UDC 004.925

ALGORITHMS AND ARCHITECTURE OF THE SOFTWARE SYSTEM OF AUTOMATED NATURAL AND ANTHROPOGENIC LANDSCAPE GENERATION

Levus Ye. V. – PhD, Associate Professor of Software Engineering Department, Lviv Polytechnic National University, Lviv, Ukraine.

Morozov M. Yu. – Postgraduate student of the Informatics Department, Technical University of Munich, Munich, Germany.

Moravskyi R. O. – Postgraduate student of the Software Engineering Department, Lviv Polytechnic National University, Lviv, Ukraine.

Pustelnyk P. Ya. – Postgraduate student of the Software Engineering Department, Lviv Polytechnic National University, Lviv, Ukraine.

ABSTRACT

Context. The problem of automation of the generation of natural and anthropogenic landscapes is considered. The subject of the research is methods of procedural generation of landscapes that quickly and realistically visualize natural and anthropogenic objects taking into account different levels of detail.

Objective. The goal of the work is to improve the rendering quality and efficiency of the procedural generation process of landscape surfaces at any level of detail based on the implementation of the developed method.

Method. The proposed method of visualization involves the construction of a natural landscape using Bezier curves and surfaces and manual editing of individual segments; use of software agents that are responsible for individual steps of generating anthropogenic objects; adaptation of anthropogenic objects to the characteristics of natural landscapes; containerization of three-dimensional objects, which is used in various steps to organize the storage and loading of objects efficiently. A generated heightmap based on the Perlin noise algorithm is used to construct surfaces on individual segments of the natural landscape. Landscape processing software agents are used to unify the design of algorithms for creating and processing information about anthropogenic objects. Correct application operation and error resistance is guaranteed due to the inheritance of a specific interface by all implementations of agents. Containerization with two-level caching ensures the efficiency of display detailing.

Results. The developed method is implemented programmatically, and its efficiency is investigated for different variants of input data, which to the greatest extent determine the complexity of visualization objects.

Conclusions. The conducted experiments confirmed the efficiency of the proposed algorithmic software and its viability in practice in solving problems of automated landscape generation. Prospects for further research include improvement and expansion of the algorithms for procedural landscape generation, functionality complication of manual visualized object processing, and division of individual objects into separate hierarchies of containers.

KEYWORDS: object visualization, segments, levels of detail, Bezier curves, software agents, containerization.

ABBREVIATIONS

LOD is a level of detail;

PRNG is a pseudorandom number generator;

DDA is a digital differential analyser.

NOMENCLATURE

ch is a number of generated segments;

n is a number of points in a row or a column of the heightmap;

h_{xy} is a surface height at the point *(x, y)*;

step is a step between nodes in the height map (*x* and *y*);

seed is a input parameter for generating pseudo-random numbers;

chunkSize is a segment side size;

curvesPerChunkSize is a number of Bezier curves modelling the surface per landscape segment;

bezierCurveDivisions is a number of pieces to divide the Bezier curve into, used in algorithms for converting to a three-dimensional model or elevation map;

$\tilde{P}_0, \tilde{P}_3$ are start and end points of the third-order Bezier curve, which is used to describe segments of the natural landscape;

C_1, C_2 are checkpoints that do not belong to the third-order Bezier curve used to describe segments of the natural landscape;

B(t) is a parametric Bezier curve;

$\beta_i^3(\tau)$ is a third order Bernstein polynomials;

a is a constant that specifies the "tightness" of the curve;

K is a constant used in the condition of continuity of curves;

d is a maximum allowed distance between cities in the case of laying a road between them;

r is a city range;

C_M is a model visualization complexity.

INTRODUCTION

As a result of constant technical and scientific progress, the tasks of modelling and generation of three-dimensional virtual worlds are gaining popularity not only in the game industry or cinema but also in the fields of education, science, architecture, design, and more. This trend necessitates not only the manual creation of highly realistic models of landscapes, natural and anthropogenic objects through available hardware but also the productive performance of their automatic generation and visu-

DOI 10.15588/1607-3274-2022-2-15

alization with different levels of detail. In addition to performance, other requirements for visualization software systems are essential. In particular: a sufficient level of uniqueness of anthropogenic objects; their automatic adaptation to the characteristics of the landscape on which they are located; high realism of visualization; the ability to supplement the system with new algorithms [1–3].

Automated systems for landscape visualization have become widespread compared to the manual creation of three-dimensional worlds due to their simplicity and speed. The quality of the images used depends on the accuracy of their perception by users of educational or scientific programs, the profitability of the game or movie, as well as the further development of appropriate software systems.

The main problems of the available solutions are their insufficient realism for large scales, limited means for providing details in the results, the price of full versions of the relevant software. Also, these software applications have closed-source, hard-to-extend libraries of landscape generation algorithms, which reduces the flexibility of development. Within projects that require relatively small landscapes, existing solutions can be applied effectively despite existing shortcomings.

