# Optimization of the process of 3D visualization of the model of urban environment objects generated on the basis of the attributive information from a digital map

**M P Osipov**[1], **O A Chekodaev**[1]

e-mail: osipovmp@mail.ru

[1]National Research Lobachevsky State University of Nizhny Novgorod, Prospekt Gagarina, 23, Nizhny Novgorod, Russia, 603950

**Abstract.** The paper presents methods for optimizing the process of visualization of the urban environment model based on the characteristics of its presentation. Various approaches are described which provide a reduction in computational complexity in visualizing three-dimensional models that can optimize the display of their geometry and the amount of video memory used. Methods are considered that allow optimizing both the scene as a whole and its individual components.

## 1. Introduction

The visualization of the urban environment model is widely used to solve various problems in such areas as construction, urban management, education, culture, etc. [1-3]. Three-dimensional visualization allows users to perceive information in a spatial form that is familiar to them. Its analysis does not require any special technical knowledge, so decision-making on the basis of the information thus provided is much faster and easier. The most complete information about the object, which allows the immersion into the model space, is provided by real-time visualization that imposes significant time constraints on the process of scene rendering.

## 2. Problem statement

The process of urban environment model visualization is a computationally complex task. In [4], a method for automatic generation of three-dimensional models of objects in the urban environment using attributive information from a digital map is described. To render the surface of extended areas of terrain with highly detailed objects, an enormous number of polygons is required. Therefore, the need arises to use special algorithms that optimize the 3D representation of the urban environment model.

The aim of this paper is to analyze the existing methods for optimizing the process of 3D visualization, to choose some more efficient approaches and to adapt them to the specifics of the rendering of a three-dimensional urban environment model.

## 3. Methods for optimizing the process of 3D visualization of the urban environment model

The urban environment model (Figure 1) includes a digital terrain model and models of urban environment objects.

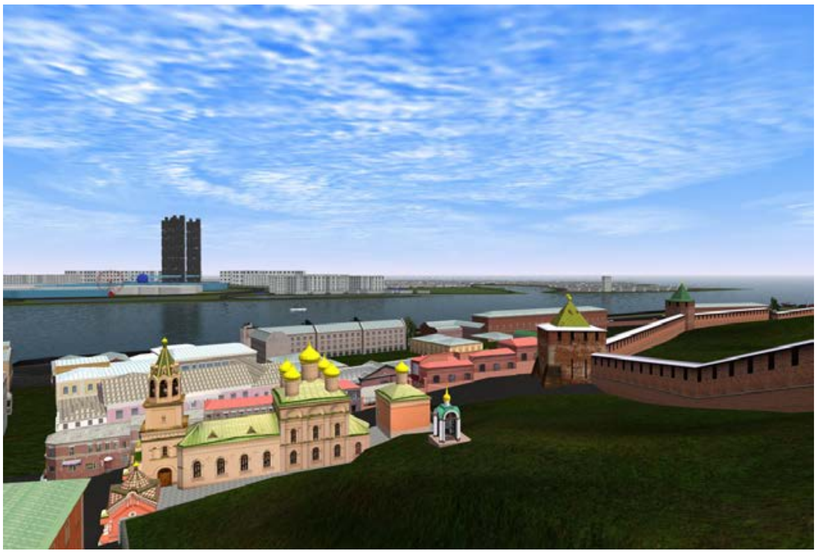

**Figure 1.** 3D visualization of the urban environment model.

### *3.1 Visualization of the digital terrain model*

### *3.1.1 Geometry optimization*

The process of displaying a terrain model, which is a highly detailed description of an extended terrain surface, is a computationally complex element in visualizing an urban environment model. Displaying on the screen the entire terrain model at once will result in a noticeable loss of performance. At the same time, when forming a frame, most of the model will be out of sight. Therefore, it is necessary to develop algorithms that will cut off the data that in any case will not be displayed on the screen, thereby optimizing the visualization process. The surface in the background, due to its remoteness, does not require high detail. Therefore, the visualization algorithm should dynamically change the level of the model detail, depending on the distance from the observer.

The digital model of the urban environment is built on the basis of the elevation information (horizontals, elevation marks, etc.) contained in the digital map. Since the elevation information is an irregular point cloud, the surface is approximated by means of irregular triangulation (TIN).

