Scientific Visualization, 2025, volume 17, number 5, pages 19 - 28, DOI: 10.26583/sv.17.5.03

Adapting Virtual Vegetation Samples to Multiobject Visualization of a New Type

P.Yu. Timokhin[1], M.V. Mikhaylyuk[2]

Scientific Research Institute for System Analysis of the National Research Centre "Kurchatov Institute

[1] ORCID: 0000-0002-0718-1436, p_tim@bk.ru
[2] ORCID: 0000-0002-7793-080X, mix@niisi.ras.ru

Abstract

The paper focuses on the visualization of vegetation arrays within virtual environment systems through a novel type of virtual object known as *multiplicable point cloud* (MPC). It addresses the challenge of adapting virtual vegetation samples, derived from 3D scans of real natural objects, to the MPC-format. The research highlights key discrepancies between these samples and the MPC-format, and presents a method to resolve these issues using the open-source CloudCompare software. Additionally, the paper proposes an output file format for the adapted virtual samples, utilizing a straightforward and easily interpretable PLY-syntax (Polygon File Format). Under the approbation of the proposed solutions, the adaptation and multiobject visualization of various virtual samples of tree- and grass-like plants were conducted. The results of the approbation affirm the expediency of the proposed method and approach, as well as their potential to effectively enhance the realism of virtual environments.

Keywords: virtual environment system, vegetation cover, multiobject visualization, point cloud, object coordinate system, PLY.

1. Introduction

An important direction in the development of modern virtual environment systems is the real-time visualization of plant cover prototypes [1, 2]. In particular, there is a high demand for visualizing arrays of tree- and grass-like vegetation both from a pedestrian's view and from a bird's-eye view. This is applicable in situational modeling [3, 4], wildfire prediction [5], and urban planning [6]. Such scenarios require millions of virtual plants, leading to huge geometric complexity in the prototype, which even high-end graphics cards are unable to handle.

A promising solution to this challenge is *multiobject visualization*, where the array of plants is described as a virtual vegetation sample along with sets of distinctive features[1] of its instances. The key feature of this approach is that the transformation of the sample into multiple instances is performed directly on the graphics processing unit (GPU) during the frame synthesis phase of the visualization. This eliminates the need to store a vast number of sample copies in video memory, significantly reducing the overall geometric complexity of the virtual prototype.

In many multi object solutions [7-9], polygonal 3D models are used as virtual plant samples, and the rendering of numerous instances is executed on a rasterization-based GPU graphics pipeline. While this approach enables the images of virtual vegetation arrays to be synthesized, it is associated with the generation of a large number of triangle (polygon)

[1] The basic set comprises data on the position, orientation, and scale of the instance.

threads, which becomes a bottleneck for real-time visualization systems. In study [10], within the framework of the multiobject forest visualization task, a novel approach was introduced for the first time: combining hardware-accelerated ray tracing [11] with tree models represented as *multiplicable* (multiply replicated) point clouds - hereafter referred to as MPCs. This solution overcame the limitations of previous methods and demonstrated promising potential for implementing plant diversity, close to real counterparts, in virtual environment systems. To unlock this potential, a broad collection of virtual vegetation samples adapted for MPC-visualization is required - that is, samples converted to comply with the MPC-format.

The present work addresses the challenge of adapting virtual samples derived from 3D scans of real-world vegetation (hereafter referred to as scan-samples). Section 2 provides an analysis of existing scan-samples. Section 3 presents a generalized method for adaptation of scan-samples to the MPC-format, including the stages of cleaning point cloud and unification of the local coordinate system. The adaptation is performed using a toolkit of the CloudCompare program complex [12], an open-source software with an intuitive interface - an important factor for up-to-date scientific and practical applications [13]. Section 4 reports the results of testing the proposed solution on a series of scan-samples representing both tree- and grass-like plant types, and also draws conclusions about the solution obtained.

2. Analysis of Vegetation Scan-Samples

At the beginning of our research, an analysis was conducted to assess the compliance of existing vegetation scan-samples with the requirements of the MPC-format [10]. The primary source of input data was Sketchfab [14], the largest online platform offering a wide range of digital 3D models, including various vegetation scan-samples (see Fig. 1).

