[2]. Metody klassicheskoy i sovremennoy teorii avtomaticheskogo upravleniya: Uchebnik v 5-i tt.; 2-ye izd., pererab. i dop. T.2: Statisticheskaya dinamika i identifikatsiya sistem avtomaticheskogo upravleniya / Pod red. K.A. Pupkova and N.D. Egupova. – M.: Izdatel'stvo MGTU im. Baumana, 2004. 640s.
[3]. Tekhnologi konstruirovaniya sovremennykh konkurentosposobnykh kompleksov upravleniya stokhasticheskim dvizheniyem ob"yektov: Monografiya / L.N. Blokhin, S.I. Osadchiy, A.K. Didyk i dr. – Kirovograd: izdatel'-Lisenko V.F., 2015. 284s.
[4]. Osadchyy S.I. Avtomatyzatsiya dynamichnoho proektuvannya optymal′nykh bahatovymirnykh robastnykh system stokhastychnoyi stabilizatsiyi. Konstruyuvannya, vyrobnytstvo ta ekspluatatsiya sil′s′kohopodars′kykh mashyn: Zahal′noderzhavnyy mizhvidomchyy naukovo-tekhnichnyy zbirnyk, vypusk 40. ch.1 - Kirovohrad, 2010. S.25-34.
[5]. Osadchiy S.I. Nerekursivnyy kombinirovannyy algoritm faktorizatsii polinomial'nykh matrits. Nauchno-tekhnicheskiy zhurnal «Aviatsionno-kosmicheskaya tekhnika i tekhnologiya» №6(63). 2009. S. 54-59.
[6]. Osadchy Sergei I., Zozulia Valerii A. Passive identification of multivariable stabilization system elements' dynamics. Automation of Technological and Business Processes. – 2020. Vol. 12, Issue 1. pp. 32 – 40. https://doi.org/10.15673/atbp.v12i1.1701
[7]. Korn G., Korn T. Spravochnik po matematike (dlya nauchnykh rabotnikov i inzhenerov). Per. s angl. I.G. Abramovicha i dr. M.: Nauka, 1977. 831s.
[8]. Aliyev F.A., Bordyug V.A., Larin V.B. Vremennyye i chastotnyye metody sinteza optimal'nykh regulyatorov. Baku: In-t fiziki AN Azerbaydzhanskoy SSR, 1988. 46s.
[9]. Debni Dzh., Kharman T.L. Simulink 4. Sekrety masterstva. Per. s ang. M.L. Simonova. M.: BINOM. Laboratoriya znaniy, 2003. 403 s.
[10]. Rozrobka fizychnoyi modeli verstata na osnovi mekhanizmu paralel′noyi struktury z systemoyu keruvannya pryvodamy peremishchennya robochoho orhana: Zvit po NDDKR Kirovohrads′kyy natsional′nyy tekhnichnyy universytet. № DR 0109U00210, oblik. № 0211U005056. Kirovohrad, 2011. 176s.
[11]. Broshyura po beskontaktnym enkoderam L-9517-9494-01-A. URL: http://www.koda.ua/download/Brosura._Enkodery_Renishaw._Azyk_-_russkiy..pdf (data zvernennya: 11.12.2021)

Отримана в редакції 08.11.2021. Прийнята до друку 29.11.2021. Received 08 November 2021. Approved Approved 29 November 2021. Available in Internet 04 December 2021.

---

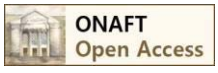
**УДК 004.925.4**

# MIP MAPPING THE VIRTUAL ENVIRONMENT FOR COMPUTER GAMES

**Mykhaylov P.[1], Chekhmestruk R.Y.[2], Romanyuk O.N.[3], Kotlyk S.V.[4]**

[1]CEO 3D GNERATION GmbH, Dortmund, Germany, [2]3D GENERATION UA, Vinnytsia, Ukraine, [3]Vinnytsia National Technical University, Vinnytsia, Ukraine, [4]Odessa National Academy of Food Technologies, Odessa, Ukraine
ORCID: [1]https://orcid.org/0000-0001-5861-5970, [2] https://orcid.org/0000-0002-5362-8796, [3] https://orcid.org/0000-0002-2245-3364, [4] https://orcid.org/0000-0001-5365-1200
E-mail: [1]pm@3dgeneration.com, [2]Rc.ua@3dgeneration.com, [3]rom8591@gmail.com, [4]sergknet@gmail.com