The process of visualization of an irregular network of triangles is difficult to optimize, because, unlike in the case of the regular representation, it is difficult to parallelize this process on multiprocessor architectures. Therefore, conversion from an irregular to a regular structure is envisaged.

Optimizing the process of a terrain model visualization often consists in producing a tree structure from surface areas that have varying degrees of detail. Each link in the tree approximates a certain part of the surface. The descendants of this link break the area into parts and approximate them with greater accuracy. The process continues up to a given limit of approximation accuracy. All hierarchical algorithms are similar in terms of their principle of operation and differ mainly by the types of entities chosen as links. In particular, the ROAM algorithm [5] implies a binary tree of triangles, the QuadTree [6], a quad-tree of squares, the Chunked LOD [7], a quad-tree of rectangular patches, which are separate triangular grids, etc. [5]. When visualizing, such structures can be used to quite effectively cut off the areas invisible to the observer and to remove unnecessary details in the distant parts of the terrain model. The main disadvantage of such hierarchies is the presence of recursiveness and complexity of their transfer to the GPU. Besides, considerable time is required to form the structure of the hierarchical algorithms.

The Displacement Mapping technique [8] allows transferring the model optimization problem to a graphics processor. Unlike hierarchical methods, Displacement Mapping, when using hardware tessellation, performs all calculations on the GPU, with a minimal load on the central processor. The time required to prepare data is also significantly reduced. Therefore, the process of visualization of extended urban terrain models was optimized using Displacement Mapping methods with hardware tessellation (Figure 2).

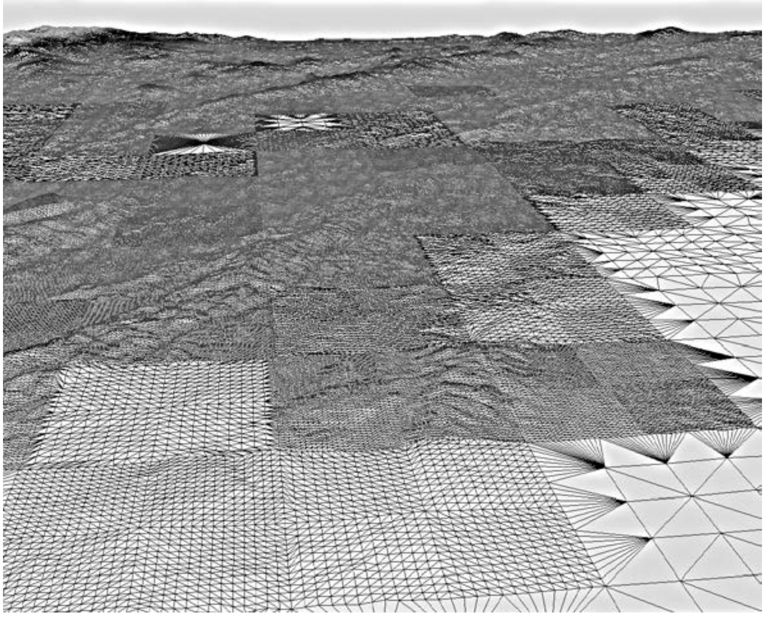

**Figure 2.** Visualization of a digital terrain model using Displacement Mapping methods.

*3.1.2 Material optimization*

The underlying surface characterizes the features of the upper part of the earth crust. In the urban environment, objects that denote paved areas, green areas, squares, courtyards, etc., can act as the underlying surface. These objects completely repeat the geometric features of the terrain, so it is reasonable to represent them as part of the digital terrain model rather than as a separate object and to designate them using an appropriate texture.