This area of research is promising, as, on the one hand, it is necessary to improve the appearance of models, and, on the other hand, it is impossible to go beyond the possible number of calculations, i.e., it is necessary to develop new models and more optimized algorithms. Developing a new, optimized method of landscape generation is an urgent task both from a scientific and practical point of view.

The object of research is the task of generating natural and anthropogenic landscapes. The research subject is landscape generation methods that quickly and realistically visualize objects taking into account different levels of detail. The work aims to improve the quality of reflection and efficiency of the process of landscape surface generation at any level of detail.

1 PROBLEM STATEMENT

According to the purpose of work, the following tasks are solved:

1) determination of functional requirements to the system of landscape visualization based on the analysis of popular software systems of such purpose;

2) modelling of landscape surfaces;

3) modelling of anthropogenic objects according to the landscape;

4) development of a containerization method for objects in three-dimensional scenes with different levels of detail;

5) software implementation of methods for three-dimensional model visualization based on generated object data.

The inputs that directly describe the objects for the visualization system are:

– landscape segment size $chunkSize \in [2; 2^{10}]$;

– the size of the region of landscape segments $regionSize \in [chunkSize; 2^{16}]$;

– the number of curves modeling the surface per landscape segment $curvesPerChunkSize \in [2; 20]$;

– parameter of maximum height (also depth) when using noise functions $\max Height \in [200; 5000]$;

– landscape segment height offset for the noise function $zOffset \in [0;1000]$;

– parameter for the coordinate step in the noise function $noiseStepDivisor \in [500;5000]$;

– the distance to which the city extends its influence to the generation $cityEffectRange \in [chunkSize; regionSize]$;

– the minimum distance between roads in the city $roadMinDistance \in [0; chunkSize]$;

– minimum road width $roadMinWidth \in [0; chunkSize]$;

– maximum road width $roadMaxWidth \in [Min; chunkSize]$;

– maximum distance between cities for construction of long-distance communication $maxNearbyCityDistance \in [chunkSize; 2^{16}]$;

– the minimum width of the architectural element $architectureMinWidth \in [0; chunkSize]$;

– the maximum width of the architectural element $architectureMaxWidth \in [Min; chunkSize]$;

– the minimum area of the architectural element $architectureMinArea \in [0; chunkSize]$.

2 REVIEW OF THE LITERATURE

At the beginning of the researched problem (the 1980s), algorithms of calculation and representation of fixed primitive landscapes were considered. An algorithm for dividing the surface into triangles with its subsequent vertical deformation with the help of fractal noise was created [4], and methods for collecting, storing, and processing landscapes were developed [5].

The intensive advancement of software tools for visualization of natural and anthropogenic landscapes requires the development of visualization methods that provide high-quality images and, at the same time, their cost-effectiveness in terms of computing resources [3, 6–12].

The most common functions of software applications for landscape visualization include:

1) generating a three-dimensional landscape of a specific size based on parameters (e.g., mountain locality based on relief data);

2) loading an existing landscape for processing;

3) processing of the landscape utilizing manual re-editing with available tools and automated systems;

4) generating and processing landscape characteristics, such as water bodies, climate, soils, erosion and more;

5) visualizing the treated landscape, presenting, and preserving the obtained result;

6) generating the scenery of the landscape surface taking into account its characteristics (in particular, trees, hills, ravines, dams, etc. are generated);

DOI 10.15588/1607-3274-2022-2-15

7) generating three-dimensional objects of architecture and infrastructure taking into account the characteristics of the landscape:

7.1) designating anthropogenic areas of the landscape;

7.2) generating infrastructure objects, namely roads, communication lines, power grids, and other connections, taking into account the properties and curvature of the landscape;

7.3) generating architectural objects, in particular buildings for industrial, residential, commercial and cultural purposes, objects of transport and means of communication, taking into account the characteristics of the landscape;

7.4) generating anthropogenic scenery: lamps, greenery, parks;

7.5) visualizing generated three-dimensional objects;

7.6) saving the generated three-dimensional objects.

Generation and visualization of anthropogenic objects is the most extensive and complex process in software systems, which requires the most time and money [13, 14].

Software systems for generating three-dimensional anthropogenic objects have the following advantages:

– Flexibility of application. Objects generated in such systems are not static three-dimensional models but parameterized system objects with additional data that can be modified to achieve the desired result.

– Using templates. After manually changing the generation parameters of architectural and infrastructural objects, they can be saved as templates for later use, which speeds up the work with the system.

– Levels of detail. Generation algorithms support different levels of detail of generated three-dimensional objects, allowing more flexible use of the system.

– Integration into general-purpose software.

Despite the advantages described above, systems for generating three-dimensional objects of architecture and infrastructure have some disadvantages:

– Dependence of the visualization quality and efficiency on the machine's computing power.

– High cost of memory resources to ensure realism.

– Presence of artifacts. Errors occur when generating architectural and infrastructural objects. Generation results require manual verification and adjustment.

