Transdisciplinary Engineering for Complex Socio-technical Systems
K. Hiekata et al. (Eds.)

doi:10.3233/ATDE190175

Visualization of Complex Industrial Products and Processes Using Hierarchical Dynamic LODs

Vitaly SEMENOV[a,b,c,1], Vasily SHUTKIN[a] and Vladislav ZOLOTOV[a]
[a] *Ivannikov Institute for System Programming of the Russian Academy of Sciences*
[b] *Moscow Institute of Physics and Technology (State University)*
[c] *Higher Scool of Economics (State University)*

Abstract. Visualization of complex industrial products and processes is a challenging problem connected with emerging collaborative environments and integrated product development technologies. Due to the large number of components and the complexity of their geometrical models, direct display of products and processes by current graphics systems at reasonable frame rates becomes impossible. One of the promising approaches to visualization of complex scenes is a simplification of polygonal models often hidden under the term *level of details* (LOD). Nowadays, different techniques, including hierarchical level of details (HLOD), are widely employed for visualization of large scenes by means of the geometry simplification and representation of entire groups of objects. However, both LOD and HLOD techniques face dramatic difficulties when scenes undergo modifications. Known attempts to adapt techniques for dynamic scenes did not result in significant progress. In the paper we present a concept of hierarchical dynamic level of details (HDLOD) and prove its feasibility and applicability to deterministic pseudo-dynamic scenes. Such scenes arise in various problems connected with visual modeling of complex construction projects, city infrastructure programs, and machinery assembly production. The results of computational experiments confirm the prospects of the introduced HDLODs and proposed complementary method for their automatic generation and application in industrial practice.

Keywords. Product and Process Modeling, Visualization, Level of Details

Introduction

Visualization of complex industrial products and processes of their manufacturing, assembly and mounting is a challenging problem connected with emerging collaborative environments and integrated product development technologies [1, 2]. Due to the large number of components and the complexity of their geometrical models, direct display of products and processes by current graphics systems at reasonable frame rates is often impossible.

One of the promising approaches to visualization of complex scenes is a simplification of polygonal models that has long been a topic of research in computer graphics. The first milestone was set by James Clark 1976 when he introduced the term

[1] Corresponding Author, Email: sem@ispras.ru.

level of details (LOD) [3]. During visualization, the appropriate LODs are selected and rendered so that finer representations are used for the nearest objects and coarser approximations are applied to the farther objects.

Numerous polygonal simplification algorithms have been proposed following the same goal: to reduce the number of polygons of the mesh while preserving details and characteristics of the original as much as possible. The algorithms can be categorized by two main aspects: the underlying decimation operation used to transform the mesh and the error metric used to measure the geometric error introduced by the operations. The underlying operations used in computing LODs include vertex removal, edge collapse, face collapse, vertex clustering and vertex merging. Due to restrictions we do not review all of these algorithms here. Instead, we refer to existing surveys on the topic [4, 5].

The aforementioned algorithms can help in the generation of simplified representations for single polygonal objects, but are quite limited to manage the complexity of the scenes containing hundreds of millions of polygonal objects and billions of primitives. Therefore, more general and effective computational strategies are required to visualize large scenes, for example, the urban scenes hierarchically organized as compositions of detailed landscape elevations, extended district maps, and advanced buildings models.

Nowadays, hierarchical level of details (HLOD) techniques are widely employed for visualization of large scenes [6]. Unlike traditional ones, HLODs represent the geometry not of the individual objects, but of the entire groups of objects. Instead of analyzing and rendering an appropriate level of detail for every object, visualization systems become capable to render the appropriate approximations of the whole branches when traversing HLOD trees. Visiting and processing all descending nodes and individual objects becomes generally unnecessary.

However, both LOD and HLOD techniques face dramatic difficulties when scenes undergo modifications. Possible assumptions about the rigidness of objects and their infrequent motions do not change the situation. HLOD representations have to be recomputed whenever events of adding, deleting or moving the objects occur in the scene. The time required for recomputing is typically greater than that for rendering the scene, which makes it impossible to visualize and animate large dynamic scenes at interactive frame rates. Known attempts to optimize HLODs by using incremental updates and their asynchronous parallel execution on shared memory multiprocessor systems, haven't resulted in significant success [6].