**DOI: 10.15673/atbp.v13i4.2220**

***Abstract.*** *The paper describes how MIP-mapping and paging can be used to represent not only terrain imagery, but also terrain elevation. Previously the only things missing to implement Earth coverage were computing power, input/output bandwidth, graphics processing units (GPUs) and techniques to deal with large data sets. The article describes the rendering method that uses the GPU for most of the calculations. Modern graphics accelerators for personal computers allow you to solve problems that require the generation of images of the visual environment of photographic quality in real time. High-quality visualization of large continuous spaces of 3D vegetation is one of the requirements when creating most land transport simulators, as well as many virtual reality applications. Representation of large forest spaces in the form of free-standing tree models that have an acceptable visual quality for close-up display from the observer, it is impossible due to the huge number of primitives required for visualization. The article proposes use parallel calculations in GPU to accelerate rendering. We successfully integrated proposed visualization method into the standard rendering pipeline. For considered tests the*

*application with GPU average ten times faster, than the version using only CPU. The proposed method takes into account the features of the architecture of modern GPUs and allows you to distribute the load on the display between the CPU and GPU. In this case, the method does not require significant resources for visualization. The implementation of the method showed that received speed in most cases is sufficient for the effective application of the method in computer games. The paper also proposes a method for high-quality visualization of large continuous spaces of 3D vegetation.*

***Аннотація.*** *У статті описується, як MIP-текстурування та пейджинг можуть бути використані для представлення не тільки зображень місцевості, але й висоти рельєфу місцевості. Раніше для реалізації покриття землі не вистачало лише обчислювальної потужності, пропускної здатності введення/виведення, графічних процесорів (GPU) і методів роботи з великими наборами даних. У статті описується метод візуалізації, який використовує графічний процесор для більшості обчислень. Сучасні графічні прискорювачі для персональних комп'ютерів дозволяють вирішувати задачі, що вимагають генерування зображень візуального середовища фотографічної якості в режимі реального часу. Якісна візуалізація великих безперервних просторів 3D рослинності є однією з вимог при створенні більшості симуляторів наземного транспорту, а також багатьох додатків віртуальної реальності. Представлення великих лісових просторів у вигляді окремо стоячих моделей дерев, які мають прийнятну візуальну якість для відображення з боку спостерігача крупним планом, неможливо через величезну кількість примітивів, необхідних для візуалізації. У статті пропонується використовувати паралельні обчислення в GPU для прискорення візуалізації. Ми успішно інтегрували запропонований метод візуалізації в стандартний конвеєр візуалізації. Для розглянутих тестів програма з GPU в середньому в десять разів швидше, ніж версія, що використовує тільки CPU. Запропонований метод враховує особливості архітектури сучасних графічних процесорів і дозволяє розподілити навантаження на дисплей між CPU і GPU. У цьому випадку метод не потребує значних ресурсів для візуалізації. Реалізація методу показала, що отриманої швидкості в більшості випадків достатньо для ефективного застосування методу в комп'ютерних іграх. Також у роботі запропоновано метод якісної візуалізації великих безперервних просторів 3D рослинності.*

**Keywords:** mip-mapping, height maps, irregular mesh, vegetation.
**Ключові слова:** MIP-текстурування, карти висот, нерегулярна сітка, рослинність

## 1. Introduction

There have been remarkable advances in image generation since paper "Pyramidal Parametrics" [1] – or have there been? Although paper [1] is best remembered for its introduction of texture MIP-mapping, a method intended to increase image-generation speed and reduce rendering artifacts; it also introduced pyramidal data structures, parametric interpolation, highlight anti-aliasing and levels of detail in surface representation. MIP is an acronym for multum in parvo – Latin for "much in little".

The paper [1] planted the seeds and set everything in place to allow image generators to process and render the entire Earth's surface, imagery and terrain skin. At the time the paper was written, the only things missing to implement Earth coverage were computing power, input/output bandwidth, graphics processing units (GPUs) and techniques to deal with large data sets.

One such technique was put forth in paper [2]. The paper proposed a method to structure very large textures such that an image generator could page –in- only the visible portion of the texture's entire MIP-map, simplifying the texture paging process and enabling the creation of large, geospecific-imagery-based terrain databases.