The popular main algorithms for containerizing three-dimensional scenes are Object Container Streaming from Cloud Imperium Games, Asset Streaming from Umbra Software, and Partitioned Scene Loading from Unreal Engine. Among their disadvantages is the high probability of data corruption during the transfer of containers; closed software source code, which makes it impossible to use the software in other systems; redundancy of information due to the need for metadata about the components in the files; uneven distribution of objects in sections [15–17].

3 MATERIALS AND METHODS

The proposed method of visualization involves:

1) construction of a natural landscape with the help of curves and Bezier surfaces and manual editing of individual segments;

2) the use of software agents that are responsible for individual steps in the generation of anthropogenic objects;

3) adaptation of anthropogenic objects to the characteristics of natural landscapes;

4) containerization of three-dimensional objects, which is used in various steps to organize the storage and loading of objects effectively.

The advantages of this method are:

– Realistic images due to high-quality consideration of landscape characteristics in the visualization of anthropogenic objects, which involves the use of optimal segmentation and containerization.

– Effective display detail based on containerization with two-level caching.

– Flexibility and ability to develop through the use of software agents.

The algorithm for constructing surfaces for individual parts of a natural landscape segment consists of the following steps:

Step I: Procedural generation of the primary heightmap.

The “chunkSize” parameter determines the size of the generated altitude map. The parameters for generating the heightmap based on the Perlin noise algorithm are set: the input parameter “seed” and the coordinates of the requested segment (x,y).

The height value at points (x, y) is calculated using the following function:

$$h_{xy} = Noise(seed,x,y). \tag{1}$$

Step II: Selection of control points based on the heightmap to specify Bezier curves describing a segment of the natural landscape.

The number of points required to describe one row or column of the heightmap is calculated:

$$n = 3 \cdot curvesPerChunkSize + 1. \tag{2}$$

The step for the heightmap is calculated as follows:

$$step = \frac{chunkSize}{3 \cdot curvesPerChunkSize}. \tag{3}$$

Using formulas (1)–(3) and the points of the segment P_{ij}, which are used to describe the curve, we can define the following statements:

$$P_{ij} = P(x_i, y_j, h_{x_i y_j}),\ i, j=1,\dots,n,\ x_1=0,\ x_i=x_{i-1}+step,$$
$$x_n=chunkSize,\ y_1=0,\ y_j=y_{j-1}+step,\ y_n=chunkSize.$$

It is assumed that the four points used to describe the third-order Bezier curve are taken from the heightmap points P_{ij}, and they pass through this curve. The landscape segments generation algorithm automatically calculates the control points $\{C_1, C_2\}$ for the selected four points

$\{\tilde{P}_0, \tilde{P}_1, \tilde{P}_2, \tilde{P}_3\}$ from the array of values P_{ij}, through which the curve passes:

$$C_1 = \frac{l_1^2 \tilde{P}_2 - l_2^2 \tilde{P}_0 + (2l_1^2 + 3l_1 l_2 + l_2^2)\tilde{P}_1}{3l_1(l_1 + l_2)}, \tag{4}$$

$$C_2 = \frac{l_3^2 \tilde{P}_1 - l_2^2 \tilde{P}_3 + (2l_3^2 + 3l_3 l_2 + l_2^2)\tilde{P}_2}{3l_3(l_3 + l_2)}, \tag{5}$$

where $l_i = \left|\tilde{P}_i - \tilde{P}_{i-1}\right|^a$, used value a=0.5.

After calculations (4), (5) the curve is described by control points: $\{\tilde{P}_0, C_1, C_2, \tilde{P}_3\}$.

For each pair of curves that have a common point, the control points are modified to ensure the continuity of parametric curves:

$$B_1'(t)=K\cdot B_2'(t). \tag{6}$$

If α*3* is the end point of curve $B_1(t)$, α*2* is the control point of curve $B_1(t)$, β0 is the starting point of curve $B_2(t)$, β1 is the control point of curve $B_2(t)$, then condition (6) is as follows:

$$|\alpha3-\alpha2|=K\cdot|\beta1-\beta0|. \tag{7}$$

All these points form the description basis for the segment in further processing (export or editing). The height values are found using the local coordinates of the calculated control points within the segment, rounding them to integers and using them as indices for the two-dimensional array of the heightmap. The control points are modified for each pair of curves with a common point to ensure the curve continuity.

Step III: Joins with previous segments. If there are adjacent segments to the requested one, the values of the current control points are modified so that there is a smooth interpolation on the boundaries of the segments (7).

Step IV: Formation of the vector polygonal model for the generated segment.

Every four calculated control points on each axis of the plane on individual fragments of the segment (Step II) form a bicubic Bezier surface, the parametric form of which is as follows:

$$\Omega(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} \beta_i^3(u)\cdot\beta_i^3(v)\cdot P_{ij}\ .$$

When converted to a vector polygonal model, each bicubic surface is considered as a set of control points that form Bezier curves in the x direction.