In the paper we present a concept of hierarchical dynamic level of details (HDLOD) and prove its feasibility and applicability to deterministic pseudo-dynamic scenes. Such scenes have received an increasing interest in recent years due to the rapid acceptance of Integrated Product Development (IPD), Building Information Modeling (BIM), and nD Modeling technologies [1, 7]. Particularly, such scenes arise in applications connected with visualization of complex construction projects, city infrastructure programs, and machinery assembly production as well as with spatio-temporal modeling of production processes and validation of project schedules. Due to a simplified model caused by deterministic a priori known events and the lack of continuous movements, large dynamic scenes can be visualized and animated more effectively.

In Section 2 we specify pseudo-dynamic scenes and provide mathematical definitions of the key metrics to measure the object proximity. The HDLOD concept is introduced in Section 3 with explanatory descriptions of the underlying data structures and spatio-temporal tolerances required for their meaningful use. A method of automatic generation of HDLODs is presented in Section 4. The consideration is supplemented by

a list of general principles and requirements that should be taken into account when building HDLODs. The results of computational experiments are presented in Section 5. The benefits of the HDLODs and their prospects for wide industrial use are shortly summarized in Conclusions.

1. Deterministic pseudo-dynamic scenes

Let the scene $S(t)$ be defined in three-dimensional Euclidian space E^3 on the model time period $t \in [0, T]$ and be composed of objects $s(g_s, f_s) \in S$ having fixed spatial positions and unchangeable geometry representations $g_s \subseteq E^3$. The presence of objects in the scene is determined by the behavior functions $f_s(t)$: $[0, T] \to \{0,1\}$ so that the function $f_s(t)$ takes the unit value if the object s_i is present in the scene at the time t and the zero value the object is absent. Throughout the work we call such scenes deterministic pseudo-dynamic. It is worth to mention that they can also be used for modeling of continuously moving objects. This can be achieved through the discretization of a continuous path and simulation of successive events of the appearance and removal of object instances along this path. Since the object instantiations may result in redundant representations, these possibilities are omitted from further consideration. We suggest that relatively few objects are continuously moved or even such movements are completely absent in the scene.

Contrary to this limitation, the considered class of scenes has numerous industrial applications. Figure 1 shows an example of a scene simulating the construction of a skyscraper in accordance with a preliminary prepared project schedule. As model time changes, building elements and pieces of equipment are mounted, installed on a construction site or removed. The landscape and part of the elements belonging to the surrounding buildings remain unchanged throughout the entire modeling period. Such behaviors can be reproduced by standard functions presented in Figure 2.

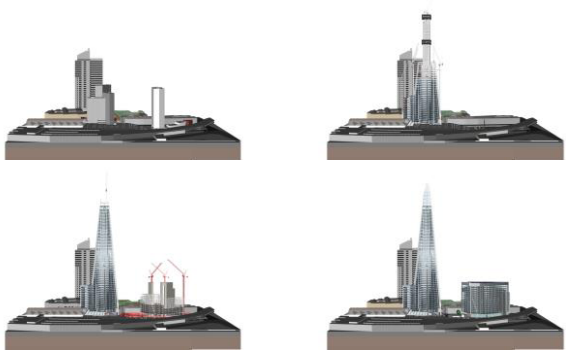

Figure 1. Example of a pseudo-dynamic scene simulating the construction of a skyscraper.

In the following discussion we assume that the scene objects are geometrically represented as triangle sets with vertices referring to common set of points. No assumptions about solidness of the objects and manifoldness of their boundary representations are made because CAD/CAE/CAM systems often provide so-called polygon soups only suitable for visualization purposes.