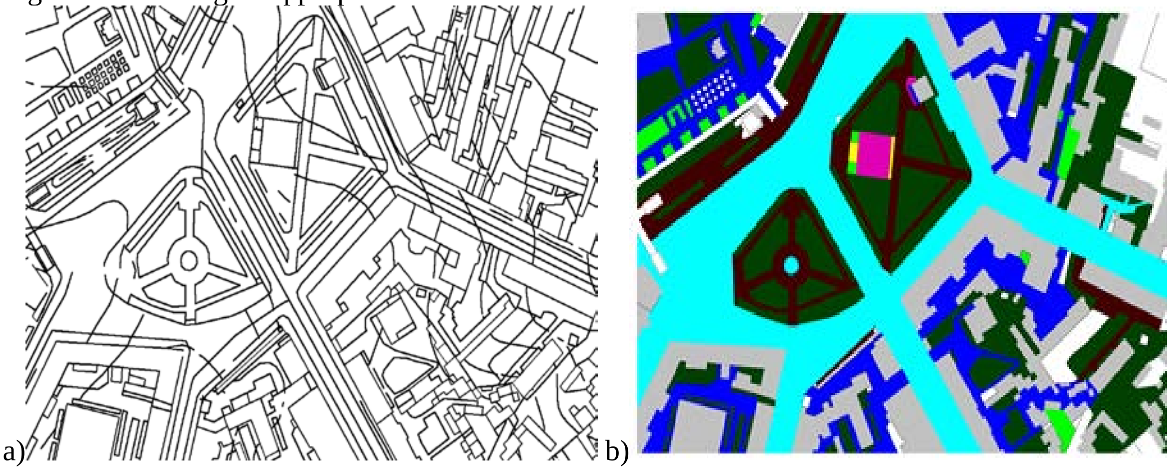

**Figure 3.** A fragment of a digital map (a) and a mask (b) compiled from its data.

To render highly detailed information about the underlying surface of an extended terrain, a super-large texture are required. The information about the elements of the underlying surface is stored in the digital map as a set of characteristics describing these elements. All elements of the underlying surface are categorized into types and their color at each point of the surface can be formed at the moment of visualization. For this purpose, an algorithm was developed that forms a mask in advance, based on the information of the digital map. The mask is in the form of a graphic file with dimensions multiple to the size of the digital map (Figure 3).

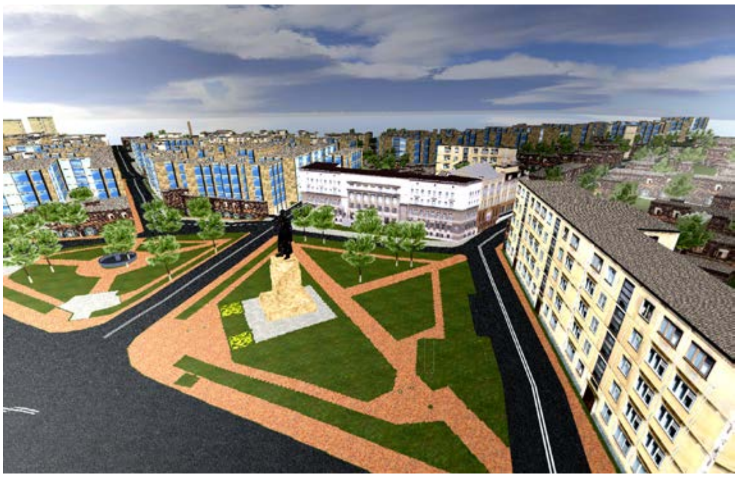

**Figure 4.** A 3D scene of the urban environment with the underlying surface formed using the mask from Figure 3.

Each pixel of the mask covers a specific area of the terrain and contains information about the type of coating of the underlying surface in the corresponding area. The information about the texture of a certain type of the underlying surface is stored in the texture atlas as a set of tiles. Thus, the color at each point of the underlying surface can be calculated at the stage of visualization in the fragment shader based on the information of the mask map and texture atlas (Figure 4). Based on this principle, an algorithm was developed for texturing the underlying surface, which makes it possible to avoid producing the texture of the entire underlying surface in advance and to achieve significant memory saving.

### *3.2 Visualization of urban environment objects*

#### *3.2.1 Geometry optimization*

When rendering a model of the urban environment, the visualization of buildings, vegetation, small architectural forms, etc., accounts for the greatest computational load. Optimizing the process of visualization of such objects consists in cutting off the data that are outside the field of vision (Figure 5). By means of tree-like segmentation of the scene, it is possible to cut off such data more efficiently. As part of our research, segmentation algorithms based on octrees, R-trees and Priority R-tree (a hybrid between a k-dimensional tree and an R-tree) have been implemented [9]. It was established experimentally that segmentation of a scene based on an Priority R-tree is most suitable for urban objects. Based on the nature of the location of objects in the urban environment, optimal segmentation parameters were selected.