The input data for converting a landscape segment into a vector polygon model are the calculated control points in the Step II. The output data is a set of elementary polygons in the form of an array of vertices and an array of indices of these vertices.

Creating a vector polygonal model consists of the following calculations:

1) The primary Bezier curves are constructed based on the control points of the Bezier surface in the *x*-direction. The number of such curves is equal to *curvesPerChunksSize·n*. For example, the curve Q_1 is built based on points P_{i1}, i=1,…,4. The last control point of each curve constructed in the *x*-direction is the first for the next curve, except the last.

2) Depending on the *bezierCurveDivisions* parameter, the intermediate points L_m of each curve constructed in substep 1) are calculated. These intermediate points are stored in a one-dimensional array *M* of size N_m, which is calculated by the formula (8):

$$N_M = bezierCurveDivisions^2,\ \text{if}\ curvesPerChunkSize = 1$$
$$N_M = (curvesPerChunkSize\cdot bezierCurveDivisions - 1)^2,\ \text{if}\ curvesPerChunkSize > 1 \tag{8}$$

3) To determine simple polygons an array of indices *I* is formed for vertices from array *M*. The total number of vertices on the segment side is calculated by the formula (9):

$$k = bezierCurveDivisions,\ \text{if}\ curvesPerChunkSize = 1$$
$$k = curvesPerChunkSize\cdot bezierCurveDivisions - 1,\ \text{if}\ curvesPerChunkSize > 1. \tag{9}$$

The number of polygons is determined by the formula (10):

$$N_T = 2\cdot curvesPerChunkSize^2\cdot(bezierCurveDivisions - 1)^2. \tag{10}$$

Passing along the grid of polygons is in the x-direction the indices of the vertices *a, b, c, d* of two triangles are calculated according to formulas (9), (11). These triangles form a square (Fig. 1).

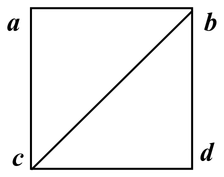


Figure 1 – Indexed polygons that form square

Indices (in the direction *x* – λ and in the direction y – μ) of the square formed by the triangles of the polygon are used to calculate:

$$\begin{aligned} a &= \lambda + (\mu - 1)\cdot k,\ \lambda=\overline{1, N_T},\ \mu=\overline{1, N_T} \\ b &= a + k, \\ c &= a + 1, \\ d &= a + k + 1. \end{aligned} \tag{11}$$

The results of the described calculations for the values of *bezierCurveDivisions* = 5 and *curvesPerChunksSize* = 1 are shown in Fig. 2.

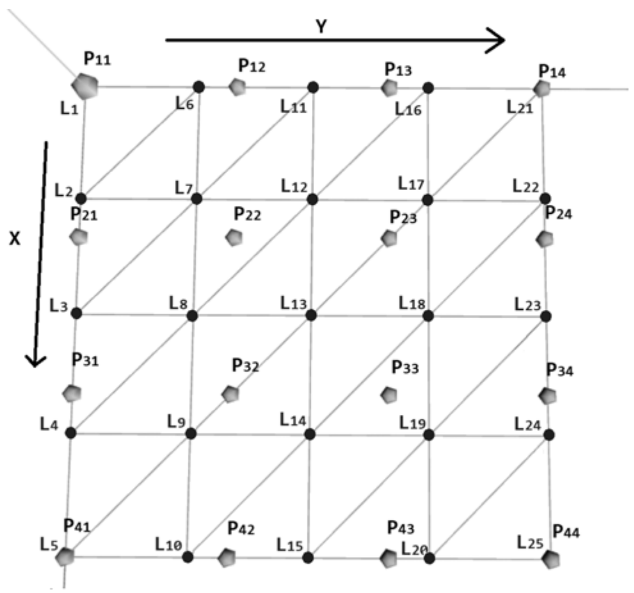


Figure 2 – Example of the vector polygonal model

Information about the fully generated and processed segment is returned to the caller for further processing (visualization or export).

Pre-generated or manually edited landscape segments can be exported to a three-dimensional polygonal model.

For the landscape generation module, it is necessary to retrieve information about the same landscape segment. To not perform generation steps each time, the caching logic for the manually created or edited information is used, which will ensure the processing of repeated queries on landscape segments as a simple search in the cache.

Landscape processing software agents are used to unify the design of algorithms for creating and processing information about anthropogenic objects. Correct application operation and error resistance is guaranteed due to the inheritance of a specific interface by all implementations of agents.

At the user's request, the system prepares the landscape segments to create anthropogenic objects and alternately uses pre-selected agents. After that, the system cache of landscape segments is updated, and the response is returned to the user (Fig. 3).