We also suggest that each object $s \in S$ is specified by a relative dimensional parameter $0 < w_s \leq 1$. The parameter may express the overall object size, surface area

or volume in relation to the entire scene. The exact semantics is not important, since the parameters are used as weights when evaluating the visual strength of individual objects.

The introduced HDLOD concept implies a process of simplifying the scene objects. To measure the proximity or similarity of their alternative representations we need distance functions or metrics applicable to both the geometry of objects and their behavioral functions.

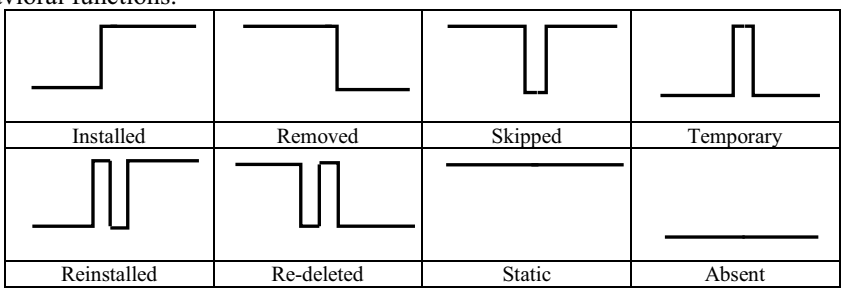

Figure 2. Standard behavior functions.

From a formal point of view, deviations in the geometry representations of an original object and its simplified version can be naturally expressed by the Hausdorff distance. Let A, B be non-empty sets of points in Euclidian space. Then their Hausdorff distance in Euclidian space with the metric function $D(x, y)$ is defined as follows:

$$D_H(A, B) = max\{max_{x \in A}\, min_{y \in B}\, D(x, y), max_{y \in B}\, min_{x \in A}\, D(x, y)\}$$

Informally, the Hausdorff distance is the greatest of all the distances from a point in one set to the closest point in the other set. Sometimes, its usage is burdensome and a non-symmetric distance function can be employed to evaluate the geometry deviations:

$$D(A, B) = max_{x \in A}\, min_{y \in B}\, D(x, y)$$

Let us make a few remarks on the behavior functions. They can be defined and processed as sets of segments $\tau = \{\tau^k\} \subseteq [0, T]$, in which the objects are present in the scene. The ends of the segments τ^k correspond to the times of events in the scene in which the objects change the presence status and thereby change the scene representation. Thereby, the function $f(t)$ can be presented using the segment representation as follows

$$f(t) = \begin{cases} 1, & t \in \{\tau^k\} \\ 0, & otherwise \end{cases}$$

To measure the proximity of behaviors $f_A(t), f_B(t): [0, T] \to \{0,1\}$, one can utilize the segment representations and apply the Hausdorff distance or non-symmetric distance function defined above. However, the following functional metric looks more straightforward in this case:

$$D_F(f_A(t), f_B(t)) = \frac{1}{T}\int_0^T |f_A(t) - f_B(t)|dt$$

Since the distance functions are used only to estimate the proximity or similarity of objects of the scene, the choice of a specific function and the method of its computation is not essential for further discussion.

2. Hierarchial dynamical LODs

Further in the work the scenes are considered to be the flat compositions of objects. The availability of the scene hierarchy is not taken into account, because CAD systems commonly group objects by functional characteristic rather than by spatio-temporal relationships that could be established between objects. Direct use of the scene hierarchies could impede rational optimization of LODs for rendering purposes and therefore sometimes it is better to completely ignore them or consider as a minor factor.

We use hierarchical clustering to create a more coherent representation of the scene. Such a representation should be efficient for both view-dependent rendering and view-frustum culling purposes. It is assumed that HDLODs have a uniform organization that allows you to compose high-level structures from the structures already prepared for separate components or fragments of the scenes. Thereby, there are no principal distinctions between internal and external organization of HDLODs.