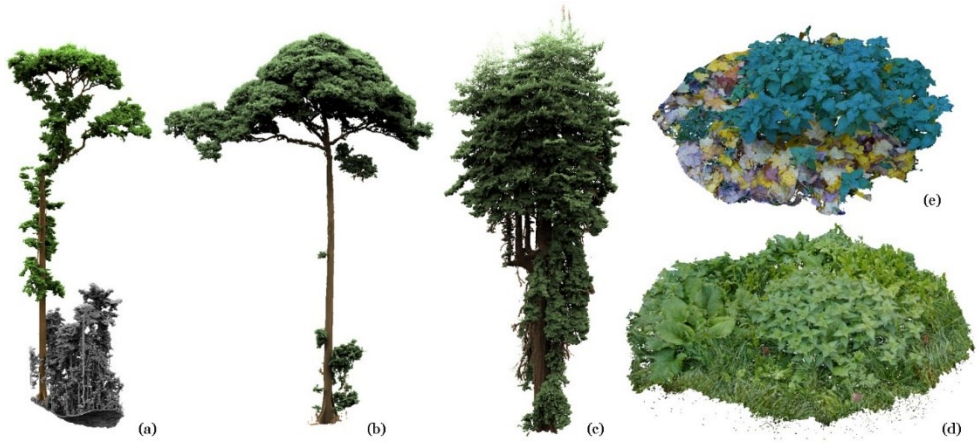

Fig. 1. Examples of vegetation scan-samples[2]: a), b) dipterocarps; c) sequoia; d) summer grass; e) autumn grass (images obtained using CloudCompare).

For this study, a number of scan-samples was taken, represented as colored point clouds having sufficient detail for an adequate description of geometric properties of the plant-object (on the order of 1-10 million points). The chosen scan-samples were evaluated based on two key criteria:

1) Coverage. The point cloud must provide full 360-degree visibility of the plant-object, and its axis-aligned bounding box (AABB) must fit the sample tightly. This is essential to ensure a vegetation array to be correctly composed of multiple scan-sample instances.

[2] a) «Dipterocarp in Maliau Basin» by kungphil (https://skfb.ly/ovL6M); b) «037_T0» by kungphil (https://skfb.ly/oEKZS); c) «Coastal redwood from Grove of Old Trees» by kungphil (https://skfb.ly/oxL6p); d) «Scan Unscannable: Grass Cloud 1» by Epic_Tree_Store (https://skfb.ly/TAF9); e) «Grass Autumn Update» by Epic_Tree_Store (https://skfb.ly/V6JS). These scan-samples are distributed under Creative Commons License Attribution 4.0 International (CC BY 4.0), http://creativecommons.org/licenses/by/4.0/.

2) Anchoring. The local coordinate system (LCS) of the point cloud must be aligned with the vertical axis of symmetry of the plant-object and anchored at the point where it connects to the ground. This is essential for the correct positioning, orientation, and scaling of sample instances during the multiplication process.

According to these criteria, three types of scan-samples were identified, and an assessment was conducted regarding their adaptability to the MPC-format.

Type I scan-samples almost fully satisfy the coverage and anchoring criteria (with possible local deviations such as residual point clusters or a slight offset of the LCS origin from the ground connection point). Such samples are often created under exploratory research projects using a combination of stationary and remote 3D scanning technologies. Examples include scan-samples of dipterocarps from the Maliau Basin reserve (see Fig. 1a, 1b) and sequoia from the "Grove of Old Trees" reserve (see Fig. 1c). Adapting these samples requires only minor corrections to the point cloud and LCS.

Type II scan-samples partially satisfy the coverage criterion: the point cloud of the plant-object includes fragments of surrounding vegetation, terrain, or isolated points (see Fig. 1d, 1e). The anchoring criterion is not met due to the arbitrary placement and orientation of the LCS within the scan-sample. Adapting such samples is possible, provided that all extraneous elements are fully removed and the LCS is re-anchored.

Type III scan-samples do not meet the coverage criterion: substantial parts of the plant-object are missing in the point cloud. Adapting such samples is extremely challenging and requires advanced data completion techniques (e.g., using neural networks [15]), which fall outside the scope of this work.

In the following sections, we will focus on the adaptation of Type I and Type II scan-samples.