Modern graphics accelerators for personal computers allow you to solve problems that require the generation of images of the visual environment of photographic quality in real time. High-quality visualization of large continuous spaces of 3D vegetation is one of the requirements when creating most land transport simulators, as well as many virtual reality applications. Representation of large forest spaces in the form of free-standing tree models that have an acceptable visual quality for close-up display from the observer, it is impossible due to the huge number of primitives required for visualization. The representation of large areas of the forest, in the form of a textured surface, which is acceptable for visualizing a forest located far from the observer, becomes unacceptable as you approach it. Thus, it is necessary to have a continuous representation of the 3D vegetation visual model, which allows, on the one hand, displaying them in real time using modern graphics accelerators, and having, on the other hand, an acceptable visual quality at all levels. the visualization space. As well in this paper, we present a method for high-quality visualization of large continuous spaces of 3D vegetation, which allows you to evenly distribute the visualization load between the CPU and GPU.

## 2. Triangulated Irregular mesh

Most GPU-s have used MIP-mapping and derived techniques only to texture terrain and three-dimensional models. They also offer various paging schemes allowing the efficient rendering of large, high fidelity, geospecific-imagery-based terrain databases. Most do not, however, use MIP-mapping techniques to create and render terrain skins. There are two major reasons for this: First, it is often believed that image-generator performance is directly proportional to the number of polygons in a rendered scene; second, to enhance the accuracy of the terrain, 2-D features such as roads, rivers and lakes are integrated in the terrain.

To reduce the number of rendered polygons, current state-of-the-art visual databases are built using a tiled triangulated irregular network to represent their terrain skin. Triangulated irregular network are created using digital elevation maps constrained by 2-D vector information in the form of point features, lineal features and aerial features. In those databases, terrain is usually stored as tiled blocks with few levels of detail – typically, three or four. Triangulated irregular network have

the advantage of accurately positioning terrain peaks and valleys. They also allow fine resolution and coarser resolution features, such as winding rivers on rolling hill terrain, to coexist while maintaining a low polygonal count.

Triangulated irregular network has major drawbacks, however. It is difficult to seamlessly connect the different terrain levels of detail without creating cracks in the terrain. Some terrain creation techniques force the terrain to "pop" between levels of detail. Other techniques have the levels of detail blend in through transparency gradation. These techniques drastically increase the polygonal count and create unsightly artifacts, such as sudden terrain shifts and imagery blur, in the rendered scene. Constrained triangulated irregular network terrain is created using algorithms derived from Delaunay triangulation [3]. These triangulation techniques are computationally intensive and cannot be performed at runtime as part of an image-generator operation.

**3. MIP-mapped Terrain**

Today, most image generators are built using commercially available GPUs, which can process polygons at a rate exceeding hundreds of millions of polygons per second. There is a catch, however. These extreme rates can sustained only if the texture, color and material are the same for all polygons – that is, if all polygons are rendered within the same graphic state. The rendering rate is reduced to 100,000 polygons per second if each polygon is rendered in its own graphic state. Thus, the limiting factor is no longer the polygon but the graphic state. To take advantage of GPU polygonal processing capabilities, the number of graphic states must be reduced at the expense of higher polygonal count. Future GPU development will also increase polygonal rendering rates faster than graphic state switching rates.

One method to handle rendering terrain while optimally feeding the graphics pipeline was described in [4]. The author exploits the ideas set non-polygonal terrain. Terrain generation follows a technique similar to texture MIP-map generation. High-frequency content is removed as coarser terrain resolutions are generated, resulting in smoother transitions between levels of detail.

Flight training and, more specifically, helicopter flight training require high-resolution terrain and imagery. Terrain grid post spacing and imagery texels spanning less than 0.027 arc-seconds (about one foot) will become more and more common. To efficiently use GPUs to render terrain elevation and terrain imagery at those fine resolutions, we have to use methods to generate extremely large grids of terrain skin and extremely large terrain imagery textures in a structure that allows an image generator to seamlessly load portions of the textured skin around the viewpoint. We need a virtual MIP-map of the terrain elevation and a virtual MIP-map of the imagery.