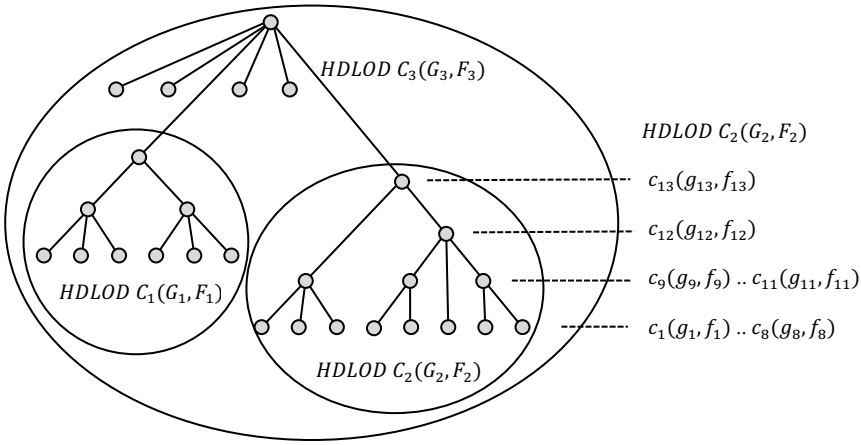

Figure 3. Example of composition and unbalanced HDLOD trees.

We will call the HDLOD a cluster tree $C(G, F) = \{\{c(g_c, f_c)\}, \prec\}$ represented by a set of clusters $c(g_c, f_c)$ with prescribed geometry representations g_c and behavior functions f_c. For the clusters an agglomerative relation $\prec$ is defined so that $c' \prec c$ only if the cluster $c' \in C$ is a direct child of the parent cluster $c \in C$. The leaves of the tree are the precisely defined individual objects. Interior nodes represent clusters aggregating the geometries and behaviors of the objects in corresponding subtrees. The root is the cluster containing the least accurate and most simplified representation of the entire scene. No preliminary assumptions are made about the degree of nodes and the balance of the tree. Figure 3 provides an example of the HDLODs composition and unbalanced HDLOD trees with the nodes of varied degree.

Each cluster $c \in C$ can store also such derived attributes as axis-aligned or oriented bounding box bb_c, weight or strength w_c, spatial tolerance ε_c, temporal tolerance γ_c as well as various complexity characteristics.

When visualizing a scene, the cluster tree is traversed starting from the root and descending to the leaves. At each node the cluster attributes are analyzed to determine the necessity of the further traversal of the subtree. It is checked whether the cluster is present in the scene at a specified model time, whether the cluster bounding box falls into the view frustum, and whether the spatial and temporal tolerances are small enough to obtain the required resolution. If all the conditions are satisfied, the cluster representation is applied for rendering immediately. In such case, a whole subtree can be pruned from the traversal, which decreases the number of visualized objects and accelerates the rendering process. If any of the conditions is not met, the traversal of the cluster subtree continues and descending nodes are underwent to similar checks until the leaves with precisely given scene objects are reached. Due to hierarchical cluster organization and undertaken simplifications of the geometry and behavior, significant performance gain can be reached for rendering both static and dynamic scenes.

Advantages of HDLODs over alternative techniques are especially prominent when you need to animate large dynamic scenes and to produce series of snapshot images or videos. Because the presence, frustum, and tolerance checks are naturally combined when traversing the cluster tree, additional overheads for the analysis of individual objects are avoided. Another benefit is achieved when scene should be redrawn multiple times during the animation. HDLOD trees are better adapted to dynamic scenes and allow us to easily control changed object clusters and to determine regions in target frames which could be locally redrawn instead of rendering the entire scene.

In order to make a decision about the applicability of clusters, two aforementioned tolerance parameters ε_c, γ_c are used. These parameters allow different interpretations. For definiteness, we assume that ε_c is the absolute tolerance which establishes the possible maximum local deviation of the geometric representation of the cluster c from an aggregated representation of the originating objects $o \prec .. \prec c' \prec c$. In particular, this means that any triangular face with edges smaller than a given tolerance ε_c can be collapsed without any violation of the accuracy requirements. The spatial tolerance can be recurrently defined using the Hausdorff distance:

$$\varepsilon_c = \max_{c' \prec c}(\varepsilon_{c'}) + D_H\left(g_c\,,\ \bigcup_{c' \prec c} g_{c'}\right)$$

We also admit the use of non-symmetric distance function for such purpose:

$$\varepsilon_c = \max_{c' \prec c}(\varepsilon_{c'} + D(g_{c'}, g_c))$$

Regardless of how the spatial tolerance is defined, it should be evaluated during geometry simplication using one of the methods mentioned in the introduction.

