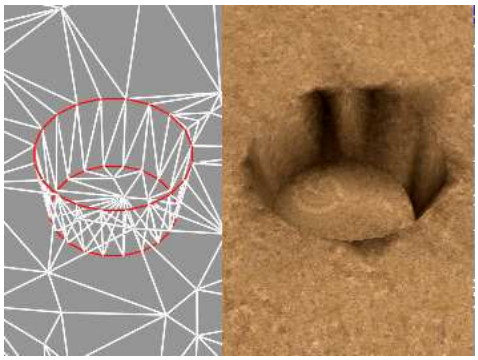

Figure 4: Stamping of a VR tool. On the left we can see the wire-frame model (red lines are constrained edges). On the right we can se the textured model. Both are without erosion; taken from [11].

The inner part of the VR tool

The purpose of the inner part of the VR tool is to deform the terrain in a required way. Every stamping by the VR tool changes the height of the terrain according to the depth level of the virtual tool. Because of this, the inner part is a simple plane in most cases. Vertices located in this area are redundant and can be deleted from the TIN. Removal of these vertices can be done by an automatic TIN simplification which is described below. This requires a little bit more computational time. We walk through all triangles inside the area and check these vertices. The vertex P_i is redundant if:

- the vertex depth is close to the tool depth level
  and
- the removed vertex does not belong to the set of border vertices
  and
- the vertex has no neighbor, whose depth is greater than the tool depth level.

The TIN can also contain edges from the previous stamping. Each set of edges has a unique number, which can determine the "generation" of each constrained edge. If a constrained edge is discovered during walking through the inner triangles, it can be deleted because it belongs to some old stamping.

The outer part of the VR tool

The outer part of the VR tool adapts to the contour of the surrounding terrain. This affects how real the stamp looks. We interpolate the height of vertices in the outer set O by a simple linear interpolation of the existing vertices. After this we constrain all edges from the set O. This gets a good result in most cases. For more realistic stamping it is necessary to adapt the outer part accurately. A good approximation is obtained by the subdivision of edges to smaller pieces which are again adapted to the height of a terrain.

It is important, in which order we constrain the parts of the VR tool. At first the outer part must be constrained into the terrain. This part adapts to the terrain. After that the inner part is constrained and makes changes "by force" in the terrain according to tool properties.

Extensions

More complex tools can be derived from the above basic tools. For a realistic appearance it is necessary to push away the material. Volume of this material should be the same as a volume of the created hole. Details of the approximate solution can be found in [11].

4 The complete algorithm

Let us first explain the meaning of the symbols in our algorithm:

L_e is the list of vertices which will be eroded.

L_r is the list of vertices used for the detection of the redundant vertices.

α_{flat} is the value of a limit angle to what extent the vertices of the nearly-flat triangles are removed from the terrain model. If α_{flat} is higher then more vertices are removed and vice versa. A proper choice of α_{flat} is addressed in Section 5.

The whole algorithm can be simply described as follows:

- Computing of the Delaunay triangulation of the initial set of vertices.
- For each iteration:
  1. Constraining deformations into the terrain on the current location of the VR tool:
     a) The height interpolation of the control points which are in the outer part of the VR tool.
     b) Constraining the set of edges of the outer tool part using *CDT*.
     c) Constraining the set of edges in the inner tool part.
     d) The height reduction in the inner VR tool part.
     e) Schedule removal of redundant vertices.
     f) Adding the changed vertices to the lists L_e and L_r.
  2. Material erosion; for each vertex P_k in the list L_e:
     a) Computation of the height difference between the vertex P_k and all its neighbors P_i (there is an edge from P_k to P_i) and choosing of the best erosion method.
     b) Computation of the material gain for the lower-lying vertices.
     c) Height reduction of the vertex P_k, height increase for the lower-lying vertices P_i.
     d) Adding changed vertices to the lists L_e and L_r,
     e) The exclusion of the vertices from L_e for which there was no material transfer.
  3. For each vertex P_l from the list L_r:
     a) The test of the angle α_i between the edge of the vertex P_l and each its neighbor P_i and a horizontal line passing through the point P_l.
        I. If $\alpha_i > \alpha_{flat}$ then removal of the vertex from the list L_r.
        II. If $\alpha_i \leq \alpha_{flat}$ then for all vertices P_i, removal of the vertex P_i from the list L_r, from the list L_e and from the TIN.