Using the techniques discussed in [4] – creating non-polygonal MIP-map terrain – will reduced rendering artifacts and will allow us to render visual databases at finer resolutions than could previously be done.

**4. Polygonal terrain**

Regular grid could be square, hexagonal, or triangular and so on. For Cartesian coordinate system most natural is square grid with axis collinear cell sides and grid-aligned origin. Let us call this grid regular and all other – irregular. "Most" an irregular is grid with randomly spaced nodes. Others could have other kind of irregularity: square grid rotated, or shifted relatively to coordinate system, or with independently shifted nodes. Some properties of these grids are "regular", and we can use them for irregular grids evaluation. Generally, to achieve higher compression it takes more processing resources (time, memory), including decompression. In our case, compression factor is a number of IG input triangles under irregular grid model with respect to that of regular grid terrain model.

Regular grid has fixed sampling rate for each LOD. In this case, LODi is a set of triangles, which approximates terrain so as the maximum error is not higher than appropriate constant.

Irregular grid has fixed bandwidth, and LOD with the same maximum error could have fewer nodes in this case. Here LODi is represented by set of triangles, organized in clusters so as for each of them, maximum approximation error is between current LOD maximum and next LOD maximum.

Irregular grid bandwidth wideness has two consequences. Each LOD can have "built-in" (implicit) surface roughness [5] and therefore database volume could be reduced.

For regular grid, the maximum amount of memory also could be specified, because of limited mountain's height. If this limit is 300m for 10m space frequency then regular and irregular grids will introduce the same error at regular cell size 8 times less than that of irregular. The later gives us a difference in triangles number about 64 times. This is in accordance with [12].

Rendering photo-realistic, complex terrain features at interactive rates requires innovative techniques. A polygonal model and geometric pipeline can be used but this introduces massive storage requirements and, ideally, a parallel implementation of the algorithm. However, features with high spatial frequency context (ridgelines and canyons) require large numbers of polygons to meet a specified level of terrain accuracy. Using traditional polygonal representation for the example complex surface give rise to a range of problems such as visible surface determination, depth complexity handling, controlling levels of details, clipping polygons by viewing frustum, geometry transformations of large number of polygons [7-15].

Numerous methods for rendering height-based terrain surfaces have been developed [16]. Databases for terrain use DEM (digital elevation model) models. This standard is designed by U.S. Geological Survey and, on essences, is a table of heights terrain with counting out through 7.5 or 15 minutes. DEM model consists of two files, binary file of data in which recorded heights in the manner of 16- bit fixed numbers, and head file which describes a format of record of numbers used in the file of data (BigEndian or SmallEndian ), but in the same way area on terrestrial surface which describe heights in the file of data. The continuous level of detail algorithm takes a two-part approach in which terrain is first divided into blocks for which a detail level can be selected at a coarse granularity [17]. The real-time optimally adapting meshes algorithm builds upon the algorithm [17] by organizing terrain meshes into a triangle bintree structure [18]. Geomorphing to the continuous level of

detail algorithms described in [19]. The progressive mesh technique was extended to height-based terrain, and it enables smooth view-dependent terrain rendering with geomorphs [20].

The terrain relief shown in Fig. 1.

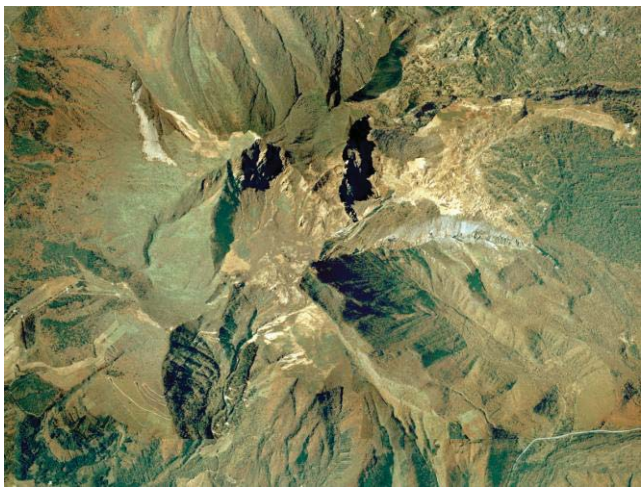


**Fig. 1. – Top view of the landscape**

**5. 3D vegetation visualization**

The method consists of two views: a forest view in the foreground, a forest view in the foreground, and a transition from one view to another.