3. Generalized Method for Scan-Sample Adaptation

Based on the results of the preliminary analysis, a generalized method for adapting scan-samples was developed. This method consists of two main stages: a) removing extraneous elements from the scan-sample (such as fragments of terrain, neighboring plants, or scattered points, see Fig. 2), and b) unifying the LCS of the scan-sample. Consider these stages in detail.

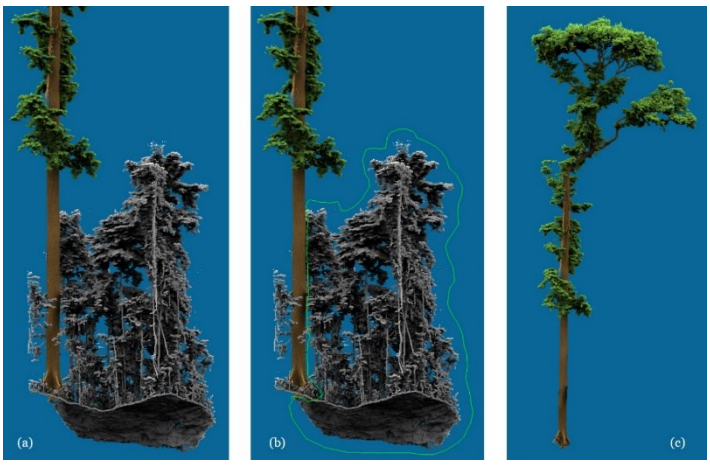

Fig. 2. Scan-sample cleaning: a) preparation of the working view; b) construction of the polygonal contour; c) final result.

Scan-sample cleaning is performed using the "Segment" tool, which is part of the CloudCompare package (available in the "Main tools" panel). The tool works by removing all points in the cloud whose projections fall within a polygonal contour drawn in the image plane. The cleaning procedure consists of the following steps. In the ***first step***, the scan-sample is rotated, scaled, and moved so that the region containing only extraneous elements

can be localized within the image plane (see Fig. 2a). In the ***second step***, a polygonal contour is drawn around this region (see Fig. 2b). The construction is performed manually using three basic operations: adding a vertex, deleting the last added vertex, and closing the contour [16]. In the ***third step***, the "Segment Out" modifier is applied to the polygonal contour. This operation splits the point cloud into two parts: the inner part (which becomes hidden) and the outer part. The inner part is deleted, and the outer part is subjected to the same three-step procedure again. These iterations continue until all extraneous elements are removed and a clean point cloud of the plant-object is formed (see Fig. 2c).

Unification of the LCS. In the MPC-format [10], the reference local coordinate system for a plant-object is the object coordinate system (hereafter referred to as OCS) used in the OpenGL graphics library [17]. By default, the origin of the reference OCS coincides with the origin of the world coordinate system, with the Z_{OCS} axis pointing toward the viewer, the Y_{OCS} axis pointing upward, and the X_{OCS} axis pointing to the right. The reference OCS is rigidly linked to the plant-object as follows: its origin is located at the center of the object's base (the approximate point where the stem transitions into the root), and the Y_{OCS} and Z_{OCS} axes represent the "up" and "forward" vectors of the object.

At the unification stage, we compute a translation and a set of rotations that align the original arbitrary LCS of the plant-object with the reference OCS defined by the MPC-format. This is implemented in the CloudCompare package using the following algorithm:

Algorithm for Unifying the LCS of a Plant-Object

1. Determine the coordinates P_{base} of the base center of the plant-object in the original LCS:

- open the "Camera Settings" tool (in the "Viewing tools" panel);
- align the scene's rotation center with the approximate base center of the plant-object by adjusting the coordinates in the "Rotation Center" section (see Fig. 3a);
- fix the P_{base} coordinates as the new scene rotation center using the "Camera Settings" tool (see Fig. 3b).

2. Align the base center of the plant-object with the origin of the reference OCS:

- open the "Camera Settings" and "Apply transformation" tools (from the "Edit" menu);
- transfer the P_{base} coordinates from the "Rotation Center" section of "Camera Settings" into the "Translation" section of "Apply transformation" (in the "Axis, Angle" tab, see Fig. 3c);
- invert the signs of the coordinates in the "Translation" section (this corresponds to a spatial shift);
- apply the "Apply transformation" tool to the plant-object.