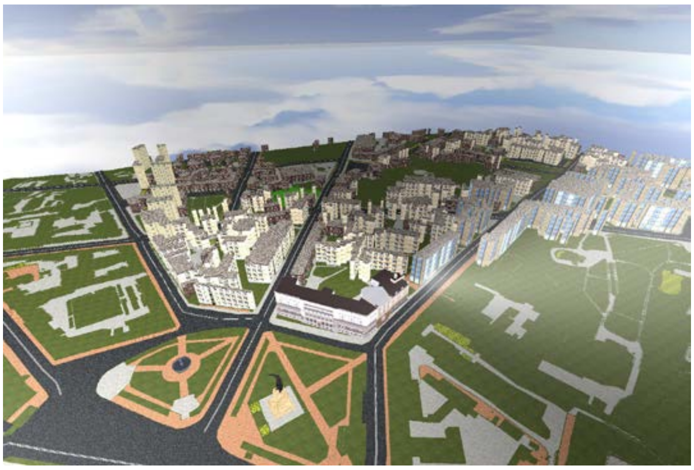

**Figure 5.** Cutting off 3D scene objects out of the field of vision by using scene segmentation based on an R-tree.

When rendering the urban environment, a significant part of the objects will fall within the field of the camera vision (viewing frustum), but will not be seen during visualization, since it will be completely obscured by other objects. The technology based on coherent hierarchical culling [10], by analyzing the previous frame, efficiently cuts off the "obscured" objects that will not be processed in the course of visualization. In combination with scene trees, this provides a very effective way of clipping - the scene is traversed hierarchically. Internal nodes contain objects and other nodes, which makes it possible to check for visibility and cut off a large number of objects at once.

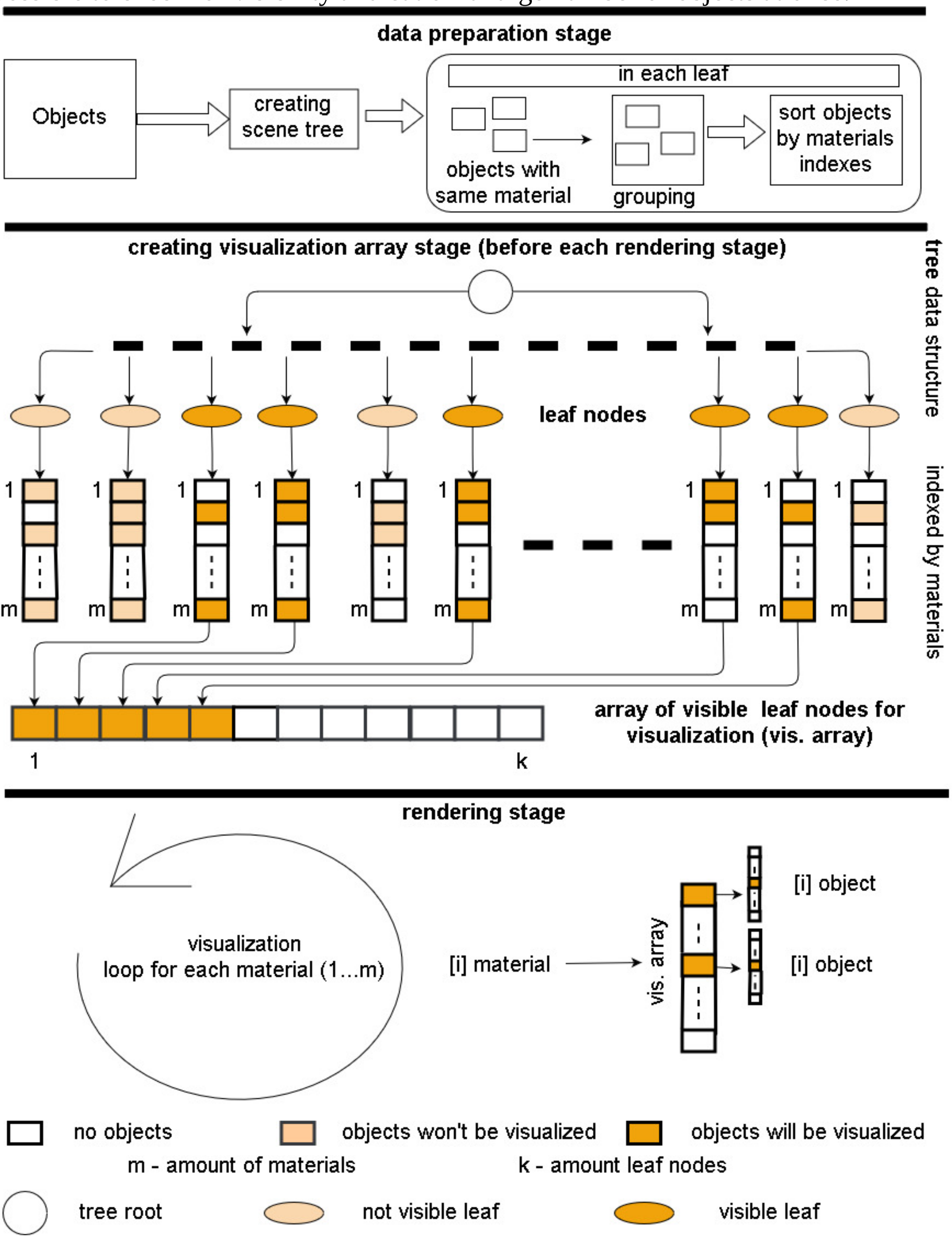
data preparation stage
Objects
creating scene tree
in each leaf
objects with same material
grouping
sort objects by materials indexes
creating visualization array stage (before each rendering stage)
tree data structure
leaf nodes
indexed by materials
1
m
array of visible leaf nodes for visualization (vis. array)
1
k
rendering stage
visualization loop for each material (1...m)
[i] material
vis. array
[i] object
[i] object
no objects
objects won't be visualized
objects will be visualized
m - amount of materials
k - amount leaf nodes
tree root
not visible leaf
visible leaf