In the distance, a forest can be represented as a 3D model of the earth, textured with a corresponding texture, but this representation does not have the necessary volume, which is typical for a forest. To add volume, the method proposed in [21] was used to visualize the fur. In this case, the forest is represented by a set of layers that repeat the geometry of the earth, and are shifted relative to it either up or in the direction of the normal for the corresponding distances. In order not to drill down on the geometry of the forest layers at the same time as detailing the geometry of the earth, and in order

to avoid excessive growth of the database volume, the geometry of the layers was generated using Vertex Shader from the earth geometry when displayed. Each of the layers is textured with a special

texture representing a horizontal slice of one or more trees. To add more volume, the textures of the lower layers are darkened. To achieve greater visual diversity, the forest layers are also textured with another texture $T_G$, which uses RGB channels to color the trees, and A channel to "cut out" the glades.

In addition, the same texture, with the same texture coordinates as on the layers, is also used to visualize the earth. The final color of the i-th layer of trees is calculated using the formula

$$C = 2T_C T_G \tag{1}$$

$$C_A = T_A T_{GA} \tag{2}$$

The spatial distribution of trees can be set using the formula

$$D = \frac{1}{2}\sum_{i=0}^{n-1} C_A \tag{3}$$

However, when using this method on the silhouettes of mountains, as well as when the direction of view is parallel to the forest layers, visual artifacts appear.

In order to hide most of these artifacts, it was proposed to use a system of vertical fences. Fences are a set of vertical two-sided faces generated on edges 3D models of terrain and trees with a characteristic height. Fences are textured with the $T_E$ texture, which is a vertical cut of one or more trees. To reduce

visual differences between the layers of the forest and vertical fences on fence superimposed with the same texture coordinates on the earth and on layers of the forest, $T_G$ texture and texture coordinates $T_E$ for the texture's selected in order as accurately as possible to match the texture of the spatial distribution trees. In addition, to avoid visual differences in the lighting of the fences, the normal to the fences used for lighting is either directed vertically upwards or perpendicular the terrain on which the fence is located. The final color of the vertical fences is calculated using the formula

$$C = 2T_E T_G \tag{4}$$

$$C_A = T_E T_{GA} \tag{5}$$

In order to reduce the number of faces forming a system of vertical fences, fences are generated only on those edges of the terrain where the angle between the faces containing this edge is less than the critical angle. The critical angle is chosen empirically depending on the nature of the terrain for which the fence system is generated. Also from the system completely

transparent fences are excluded, i.e. those fences that completely fall into the clearing. When using a vertical fence system, when the orientation of the fence is parallel to the direction of view, visual artifacts associated with trilinear texture filtering appear. In order to get rid of such artifacts, the dependence of the transparency of vertical fences on the angle between the tangent to the fence and the direction to the observer is introduced.

The transparency is calculated at each vertex of the fence using the Vertex Shader and interpolated inside the face. By combining fences with close tangents into groups and preserving the tangent cone for each of the groups, you can perform early rejection of completely transparent groups of fences.

The described representation of large forest spaces, in the form of a combination of a system of layers and vertical fences, is only applicable for displaying distant plans, so it is necessary to smoothly control the transparency of this representation of the forest, depending on distances to the observer. In addition, to reduce the artifacts associated with clipping the geometry of the far clipping plane, the transparency of the forest layers, the vertical fence system, and the ground increases as you approach the far clipping plane.

To represent the forest near the observer, a set of three-dimensional models of trees placed on the terrain is used [22]. The transition to the view in the background is carried out through a smooth change in the transparency of three-dimensional trees. In this case, the transparency changes as follows

$$A = \begin{cases} 0 \; if \;\; d > d_e \\ 1 \; if \;\; d < d_s \end{cases} \tag{6}$$

where d is the distance to the observer, $d_s$ is the distance between start and layers fence, $d_e$ is the distance between the layers and fence fully appeared.

For additional optimization, the system of layers and vertical fences is divided into spatial groups, and if this group is completely transparent, it is not visualized grouping is used to discard faces that are not in the viewport.

Moreover, since the area of disappearance of three-dimensional models of trees, and the area of appearance of forest layers and vertical fences overlap, the number of visual artifacts that appear when the observer moves is minimal.