The temporal tolerance γ_c is defined as a maximum deviation of the cluster behavior from individual behaviors of the originating objects:

$$\gamma_c = \max_{c' \prec c}\left(\gamma_{c'} + D_F(f_c, f_{c'})\right),$$

where the distance is determined using the aforementioned functional metric. Unlike the simplification of polygonal models, the topic of simplifying behavior has not been studied at all. One possible method is to compute a mean behavior function as follows:

$$f_c(t) = round_{\{0,1\}}\left(\frac{\sum_{c' \prec c} w_{c'} f_{c'}(t)}{\sum_{c' \prec c} w_{c'}}\right)$$

Here the weighted sum of the behavior functions of the originating clusters is rounded to zero or one so that the resulting function has the same value domain $\{0,1\}$. The use of cluster weights enables us to take into account the behavior of significant objects to a greater degree and to increase the trustworness of visual spatial-temporal modeling.

3. HDLODs generation method

In accordance with the previous sections, hierarchical clustering can be employed to generate HDLODs applicable for both view-dependent rendering and view-frustum culling purposes. To be effective the HDLODs should meet a number of requirements and principles:

1. The number of levels of details should be large enough so that the transitions from one level of resolution to others are carried out smoothly;
2. The number of children clusters should be reasonably limited to eliminate the need to iterate and analyze long lists of nodes when traversing the tree. To some extent this requirement is explained by the need to load clusters dynamically using the page organization of memory;
3. Clusters should be grouped so that their density in the bounding volume of their parent is maximal, and the overlap of the bounding volumes of sibling clusters is minimal. This requirement enables to decrease the number of the traversed clusters and to rise up the efficiency of the frustum culling procedure;
4. Clusters should be grouped so that their behavior is slightly different from that of the parent cluster. It enables to identify the clusters present or absent in the scene at a given modeling time quickly avoiding traverse of descending nodes;
5. Grouped clusters should have approximately the same complexity and accuracy in order to ensure a uniformly high efficiency of displaying the scene from different camera views in different animation modes.
6. Clustering should be carried out under a general strategy assuming control of both the complexity and accuracy. The goal of any meaningful strategy is to simplify the geometry and behavior to the required degree with minimal loss of accuracy. Particularly, it can be reached iteratively while weakening the accuracy requirements and controlling the achieved simplifications.

Unfortunately, classical clustering methods cannot be directly adapted to HDLOD generation problems. First of all, the imposed requirements are quite complex and can contradict to each other. That hinders the mathematical formalization of metric functions and linkage criteria necessary for the clustering methods [8]. The second reason is the necessity to combine clustering methods with simplification procedures which may have ambiguous results due to the conflicting requirements to the cluster simplicity and accuracy. The third significant reason is unacceptably high computational complexity of the clustering methods. For example, a naïve implementation of agglomerative clustering has a time complexity of $O(n^3)$ and requires $O(n^3)$ memory, which makes it useless for

even simple scenes. Faster hierarchical clustering with a time complexity of $O(n^2)$ and memory consumption $O(n^2)$ is also not feasible for the problems discussed [8].