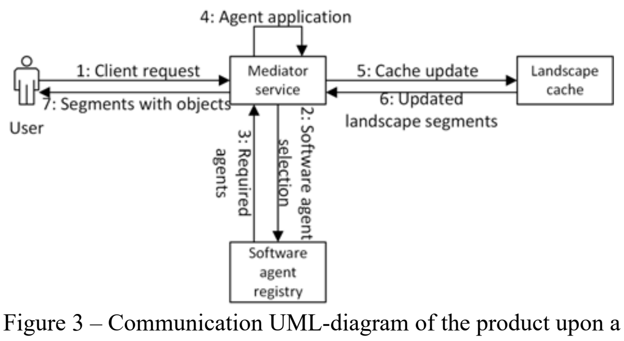


Figure 3 – Communication UML-diagram of the product upon a request from the user

The algorithms and the corresponding software agents that use them are responsible for the following steps:

1) preliminary preparation the landscape segment;

2) determination the city locations;

3) construction of transit roads within cities;

4) construction of roads between cities;

5) determination of road intersections;

6) determination of the boundaries of urban city blocks by crossroads and transit roads;

7) construction of architecture for each city block of the city;

8) algorithms for building three-dimensional models adapting to the landscape based on the received information about objects.

Based on these steps, software agents can create a variety of information about architectural and infrastructural objects and ensure the high performance of the application.

The first step in creating information is to obtain the parameters that are needed for the generation. After that, the user needs to get a prepared landscape segment with the coordinates specified in the request. The preparation includes creating a segment hull, determining its identification number, and obtaining its heightmap. The heightmap is used to adapt to the properties of the landscape when creating three-dimensional models of different levels of detail.

The creation of anthropogenic objects begins with the generation of cities. The essence of this algorithm is the choice of the city location within the region. After execution, the algorithm returns an array of cities belonging to the segment. Each entity of the city has filled in information belonging to the corresponding data structure. The algorithm consists of several sequential steps that use PRNG (Fig. 4).

The algorithm for laying roads in the city depends directly on the type of city generated in the previous step. The work created a type of city with Manhattan markings, i.e., parallel roads and rectangular city blocks. Laying roads for such a city is reduced to determining the distances between parallel streets, creating roads, and cutting roads outside the segment (Fig. 5).

For visual correctness and realism, the generation of long-distance connections requires a preliminary generation of roads in the city. Then intercity roads will join them, correctly continuing their direction.

In order to correctly build long-distance connections, the user needs to have a list of city roads in their boundary segments. This list is acquired by passing the segments in the DDA algorithm towards another city to check whether a particular segment is on the edge. Next, Bezier curves are built to pave a smooth road.

The algorithm for determining intersections searches for road intersections according to the formula for the intersection of two segments. It is key to the next step in defining city blocks, as roads and intersections can quickly identify their boundaries.

An algorithm for building city blocks is needed to determine the boundaries within which architectural objects can be built. City blocks are built between roads and intersections, which ensures the absence of roads and buildings clipping.

The algorithm for constructing city blocks is somewhat more complex than the previous ones. In the case of expanding the system by other software agents, it must correctly build city blocks at complex intersections and city forms. It is based on the Delaunay triangulation algorithm, which optimally divides a two-dimensional array of vertices into triangles, where the vertices of the city intersection are taken as vertices. From these triangles, city blocks are then formed by merging adjacent ones (Fig. 6).

Since the points-nodes of a nonconvex polygon give the city block boundaries, the algorithm for determining the city block boundaries is reduced to the construction of cyclic paths on the grid of intersections and roads that form a connected graph. Users can use two methods to provide this feature: inspecting intersections along roads and looking for closed loops, or triangulating a graph and then merging triangles into polygons. Since the first option requires the construction of graph edges on roads between different segments, which is computationally more complex, the second option was chosen, namely the Delaunay triangulation. This method of dividing a grid of points into triangles fulfils the task and allows users to quickly calculate adjacent triangles, making it easier to combine them. Once the triangles are combined into polygons, the city block information is stored in the segment.

Unlike previous algorithms, the algorithm for building architecture in a city block works within the boundaries of the city block, not the landscape segment. It arranges architectural elements by selecting their coordinates and type using PRNG.

The last step is to clean up unnecessary information in the landscape segment. Roads that do not border intersections or city blocks are discarded, intersections are merged with roads, and temporary additional information used by a software agent is removed.

To ensure the objects' realism, adaptation to the surrounding landscape takes place, the advantages of which are determined by the division of the landscape into rectangular segments. The algorithms use cubic spline interpolation to determine the height of the terrain from the nearest control points of the simulated surfaces. At the same time, the small size of the segment allows users to make queries and calculations quickly. Caching the heightmap in containers allows the system not to calculate it every time. In addition, taking into account the height values at the points of the segment, the number of the order of 10^8 due to caching allows avoiding display errors (caves under the ground, overhangs in cliffs, etc.). Thus, the speed of adaptation is not limited by the size of the generated landscape, and, accordingly, the memory consumption – by the number of control points.