To reduce the visual artifacts associated with the transition from one view to another

, the tree models are arranged according to the distribution texture, the color of the tree is determined by the $T_G$ texture , and the tree itself is illuminated according to the normal to the terrain at the point at which the tree is placed.

Three-dimensional tree models can be visualized in n calls to the GPU, where n is proportional to the number of trees that need to be displayed at a given time. Since the number of trees to display is large enough, this approach leads to a significant performance decrease. On the other hand, at the preparation stage, you can combine three-dimensional tree models into groups to visualize them in a single call to the graphics accelerator, but this approach leads to a significant increase in the volume of the database.

Therefore, at the preparation stage, the forest is spatially divided into kd-tree, so that the leaf of the tree is either empty, i.e. does not contain any trees, or contains as many trees as possible. Each non-empty leaf contains several trees, defined by their position relative to the borders of the leaf, the normal to the surface at a given point, the color of the tree, and the number of the three-dimensional model in the list of all tree models. Moreover, to save disk space, the position and normal were set with reduced accuracy. At the mapping stage, the kd-tree is recursively traversed, and if a node does not fall within the scope, then it and its descendants are not traversed. All three-dimensional tree models from visible and non-empty nodes kd-tree, collected, sorted in the order " from depth to yourself" and are rendered in a single call to the GPU.

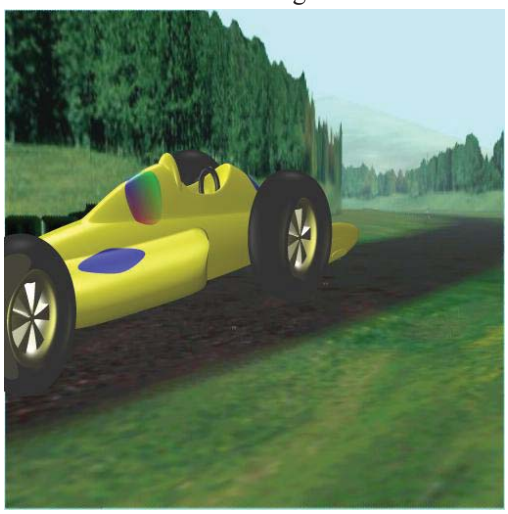


**Fig. 2. – Mountain road with forest**

**Conclusion**

The main merits of proposed approach are the following: reduction of the load on the CPU and decrease of data flow from it to the GPU. The corresponding traversing of the scene and the set of masks provide the right priority order. Rendering

method, described above, uses a GPU for most of the calculations. We can use parallel calculations in GPU to accelerate rendering. We successfully integrated proposed visualization method into the standard rendering pipeline. Verify the performance for the different scenes. For considered tests the application with GPU average ten times faster, than the version using only CPU.

A method of continuous detailing of three-dimensional vegetation models in virtual reality systems has also been developed, which has an acceptable visual quality at any distance from the observer.

The proposed method takes into account the features of the architecture of modern GPUs and allows you to distribute the load on the display between the CPU and GPU. In this case, the method does not require significant resources for visualization.

The implementation of the method showed that the visualization of a wooded surface area is 10000x10000 on an Intel Core 2 CPU E8400 3.0 GHz, and a GeForce 8800 GTX it was happening at a speed of ≈60 frames per second. This speed in most cases is sufficient for the effective application of the method in computer games.

**References**

[1]. Lance Williams, Pyramidal Parametrics, Computer Graphics (SIGGRAPH '83 Proceedings), pp. 1-11, July, 1983. https://doi.org/10.1145/964967.801126

[2]. Vyatkin S. I., Romanyuk A. N. Compression of graphic information using thematic textures. Scientific works of Donetsk national technical University. Series "Informatics, Cybernetics and computer engineering", 2010, Donetsk: DonNTU, vol. 12 (165). P. 82–86.

[3]. https://www.ti.inf.ethz.ch/ew/Lehre/CG13/lecture/Chapter%206.pdf

[4]. Vyatkin Sergey I. Three-dimensional modeling of the Volcano Bandai using shape texture // International Journal of Natural Sciences Research. – 2015. – V.3. - № 2. – pp. 21-29. https://doi.org/10.18488/journal.63/2015.3.2/63.2.21.29

[5]. R. Ferguson, R. Economy, W. Kelly, and P. Ramos. "Continuous Terrain Level of Detail for Visual Simulation", *Proceedings of Image V Conference*, June, 1990. pp. 144-151.