**Figure 6.** The process of grouping objects using the same material.

Objects in the background, due to their remoteness, do not require high detail [11]. The possibility of associating various levels of model detail with one object was implemented. Thus, if there are several models of one object with different degrees of detail, the level of detail of the group of objects being visualized in the scene node will depend on the distance between the observer and this node.

The generated model of the urban environment is characterized by a large number of identical objects. Such objects may include vegetation (trees, bushes) and objects of small architectural forms (kiosks, stalls, benches, street lights, etc.). To visualize a large number of such repeating objects in leaf nodes, the Geometry Instancing [10] technology was used, which allowed visualizing multiple copies of the object in one drawing cycle. This approach reduces the exchange of information between the CPU and the GPU, thereby significantly increasing the speed of visualization of a large number of identical objects without loss of quality.

### *3.2.2 Materials optimization*

The principle of generating a model of the urban environment based on the attributive information from a digital map implies the creation of objects of the same type that have different geometric representations, but are characterized by the same type of surface material. An example of such objects are buildings and structures. In order to reduce the number of accesses to the graphics processor, an algorithm is implemented that grouping objects using the same material. The process is divided into three stages. First, at the data preparation stage, it is monitored to ensure that materials and textures are not duplicated in memory. Then a scene tree is formed. At the end of this process, the leaf nodes of the tree will contain scene objects. After the formation of a scene tree, in each of its leaf, objects with the same material are merged. Thus, in each leaf of a tree, one material corresponds to a maximum of one object. Next, the objects contained in the leaf are sorted by materials, so that the i object of the leaf corresponds to the i material. Before the formation of each frame in the process of visualization, a walk through the scene tree is performed. During this process, a list of leaf nodes that fall into scope is formed. Thus, using established correspondences, for each material, a set of objects ready for drawing in the current frame is established. The resulting structure allows direct access to objects using the current material. The general scheme of grouping by materials is presented in the figure 6.

The process of combining involves the objects that have common material. Models of urban environment objects often have a common type of surface material. The only difference is the color map (for example, the texture of the building facades). Therefore, the textures of objects with the same type of material are combined into one texture atlas [12]. Thus, when visualizing an urban environment model containing tens of thousands of buildings, the use of the proposed three-stage material optimization algorithm reduces the number of unique materials to just several dozen.