The developed containerization algorithms provide the following features:

1) creating a hierarchy of containers for a given three-dimensional scene;

2) storing three-dimensional models and other containers inside a container;

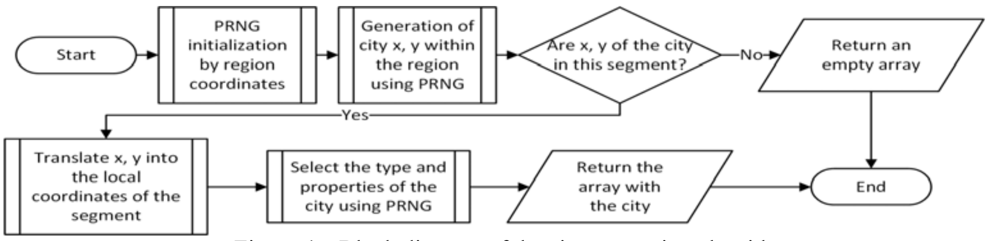

Start
PRNG initialization by region coordinates
Generation of city x, y within the region using PRNG
Are x, y of the city in this segment?
No
Return an empty array
Yes
Translate x, y into the local coordinates of the segment
Select the type and properties of the city using PRNG
Return the array with the city
End

Figure 4 – Block diagram of the city generation algorithm

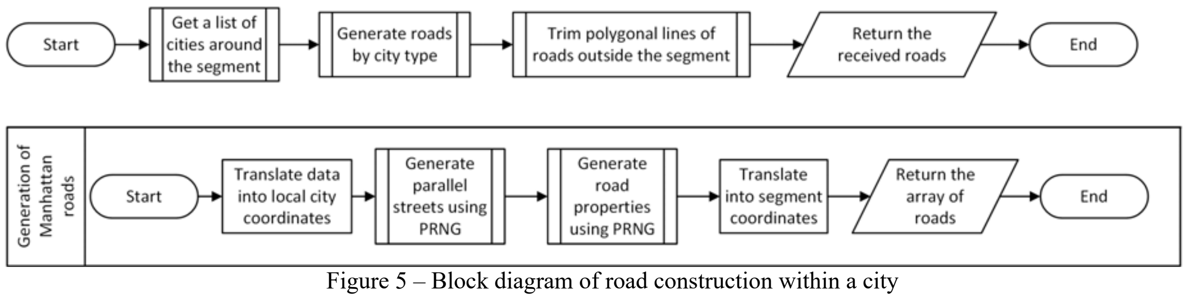

Start
Get a list of cities around the segment
Generate roads by city type
Trim polygonal lines of roads outside the segment
Return the received roads
End
Generation of Manhattan roads
Start
Translate data into local city coordinates
Generate parallel streets using PRNG
Generate road properties using PRNG
Translate into segment coordinates
Return the array of roads
End

Figure 5 – Block diagram of road construction within a city

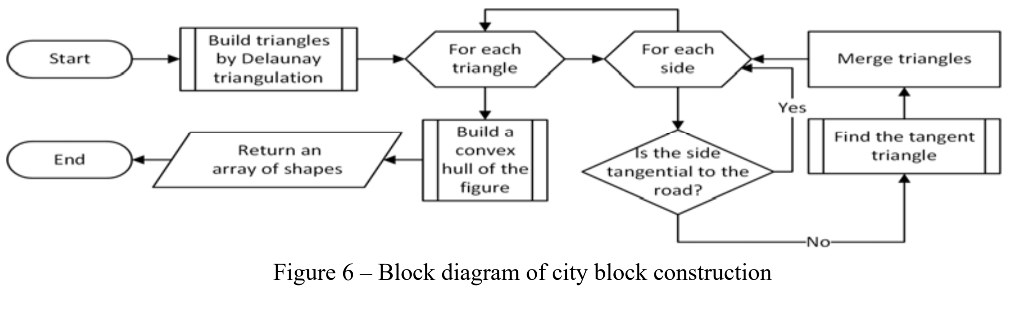

Start
Build triangles by Delaunay triangulation
For each triangle
For each side
Merge triangles
Yes
Is the side tangential to the road?
Find the tangent triangle
No
Build a convex hull of the figure
Return an array of shapes
End

Figure 6 – Block diagram of city block construction

3) calculating bounding volume for each container, which contains all three-dimensional models of the container;

4) creating a two-tier cache to optimize the process of loading and storing containers (the first tier of the cache stores objects in RAM, the second tier – in the file system);

5) saving containers in a file system in binary format (serialization);

6) saving and loading different levels of detail of objects inside containers.

The container is presented as a data model and contains information about the segment, its three-dimensional model, the current level of detail, the identifier, and other containers (Fig. 7).