[6]. L.L. Scarlatos "A Refined Triangulation Hierarchy for Multiple Levels of Terrain Detail", *Proceedings of Image V Conference*, June, 1990. pp. 115-122.

[7]. L.L. Scarlatos. Adaptive Terrain Models for Real-Time Simulation, *Proceedings of the Digital Electronic Terrain Board Symposium*, Wichita, KS, October 1989, pp. 219-229.

[8]. L.L. Scarlatos (1989). A Compact Terrain Model Based On Critical Topographic Features, *Proceedings of Auto Carto 9*, Baltimore, MD, April 1989, pp. 146-155.

[9]. L. Scarlatos and T. Pavlidis. Adaptive Hierarchical Triangulation, *Proceedings of Auto-Carto 10*, Baltimore, MD, March 1991, pp. 234-246.

[10]. L. Scarlatos and T. Pavlidis. Hierarchical Triangulation Using Terrain Features, *Proceedings of Visualization '90*, San Francisco, CA, October 1990, pp. 168-175.

[11]. L.L. Scarlatos. An Automated Critical Line Detector for Digital Elevation Matrices, *Technical Papers of 1990 ACSM-ASPRS Annual Convention*, Volume 2, Denver, CO, March 1990, pp. 43-52.

[12]. L. Scarlatos and T. Pavlidis. Adaptive Hierarchical Triangulation, *Proceedings of Auto-Carto 10*, Baltimore, MD, March 1991, pp. 234-246.

[13]. L. Scarlatos and T. Pavlidis. Hierarchical Triangulation Using Cartographic Coherence, *CVGIP: Graphical Models and Image Processing*, 54 (2), March 1992, pp. 147-161. https://doi.org/10.1016/1049-9652(92)90062-3

[14]. L.L. Scarlatos and T. Pavlidis. "Real Time Manipulation of 3D Terrain Models", *1993 ACSM/ASPRS Annual Convention & Exposition Technical Papers*, Volume 3, New Orleans, LA, February 1993, pp. 331-339.

[15]. L.L. Scarlatos and T. Pavlidis. Techniques for Merging Raster and Vector Features with 3D Terrain Models in Real Time, *1993ACSM/ASPRS Annual Convention & Exposition Technical Papers*, Volume 1, New Orleans, LA, February 1993, pp. 372-381.

[16]. R. Pajarola, and E. Gobbetti. Survey on Semi-Regular Multiresolution Models for Interactive Terrain Rendering. *The Visual Computer: International Journal of Computer Graphics*, Volume 23, Number 8, 2007, pp. 583–605. https://doi.org/10.1007/s00371-007-0163-2

[17]. P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, F. Nick, and G. A. Turner. Real-Time, Continuous Level of Detail Rendering of Height Fields. *Proceedings of SIGGRAPH 1996*, pp. 109–118. https://doi.org/10.1145/237170.237217

[18]. M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Alrich, and M. B. Mineev-Weinstein. ROAMing Terrain: Real-time Optimally Adapting Meshes. *Proceedings of the 8th Conference on Visualization '97*, pp. 81–88.

[19]. S. Rottger, W. Heidrich, P. Slusallek, and H-P. Seidel. Real-Time Generation of Continuous Levels of Detail for Height Fields. *Proceedings of WSCG '98*, pp. 315–322.

[20]. H. Hoppe. Smooth View-Dependent Level-of-Detail Control and its Application to Terrain Rendering. *Proceedings of the Conference on Visualization '98*, pp. 35–42.

[21]. J. Lengyel, P. Emil, F. Adam, H. Huges. Real-Time Fur over Arbitrary Surfaces. ACM Symposium on Interactive 3D Graphics 2001, pp. 227-232. https://doi.org/10.1145/364338.364407

[22]. K. Pelzer, P. Bytes. Rendering Countless Blades of Waving Grass. GPU Gems, Addison Wesley, 2004, pp.

Отримана в редакції 19.11.2021. Прийнята до друку 30.11.2021. Received 19 November 2021. Approved Approved 30 November 2021. Available in Internet 04 December 2021.