In this paper we propose an efficient computational method for HDLOD generation. The method leverages both agglomerative (bottom-up) clustering and multi-level accuracy control. For brevity we present only general features of the method and omit specific algorithms, heuristics and implementation details.

```
PROCEDURE GENERATE HDLOD (SCENE scene, INTEGER levels, HDLOD tree)
{
SET OF CLUSTER active = NULL, next = NULL, neighbors = NULL
FOR_EACH (OBJECT representative IN OBJECTS ( scene ) )
    {
    CLUSTER cluster
    GROUP (representative, neighbors, cluster)
    ADD TO (cluster, tree)
    ADD TO (cluster, active)
    }
FOR_EACH (LEVEL level=2 TO levels)
    {
    REAL spatial_tol, temporal_tol
    COMPUTE TOLERANCES (scene, level, spatial_tol, temporal_tol)
    WHILE (NOT IS EMPTY (active))
        {
        SELECT REPRESENTATIVE (active, representative)
        FIND NEIGHBORS (representative, active, spatial_tol, temporal_tol, neighbors)
        IF (IS EMPTY (neighbors))
                {
                ADD TO (representative, next)
                REMOVE FROM (representative, active)
                }
        ELSE
                {
                GROUP (representative, neighbors, cluster)
                SIMPLIFY (cluster, spatial_tol, temporal_tol)
                ADD TO (cluster, tree)
                ADD TO (cluster, next)
                REMOVE FROM (representative, active)
                FOR_EACH (CLUSTER neighbor IN neighbors)
                        REMOVE FROM (neighbor, active)
                }
        }
    COPY (next, active)
    EMPTY (next)
    }
}
```

Figure 4. Pseudocode of the HDLOD generation method.

The method starts with the individual objects and progressively groups these together into larger and larger clusters until the root cluster covering the entire scene is formed. The clustering process is divided into steps, at each of which the formed clusters satisfy certain requirements for accuracy. As new steps are taken, the requirements are weakened in such a way that the process is guaranteed to terminate after a specified number of steps equal to the number of levels of details L. Only active clusters are involved in the clustering process as opposed to passive clusters that have already been analyzed and grouped. Pseudocode of one of the algorithms that implement the method is given in Figure 4.

At each planned step l $(1 < l \leq L)$ of the method, an attempt to form new clusters with the preliminary computed tolerance thresholds $\varepsilon(l)$, $\gamma(l)$ is made. Classical clustering algorithms are not suitable for the reasons mentioned above. Therefore, we prefer to form clusters based on the selection of representatives and the search for their neighbors in both spatial and temporal dimensions. The representatives are selected using space-filling curves by Hilbert and Morton [9]. On the one hand, it allows representatives to be chosen throughout the entire volume of the scene, and on the other hand, to localize densely filled regions. Once a representative has been selected, its direct neighbors are searched to form a new cluster. The neighbors should satisfy the conditions of spatial and temporal proximity as well as ensure the formation of the parent cluster with the assigned accuracy. If no proper neighbors are found for the selected representative, it is excluded from the analysis at the current step, but participates in the succeeding cluster formation. The parent cluster behavior is determined using computations of the mean function as specified above. The geometry representation is merged from polygonal models of the originating clusters and then is simplified using well-known edge and face collapse methods [4,5]. Using spatial and temporal indices preliminary computed and deployed, the HDLOD tree can be generated in $O(n \log n)$, where n — the number of the scene objects. Note that we do not solve classical clustering problems at each step of the method. Instead, we form proper clusters at certain levels of the tree when it becomes possible for the accuracy reasons. This explains decreased computation complexity of the proposed method compared with the classical clustering methods.

4. Computational experiments

To validate the introduced HDLODs concept and the proposed method for their automatic generation and visualization, a series of computational experiments has been carried out. Display frame times were measured when navigating across a given scene at fixed modeling times and when animating and viewing the scene from fixed camera positions throughout the modeling period. As a benchmark, the presented in Figure 1 dynamic scene of the skyscriper construction was explored. The skyscraper model consisted of 79396 building elements represented by polygonal meshes totally containing 3632126 triangles. The skyscraper construction schedule consisted of 67099 activities each of which was responsible for certain works on the construction site and induced related events in the dynamic scene.

Table 1. Frame times (in milliseconds) during the visualization of the skyscraper construction scene.