3. Determine the rotation angles α, β and γ around the X_{LCS}, Z_{LCS}, and Y_{LCS} axes respectively, which align the orientation of the original LCS with the reference OCS:

- zero the scene rotation center coordinates in the "Rotation Center" section and the angle values in the {X1, Y2, Z3} slider block of the "Camera Settings" tool (see Fig. 3a);
- rotate the scene using the X1 slider to align the plant-object's approximate vertical axis with the image plane ($X_{OCS}Y_{OCS}$) (see angle α, in Fig. 4a);
- rotate using the Z3 slider to align the vertical axis of the plant-object with the "up" direction (Y_{OCS} axis) (see angle β in Fig. 4b);
- rotate using the Y2 slider to align the plant-object's "forward" direction with the "toward viewer" vector (Z_{OCS} axis) (see angle γ in Fig. 4c);
- record the resulting angles α, β and γ in the {X1, Y2, Z3} slider block in the opened "Camera Settings" window.

4. Rotate the space by angle α around the X_{LCS} axis:

- open the "Apply transformation" tool (go to the "Axis, Angle" tab; see Fig. 3c);
- set the X_{LCS} rotation axis by entering the coordinates {1, 0, 0} in the "Rotation Axis" section;
- transfer the angle α from the X1 slider in the "Camera Settings" tool to the "Rotation Angle" field in "Apply transformation";
- apply the "Apply transformation" tool to the scan-sample.

5. Perform the rotations by angles β and γ around the Z_{LCS} and Y_{LCS} axes, respectively, following the same procedure as in step 4.

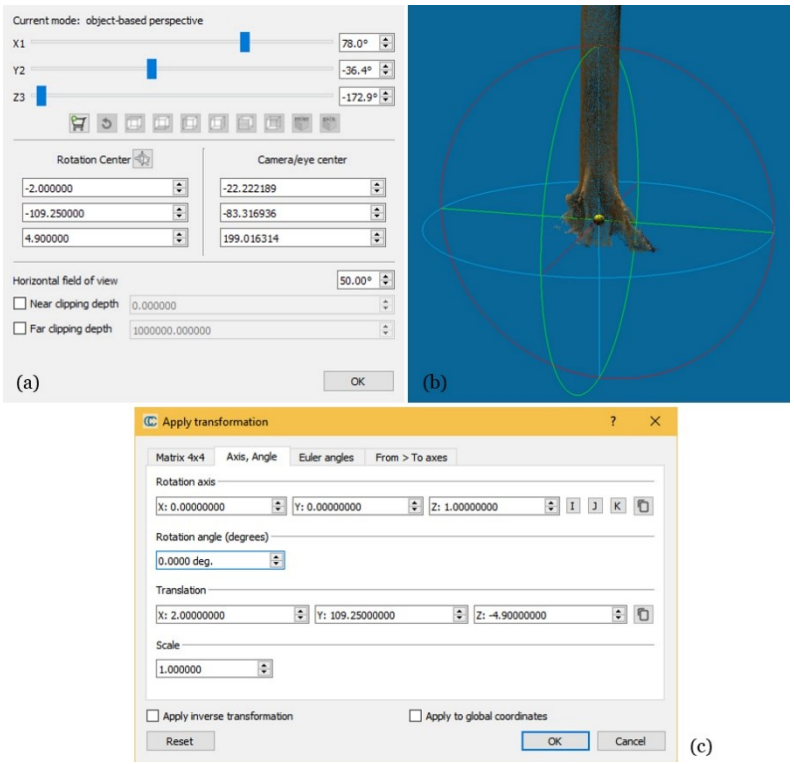

Fig. 3. Aligning the origin of the LCS with the OCS: a) "Camera Settings" window; b) aligning the scene rotation center with the plant-object's base center; c) "Apply Transformation" window.