## 4. Low-level optimization

At the moment, GPU has overtaken CPU in terms of performance, and the performance growth rate continues to increase. Therefore, it is reasonable to move the computational load of the visualization process to the GPU and to minimize the amount of data transferred between the CPU and the GPU, since this data transfer is a real bottleneck in the system performance. In the framework of the OpenGL (DirectX) specification, manufacturers of video adapters provide ample opportunities for the effective use of the equipment’s potential for 3D visualization.

In particular, in the process of visualization of the urban environment model, VertexBufferObjects (VBO), IndexBufferObject (IBO), VertexArrayObject (VAO) technologies were used to upload the geometry of the urban environment model to the video device; hardware texture compression was used to speed up the loading of textures to video memory; triple buffering, for avoiding excessive synchronization of the CPU and GPU; MultiDrawIndirect, for reducing the number of accesses from CPU to GPU; programmable pipeline, Uniform Buffer (UBO), Shader Storage Buffer (SSBO) were used to generate the underlying surface, etc. [13]. The use of low-level optimization in the proposed optimization algorithms is presented in the figure 7.

## 5. Conclusions

The paper presents some methods and algorithms for optimizing the process of visualization that have been modified, taking into account the specific features of the urban environment model. These methods and algorithms have been implemented in software. This approach has significantly optimized the process of visualization of the urban environment model. Experimental studies have shown that the use of the proposed optimization algorithms provides on the average a tenfold increase in the performance of the visualization model of urban environment objects. The tests were conducted on a scene containing 10000 objects with a total of 7000000 triangles.. The performance was 4-6 frames per second (fps) before applying all optimizations, 56-64 fps - after. Technical characteristics of the test bench: AMD fx-6300 processor 3.5GHz, Nvidia GeForce GTX 1050 Ti video card 4 GB, 12 Gb DDR3 RAM.

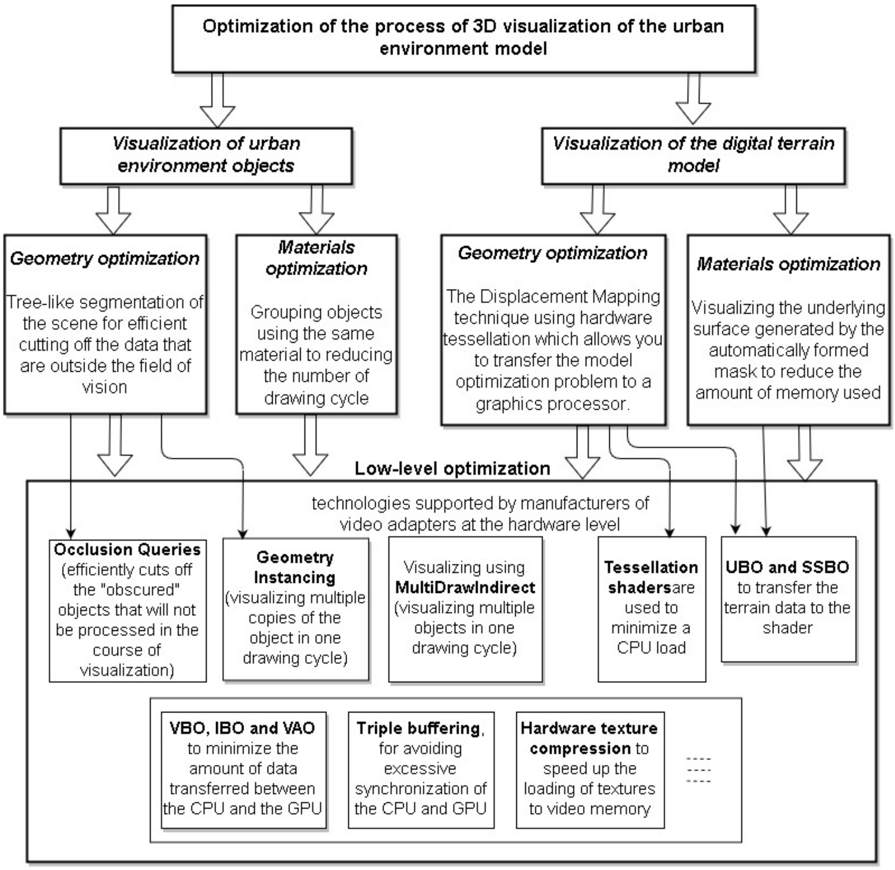
Optimization of the process of 3D visualization of the urban environment model
Visualization of urban environment objects
Visualization of the digital terrain model
Geometry optimization
Tree-like segmentation of the scene for efficient cutting off the data that are outside the field of vision
Materials optimization
Grouping objects using the same material to reducing the number of drawing cycle
Geometry optimization
The Displacement Mapping technique using hardware tessellation which allows you to transfer the model optimization problem to a graphics processor.
Materials optimization
Visualizing the underlying surface generated by the automatically formed mask to reduce the amount of memory used
Low-level optimization
technologies supported by manufacturers of video adapters at the hardware level
Occlusion Queries (efficiently cuts off the "obscured" objects that will not be processed in the course of visualization)
Geometry Instancing (visualizing multiple copies of the object in one drawing cycle)
Visualizing using MultiDrawIndirect (visualizing multiple objects in one drawing cycle)
Tessellation shadersare used to minimize a CPU load
UBO and SSBO to transfer the terrain data to the shader
VBO, IBO and VAO to minimize the amount of data transferred between the CPU and the GPU
Triple buffering, for avoiding excessive synchronization of the CPU and GPU
Hardware texture compression to speed up the loading of textures to video memory
----
----
----