5 Experiments and results

To verify the approach with a triangulated irregular network, we have created an application in C++. In the next phase it is planned to use haptical devices, so the application must be portable to other operating systems. A model visualization is done using the OpenGL library, the graphical user interface and system communication uses the multiplatform SDL library. The application is operated on Windows XP, Windows Vista and Linux. All measured data are related to a computer with Windows Vista, 2GB RAM, Intel Core 2 Duo 1.8 GHz, NVIDIA GeForce 8600M GS, 256MB.

We experimentally set the angle α_{flat} to value 0.4 (on the basis of the experiments in [11]). This brings the best results. If the angle is lower, then the terrain model is too complex, otherwise the terrain model is close to the flat surface and a lot of details have been lost.

In Figure 5 and Figure 6, we can see a comparison of solutions using regular grid and TIN. We tried to imitate the experiment and illustration from [2] to compare visual quality of our solution and [2].

Figure 5: Our method based on a TIN

Figure 6: Result on regular grid; taken from [2].

From the figures we see that due to the long edges and probably not the optimal shading model our result in Figure 5 looks a little worse. One of the future challenges is to achieve a more realistic appearance. Possibilities are as follows:

- a better shading model,
- a subdivision of the long edges which have negative impact on shading,
- a use of randomization in the sand movement simulation to achieve less regular appearance.

At this moment, we are unable to compare with [2] as to the performance as the program obtained thanks to the authors of [6] do not work on our OS but we suppose to solve it in the near future. We at least measured the performance of our solution without comparison.

In the first experiment, we measured the computation time of the main algorithm parts dependency on the TIN size or the length of tool trace. The graph in Figure 7 shows that the vertex removal is the most time-consuming part of the whole algorithm. This time consists mainly of the Devillers vertex removal algorithm [5]. The rest of time is spent by the redundant vertices detection. The used triangle walking algorithm implies the measured complexity $O(n^{0.5})$. It is one of the future work on the project to reduce this time. Calculation of the erosion model has complexity around $O(n^{0.5})$.

In the second experiment, we measured the total time consumption in dependence on the length of tool trace. The graph in Figure 8 shows the time consumption in dependence on the length of tool trace for TIN with 10000 vertices. Complexity estimate is $O(n)$ for the erosion model and $O(n^{1/4})$ for the vertex removal. Time complexity is getting better with the size of the TIN.

In all experiments we set the fixed TIN size or the length of tool trace and measure the other parameters of the model. The trace was created along the diagonal of the modeled terrain. Time data represent the accumulated time during the whole tool pulling.

We also analyzed the number of vertices after the simulation dependency on the trace length and the triangle count affected by the VR tool dependency on the trace length (Figure 9). These dependencies are nearly linear.

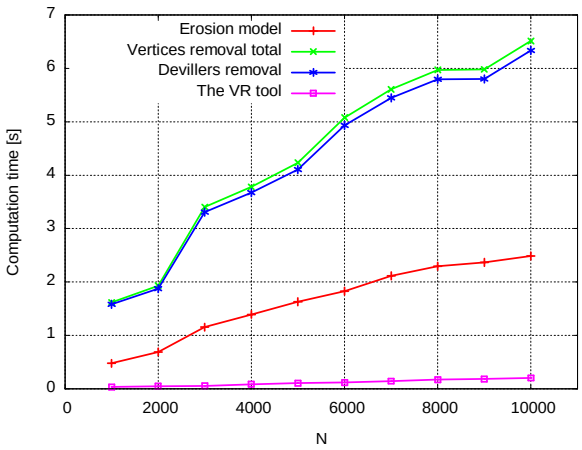

Figure 7: Time consumed by each part of computation in dependency on the TIN size (the trace length is 0.55; the whole model diagonal length is 1).