	1/1 screen view	**1/4 screen view**	**1/16 screen view**	**No HDLODs**
Start time	7.02	1.5	0.66	40.88
1/3 of the period	8.7	2.75	0.71	82.9
2/3 of the period	26.47	2.93	0.74	121.3
End time	28.17	3.15	0.76	164.83
Animation	18.63	2.87	0.72	105.4

To evaluate the effectiveness of the HDLODs with eliminated frustum culling factor, the experiments were repeated for a camera located at different distances from the scene. In the first position, the camera was placed at the closest distance, at which the entire scene was visible. In other positions, the camera was placed at the distances, at which the scene was occupied one-half, one-fourth and one-sixteenth of the screen area correspondingly. The times were chosen at the start and end of the modeling period as well as on one third and two thirds of the modeling period. The computational

experiments have been carried out on a typical hardware configuration: Intel Core i7-4790 CPU (3.6 GHz), 16 GB of RAM, GeForce GTX 750 Ti (2 GB).

Table 1 showcases the rendering performance results obtained during the specified experiments. The last column provides the results obtained using direct rendering methods without HDLODs. As seen, the use of HDLODs significantly increases the rendering performance compared to traditional techniques. The farther from the scene the camera is placed, the greater is the achieved effect. It is exhibited both when navigating the static scene fixed at the chosen times, and when animating the scene. It is worth to mention that the dependence is super-linear and this gives reason to make conclusions about the scalability of the method with respect to the complexity of both static and dynamic scenes.

5. Conclusions

Thus, the concept of hierarchial dynamic LODs and the complementary method for their automatic generation and visualization have been presented in the paper. As opposed to traditional LODs techniques suitable only for static scenes, the developed method is feasible and applicable to a wide class of deterministic pseudo-dynamic scenes arising in a variety of industrial applications, including visual modeling of complex construction projects, city infrastructure programs, and machinery assembly production. The conducted computational experiments have confirmed significant performance gain and scalability of the method which makes it applicable for the visualization and animation of very large scenes. Future research will focus on different algorithmic options for the developed method, as well as on its generalizations necessary for applying to scenes with continously moving objects.

References

[1] J. Stjepandić, N. Wognum, W.J.C. Verhagen (eds.), *Concurrent Engineering in the 21st Century: Foundations, Developments and Challenges*, Springer, London, 2015.

[2] J. Osborn, *TigerPrints/Survey of concurrent engineering environments and the application of best practices towards the development of a multiple industry, multiple domain environment*, Clemson University, 2009, Accessed: 28.03.2019. http://tigerprints.clemson.edu/all_theses/635/

[3] J.H. Clark, Hierarchical Geometric Models for Visible Surface Algorithms, *Communications of the ACM*, vol. 19, 1976, pp. 547–554.

[4] D. Luebke, M. Reddy, J.D. Cohen, A. Varshney, B. Watson, R. Huebner, *Level of Detail for 3D Graphics*, Morgan Kaufmann, New York, 2012.

[5] D. Luebke, A Developer's Survey of Polygonal Simplification Algorithms, *IEEE Computer Graphics and Applications*, vol. 21, 2001, pp. 24–35.

[6] C. Erikson, D. Manocha, W.V. Baxter III, HLODs for Faster Display of Large Static and Dynamic Environments, In: *I3D '01 Proceedings of the 2001 symposium on Interactive 3D graphics*, New York, 2001, pp. 111–120.

[7] V. Semenov, A. Anichkin, S. Morozov, O. Tarlapan, V. Zolotov, Visual Planning and Scheduling of Industrial Projects with Spatial Factors, In C. Bil, J. Mo, J. Stjepandic (eds.): *Proceedings of 20th ISPE International Conference on Concurrent Engineering*, IOS Press, Amsterdam, 2013, pp. 343-352.

[8] D. Xu, Y. Tian, A Comprehensive Survey of Clustering Algorithms, *Annals of Data Science*, vol. 2, 2015, pp. 165–193.

[9] H. Samet, *Foundations of Multidimensional and Metric Data Structures*, Morgan Kaufmann, New York, 2006.