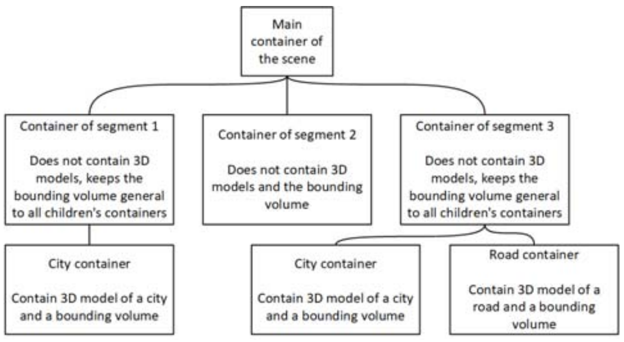

Main container of the scene
Container of segment 1
Does not contain 3D models, keeps the bounding volume general to all children's containers
Container of segment 2
Does not contain 3D models and the bounding volume
Container of segment 3
Does not contain 3D models, keeps the bounding volume general to all children's containers
City container
Contain 3D model of a city and a bounding volume
City container
Contain 3D model of a city and a bounding volume
Road container
Contain 3D model of a road and a bounding volume

Figure 7 – Container hierarchy example

The architecture of the software system that implements the described algorithms is a session-based client-server. There is a three-tier model based on software agents for creating and editing the properties of anthropogenic objects on the server side. The highest model tier, the service layer, is implemented based on the REST paradigm, which is used at the endpoints handlers of the session server for the software system in which the application is integrated. The business logic tier is at the middle tier with predefined algorithms for creating information about three-dimensional anthropogenic objects and a set of software agents for processing landscape segments. The lowest tier contains the data layer with a description of the data structures used in the application.

The software system supports the construction of bounding volumes and different levels of detail, resulting in a reduction in the data amount that needs to be downloaded. An efficient binary format is used for storing containers. The cache implementation consists of two levels and allows you to optimize the process of saving three-dimensional models. All this allows to speed up the rendering of virtual scenes, which is an essential indicator for the systems of generation and visualization of natural and anthropogenic landscapes.

4 EXPERIMENTS

The developed algorithms are software pieces implemented in the form of the LandscapeGen software system, the server part of which consists of three components (Fig. 8):

– LandscapeGen: Terrain – a component that contains services for generating landscape surfaces using Bezier curves and planes;

– LandscapeGen: Infrastructure – a component that provides services for generating anthropogenic objects of architecture and infrastructure based on ready-made landscape data;

– LandscapeGen: Containerization – a component that provides services for containerization of three-dimensional objects, their storage, and loading.

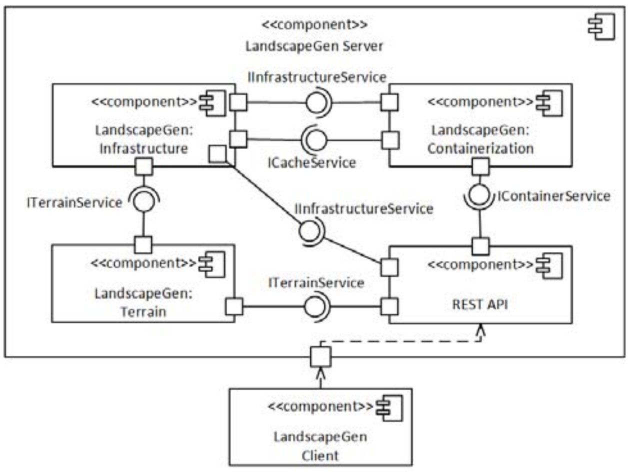


Figure 8 – Component diagram of LandscapeGen software system

The experiments were performed on a computer with the following characteristics: CPU Intel I7 9750H, GPU Nvidia 1660TI, RAM 16GB DDR4, Drive SSD NVME2 1TB. City centres were selected as the locations where the generation takes place in terms of the most significant computing power requirements. Parameters that significantly affect the generation time and do not give only a visual difference were selected for experiments. These include city range (r), the maximum allowed distance between cities in case of road construction (d), level of detail, and the number of generated segments (ch).

The complexity of the visualization model is calculated taking into account the maximum possible level of detail ($lod_{\max}$):

$$C_M = \frac{r \cdot d \cdot ch \cdot \left(lod_{\max} - lod + 1\right)}{256}.$$

A total of about 300 visualization experiments were performed for the following cases: without the containerization cache use, with the use of cache in RAM and hard drive. Used $lod_{\max}$=5, implemented levels of detail– min, low, medium, high, max. The minimum lod is for viewing the song from a distance.

The time costs for visualization of one segment for different d, r, ch at their maximum possible values, 4096, 1024, 25, respectively, were recorded (Table 1).