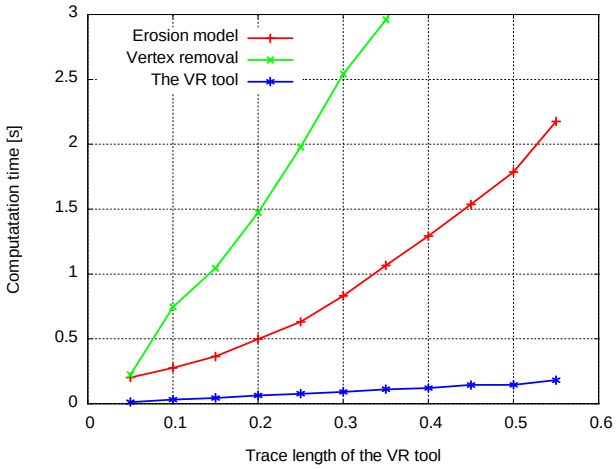

Figure 8: Time consumed by each part of computation in dependency on the trace length (the TIN size is 10000 vertices).

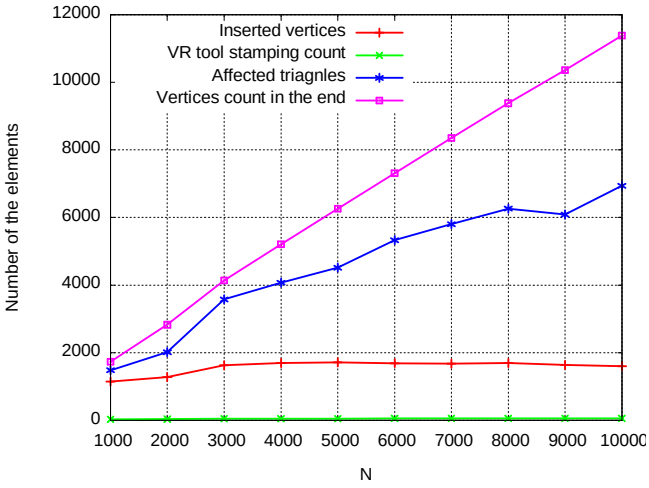

Figure 9: Number of elements in the TIN in dependency on the TIN size.

The graph in Figure 10 shows the number of elements in the TIN dependency on the trace length. Also these dependencies are nearly linear.

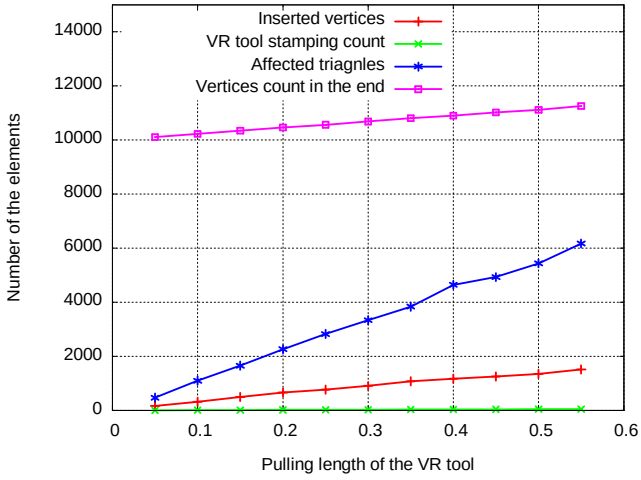

Figure 10: Number of the elements in the TIN dependency on the trace length.