As a result of executing the described algorithm, the original arbitrary LCS of the scan-sample is transformed into the reference OCS of the MPC-format, and the point cloud is assigned new coordinates according to the following expression:

$$P_{OCS} = R_Y R_Z R_X T P_{LCS},$$

where T is the translation matrix that shifts the space by the vector $\boldsymbol{d} = -(P_{base.x}, P_{base.y}, P_{base.z})$; R_X is the rotation matrix that rotates the space by angle α around the X_{LCS} axis; R_Z is the rotation matrix that rotates the space by angle β around the Z_{LCS} axis (obtained after rotation by α); R_Y is the rotation matrix that rotates the space by angle γ around the Y_{LCS} axis (obtained after the rotations by α and β).

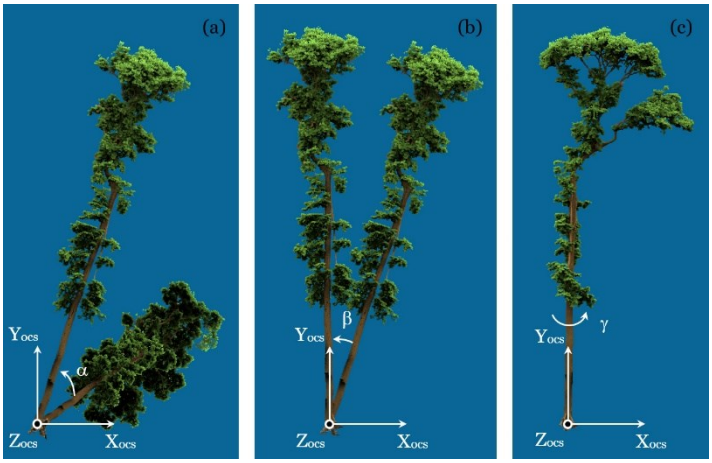

Fig. 4. Aligning the orientation of the LCS with the OCS: a) rotating the sample into the image plane; b) aligning the vertical axis of the sample; c) setting the frontal view of the sample.

Output format. In this study, the binary PLY (*Polygon File Format*) [18] is used as the output format for saving adapted scan-samples. Originally developed in 1994 at the Stanford University Computer Graphics Laboratory for storing descriptions of 3D polygonal objects, the PLY format is now widely used for storing the results of 3D scanning and is supported by major software packages such as MATLAB [19] and Blender [20].

A PLY file begins with a header - a sequence of text lines ending with carriage return characters. The header contains information about the PLY format type and version, comment lines, and a description of the "vertex" element of the point cloud. The vertex description includes its name and count, defined by the keyword "element", and a declaration of its attributes (position coordinates *x*, *y*, *z* and color components *red*, *green*, *blue*) using the keyword "property" (see Fig. 5).

```
1  ply
2  format binary_little_endian 1.0
3  comment Created by CloudCompare v2.11.1 (Anoia)
4  comment Created 29/06/2022 09:12
5  obj_info Generated by CloudCompare!
6  element vertex 1538165
7  property float x
8  property float y
9  property float z
10 property uchar red
11 property uchar green
12 property uchar blue
13 end header
```

Fig. 5. Example of a PLY file header for an adapted scan-sample.

After the end of the header (marked by the line "end_header"), the binary vertex data follows. The data is written in the order in which the vertex attributes were declared in the header: first the *x*, *y*, *z* coordinates and *red*, *green*, *blue* components of the first vertex, then in the same order attributes for the second vertex, and so on.

4. Results and Conclusions

To evaluate the adequacy of the proposed method, it was tested on a number of scan-samples of both tree- and grass-like vegetation, represented in Section 2. The testing included the preparation of PLY files for the adapted scan-samples and their multiobject visualization. Table 1 provides the coordinates of the shift vector $\boldsymbol{d}$ (in the original measurement units of the scan-sample) and the rotation angles α, β and γ (in degrees) that were used in the preparation of the scan-sample PLY files. Additionally, Table 1 includes the sizes of the scan-sample point clouds before and after the cleaning process, as well as the height h of the plant-object[3] (in meters) obtained after vertical alignment (see Fig. 4).