**Figure 7.** Methods for optimizing the process of 3D visualization of the urban environment model.

## 6. References

[1] Vasin Yu G, Osipov M P and Tomchinskaya T N 2011 Development of Interactive Virtual Models of the Urban Landscape of the Historical Center of Nizhni Novgorod *Pattern Recognition and Image Analysis* **21(2)** 351-353

[2] Pechenkin V V and Korolev M S 2017 Optimal placement of surveillance devices in a three-dimensional environment for blind zone minimization *Computer Optics* **41(2)** 245-253 DOI: 10.18287/2412-6179-2017-41-2-245-253

[3] Choudhary K, Boori M S and Kupriyanov A 2017 Mapping and evaluating urban density patterns in Moscow *Computer Optics* **41(4)** 528-534 DOI: 10.18287/2412-6179-2017-41-4-528-534

[4] Osipov M P and Chekodaev O A 2018 Algorithms of automation of 3D modeling of urban environment objects using attributive information from a digital map *CEUR Workshop Proceedings* **2212** 359-365

[5] Barton R 2010 Modern Algorithms for Real-Time Terrain Visualization on Commodity Hardware *Geinformatics FCE CTU* **5** 5-22

[6] Samet R and Ozsavas E 2007 Optimization of Quadtree Triangulation for Terrain Models *Advanced Concepts for Intelligent Vision Systems. Lecture Notes in Computer Science* (Springer, Berlin, Heidelberg) **4678** 48-59

[7] Ulrich T 2002 Rendering massive terrains using chunked level of detail control *SIGGRAPH Course Notes* **3(5)** 14

[8] González C, Pérez M and Orduña J M 2017  Combining displacement mapping methods on the GPU for real-time terrain visualization *The Journal of Supercomputing* **73(1)** 402-413

[9] Suhaibah A, Uznir U, Francois A, Darka M and Alias A R 2013 Review of Spatial Indexing Techniques for Large Urban Data Management *Workshop Conference: International Symposium & Exhibition on Geoinformation* 23

[10] Pharr M and Fernando R 2005 *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-purpose Computation* (Addison-Wesley Professional First Edition edition) p 814

[11] Gong J, Zhu Q, Zhang H, Li X and Zhou D 2011 An adaptive control method of lods for 3D scene based on r-tree index *Acta Geodaetica et Cartographica Sinica* **4** 531-534

[12] Dai X, Xiong H and Gong J 2015 A multi-texture automatic merging approach for the 3D city models *Wuhan Daxue Xuebao (Xinxi Kexue Ban)* **40(3)** 347-352

[13] OpenGL Wiki URLt: https://www.khronos.org/opengl/wiki/Main_Page

**Acknowledgements**

This research was supported by the Russian Science Foundation, grant No. 16-11-00068.