6 Conclusion

Our method based on the triangulated irregular network as a terrain model can be used in practical applications. Further work will aim at the appearance improvement of the model and optimization of the vertices removal. At present we started to collaborate with Masaryk University in Brno in using haptical devices with our application. This will allow a real touching feeling of the granular material. It will be possible to create a sort of virtual sandbox. The application will be extended with more accurate physical models. There is also an effort to map the whole TIN onto 3D objects. Other possibilities would be to simulate various physical processes. The implemented erosion algorithm can be easily replaced by another physical model, which will operate on the existing geometric model of the terrain. There is a future possibility that the application would be used for e.g. visually impaired people because of simulation of touch sensation.

Acknowledgements

I would like to thank to Doc. Dr. Ing. Ivana Kolingerová (University of West Bohemia, Czech Republic) for inspiration and help with solving of this problem. Next I would like to thank to my colleagues Jan Kadlec, who created the geometric model used, and Jiří Sedmihradský, who is the author of the erosion simulation and model visualization. Both participated in the development of the first phase of the project. My thanks go also to Ing. Bedřich Beneš, Ph.D. (University of Purdue, USA) for his help with visualization and erosion modeling. I thank also to anonymous reviewers for valuable comments which enabled to improve the paper and brought inspiration for future work.

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic, project Kontakt No. ME09051.

References

[1] Anglada, M., V.: *An Improved Incremental Algorithm For Constructing Restricted Delaunay Triangulations*. Computers & Graphics, Vol. 21, No.2, pp. 215-223, 1997.

[2] Beneš, B., Dorjgotov, E., Arns, L., Bertoline, G.: *Granular material interactive manipulation: Touching sand with haptic feedback*. In Proceedings of the 14-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2006, pp. 295–304, 2006.

[3] de Berg, M., van Kreveld, M., Overmans, M., Schwarzkopf, O.: *Computational Geometry: Algorithms and Applications*. Berlin: Springer-Verlag, 1997.

[4] Chew, L., P.: *Constrained Delaunay triangulations*. Proceedings of the third annual symposium on Computational geometry, pp. 215–222, 1987.

[5] Devillers, O.: *On Deletion in Delaunay Triangulation*, Annual Symposium on Computational Geometry, pp. 181-188, 1998.

[6] Enkhtuvshin, D., Beneš, B.: *Stereo Sand Drawing*. The sandy terrain simulation program based on regular grid. Received on the 10th November 2008.

[7] Kadlec, J.: *Terrain deformations for virtiual reality – geometric model part.* Bachelor thesis. University of West Bohemia, Pilsen, 2007 (in Czech).

[8] Onoue, K., Nishita, T.: *Virtual Sandbox*, 11th Pacific Conference on Computer Graphics and Applications (PG'03), pp. 252–259, 2003.

[9] Purchart, V.: *Terrain deformations for virtual reality – basic structures, control layer and model visualization*. Bachelor thesis. University of West Bohemia, Pilsen, 2007 (in Czech).

[10] Purchart, V.: *The sandy terrain modeling for virtual reality*. ACM Student Research Competition. Prague, 2008 (in Czech).

[11] Sedmihradský, J.: *Terrain erosion and deformations modeling*. Master thesis. University of West Bohemia, Pilsen, 2007 (in Czech).

[12] Sloan, S.W.: *A Fast Algorithm for Generating Constrained Delaunay Triangulations*, Computers and Structures, Pergammon Press Ltd., Vol 47, No. 3, pp. 441–450, 1993.

[13] Soukal, R.: *Application of triangle walking algorithm in computer graphics*. Master thesis. University of West Bohemia, Pilsen, 2008 (in Czech).

[14] Sumner, R. W., O'Brien, J. F., Hodgins, J. K.: *Animating Sand, Mud, and Snow*. Proceedings of Graphics Interface '98, Vancouver, B.C., Canada, June 17-21, pp. 125-132, 1998.