Table 1. Parameters of the adapted scan-samples

No.	Name	Number of points		Shift vector $\boldsymbol{d}$	Angles α, β, γ	Height h
		before	after			
1	Dipterocarpus 1[2,a]	5 000 000	1 538 165	(2, 109.3, -4.9)	(-90, 0, -110)	54
2	Dipterocarpus 2[2,b]	7 051 757	6 445 220	(20.4, 118.9, -6.6)	(-90, 0, 113.5)	30.4
3	Sequoia[2,c]	5 000 000	4 964 165	(9.7, 3.7, 0)	(-90, 0, 147)	51.6
4	Autumn grass[2,e]	1 092 394	280 549	(0.4, 0.4, -11.3)	(153, 1.2, 60)	0.2

On Figure 6, the appearance of the adapted scan-samples from Table 1 is shown, while Figure 7 illustrates an example of multiobject visualization of a grass cover synthesized from the PLY file of scan-sample No. 4. The grass cover visualization was carried out within a hy-

[3] The height h was determined as the size of the point cloud's AABB along the Y-axis (see the "Box dimensions" field in the "Properties" panel).

brid virtual scene that included polygonal models of a wall and ground surface, using a previously developed program complex [1]. The testing was performed at Full HD resolution on a laptop equipped with an Intel Core i5-11300H 3.1 GHz CPU, 8 GB RAM, and an NVidia GeForce RTX 3050 Laptop graphics card (4 GB VRAM, 2048 CUDA-cores, 16 RT-cores, NVidia DCH driver v. 529.04).

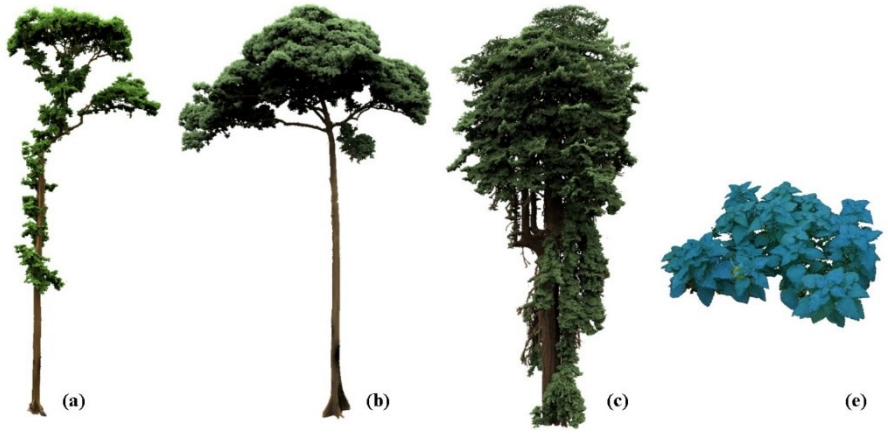

Fig. 6. Examples of scan-samples adapted using the proposed method (labels correspond to the scan-samples from Fig. 1).

The results confirmed that the proposed generalized adaptation method successfully eliminates deviations from the MOT format for scan-samples of the first and second types (see Section 2). It was also noted that most of the adapted scan-samples (except No. 1) lacked information about the actual height of the vegetation objects, which impeded their integration into virtual environments. The proposed solution enabled the recovery of this information and provided a foundation for the multiobject visualization of huge virtual grass cover (4 million instances).

The obtained results may be used for preparing source data for virtual environment systems, scientific visualization, video simulators, GIS, and more. As a direction for future work, an extension of the proposed method to the adaptation of polygonal plant models is planned.

Fig. 7. A frame of multiobject visualization of grass cover based on scan-sample No. 4, adapted using our method. The grass cover consists of 2K×2K scan-sample instances.

5. Acknowledgements

The publication is made within the state task of Scientific Research Institute for System Analysis of the National Research Centre “Kurchatov Institute” on topic No. FNEF-2024-0002 “Mathematical modeling of multiscale dynamic processes and virtual environment systems”.

References

1. Timokhin P.Yu., Mikhaylyuk M.V. Visualization of Hybrid Virtual Scenes Using Hardware-Accelerated Ray Tracing and Rasterization // Scientific Visualization. – 2024. – Vol. 16, No. 3. – pp. 60-70 (doi: 10.26583/sv.16.3.06)

2. Huang J., Lucash M.S., Scheller R.M., Klippel A. Walking through the forests of the future: using data-driven virtual reality to visualize forests under climate change // International Journal of Geographical Information Science. – 2020. – Vol. 35, No. 6. – pp. 1155–1178 (doi: 10.1080/13658816.2020.1830997)

3. Mikhaylyuk M.V., Kononov D.A., Loginov D.M. Modeling Situations in Virtual Environment Systems // Proceedings of the 23rd Conference on Scientific Services & Internet. – 2021. – Vol. 3066. – pp. 173–181 (doi: 10.20948/abrau-2021-6s-ceur)

4. Maltsev A.V. Integration of Physical Reality Objects with Their 3D Models Visualized in Virtual Environment Systems // Scientific Visualization. – 2024. – Vol. 16, No. 2. – pp. 97-105 (doi: 10.26583/sv.16.2.08).

5. Meng Q., Lu H., Huai Y., Xu H., Yang S. Forest Fire Spread Simulation and Fire Extinguishing Visualization Research // Forests. – 2023. – No. 14: 1371. – pp. 1-21 (doi: 10.3390/f14071371).

6. Thuvander L., Somanath S., Hollberg A. Procedural Digital Twin Generation for Co-Creating in VR Focusing on Vegetation // The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. – 2022. – Vol. XLVIII-4/W5-2022. – pp. 189–196 (doi: 10.5194/isprs-archives-XLVIII-4-W5-2022-189-2022)

7. Zhang X., Bao G., Meng W., Jaeger M., Li H., Deussen O., Chen B. Tree Branch Level of Detail Models for Forest Navigation // Computer Graphics Forum. – 2017. – Vol. 36, No. 8. – pp. 402-417 (doi: 10.1111/cgf.13088).

8. Kohek Š., Strnad D. Interactive Large-Scale Procedural Forest Construction and Visualization Based on Particle Flow Simulation // Computer Graphics Forum. – 2018. – Vol. 37, No. 1. – pp. 389-402 (doi: 10.1111/cgf.13304).

9. Garifullin A., Frolov V., Khlupina A. Approximate Instancing for Modeling Plant Ecosystems // Proceedings of the 31th International Conference on Computer Graphics and Vision. – 2021. – Vol. 3027. – pp. 95-104 (doi: 10.20948/graphicon-2021-3027-95-104).

10. Timokhin P.Y., Mikhaylyuk M.V. Multiobject Visualization of Vast Forests in Virtual Environment Systems // Programming and Computer Software. – 2025. – Vol. 51, No. 3. – pp. 198-206 (doi: 10.1134/S0361768825700082)

11. Rusch M., Bickford N., Subtil N. Introduction to Vulkan Ray Tracing // Ray Tracing Gems II, NVIDIA. – 2021. – pp. 213-255 (doi: 10.1007/978-1-4842-7185-8_16)

12. CloudCompare. 3D point cloud and mesh processing software. Open Source Project, 2004-2024, http://www.cloudcompare.org/.

13. Bondarev A.E. Experience of Constructing and Carrying Out the R&D "Scientific Visualization and Visual Analytics" at RTU-MIREA in 2024 // Scientific Visualization. – 2024. – Vol. 16, No. 4. – pp. 37-42 (doi: 10.26583/sv.16.4.04).

14. Sketchfab - The leading platform for 3D & AR on the web, https://sketchfab.com/.

15. Bornand A., Abegg M., Morsdorf F., Rehush N. Completing 3D point clouds of individual trees using deep learning // Methods in Ecology and Evolution. – 2024. – Vol. 15, No. 11. – pp. 2010–2023 (doi: 10.1111/2041-210x.14412).

16. CloudCompare v. 2.6.1, https://www.cloudcompare.org/doc/qCC/CloudCompare v2.6.1 - User manual.pdf.

17. Clevenger J., Gordon V.S. Computer Graphics Programming in OpenGL with C++ // 3rd Edition. – Mercury Learning and Information, Boston, Massachusetts. – 2024. – 589 p.
18. PLY - Polygon File Format, https://paulbourke.net/dataformats/ply/.
19. Point Cloud Processing: The PLY Format, MATLAB Documentation, 1994-2024, https://www.mathworks.com/help/vision/ug/the-ply-format.html.
20. Blender 4.3 Reference Manual, Importing & Exporting Files, Stanford PLY, https://docs.blender.org/manual/en/latest/files/import_export/ply.html.