5 RESULTS

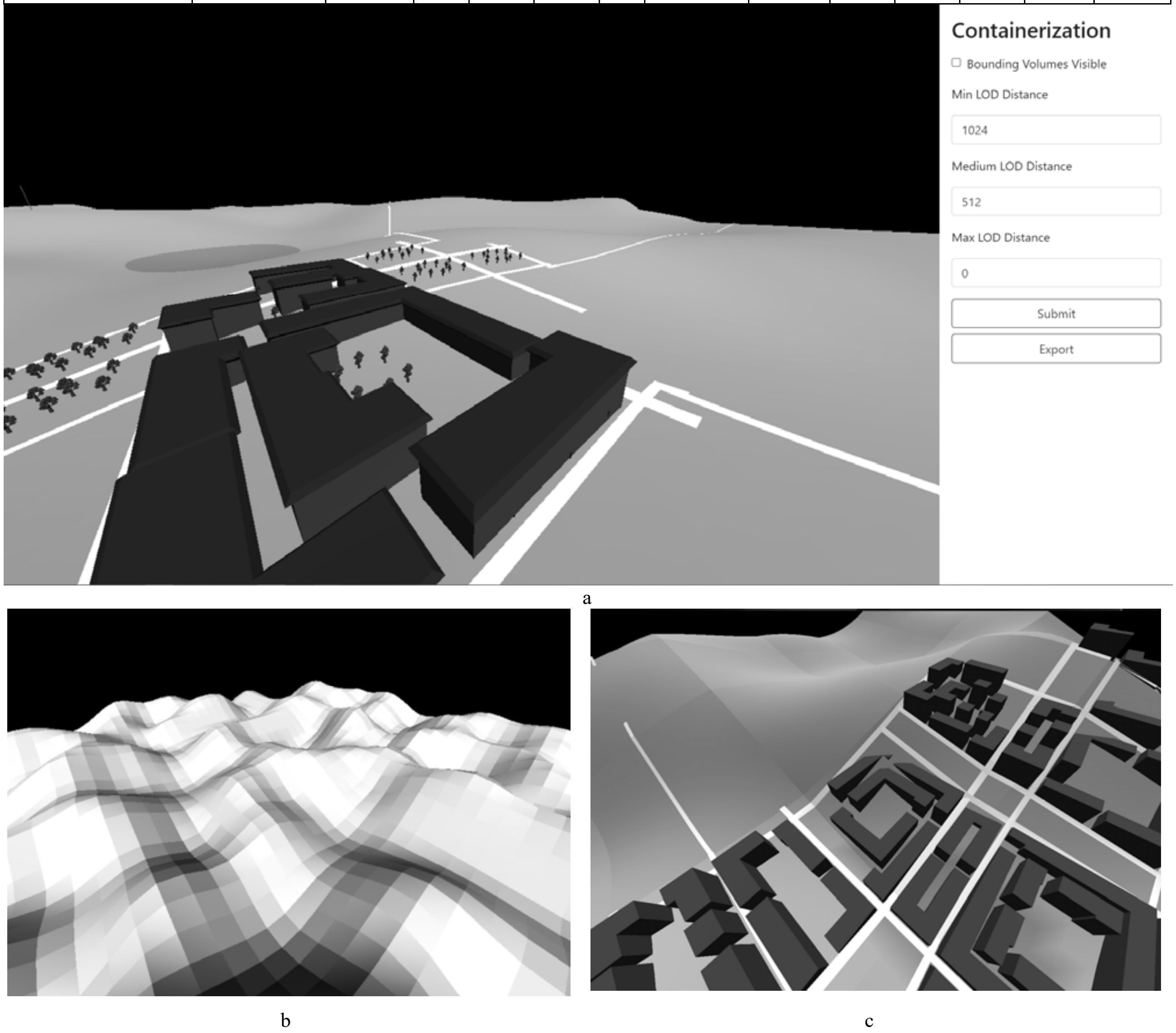
Anthropogenic and natural landscapes visualized by the LandscapeGen system are characterized by the absence of artifacts, adaptation of anthropogenic objects to natural landscapes (Fig. 9). Noise functions, which return the height value at the specified coordinates, are used to ensure the natural landscape automatic generation of the virtual world. A programming interface from the noisejs library is used to work with specific functions implementations: Perlin noise and Simplex noise.

Among the parameters that determine the quality of the obtained images used for modelling natural surfaces are: the number of Bezier curves per side of the segment, the noise reduction factor for landscape generation, the number of control points per segment. The use of these settings is not limited by the amount of memory, as caching methods are used.

The dependence of time costs for visualization of natural and anthropogenic landscapes on the complex characteristics of the complexity of the visualization model is analysed (Fig. 10).

Table 1 – Comparison of time spent for different types of visualization

Visualization type	**Average time for segment (μs)**	**Maximum time (μs) for the segment and the corresponding parameters at which it is achieved**						**The minimum time (μs) for the segment and the corresponding parameters at which it is achieved**					
	$time_{avg}$	***$time_{\max}$***	***C_M***	***d***	***r***	***ch***	***lod***	***$time_{\min}$***	***C_M***	***d***	***r***	***ch***	***lod***
Without the containerization cache use	68424	205935	192	4096	512	1	0	75084	108	4608	768	1	4
RAM cache	16	52	162	4096	1024	1	3	2	1600	4096	512	25	4
Hard drive cache	9133	98943	192	4352	1024	1	0	895	3400	4352	1024	25	4



Containerization

☐ Bounding Volumes Visible

Min LOD Distance: 1024

Medium LOD Distance: 512

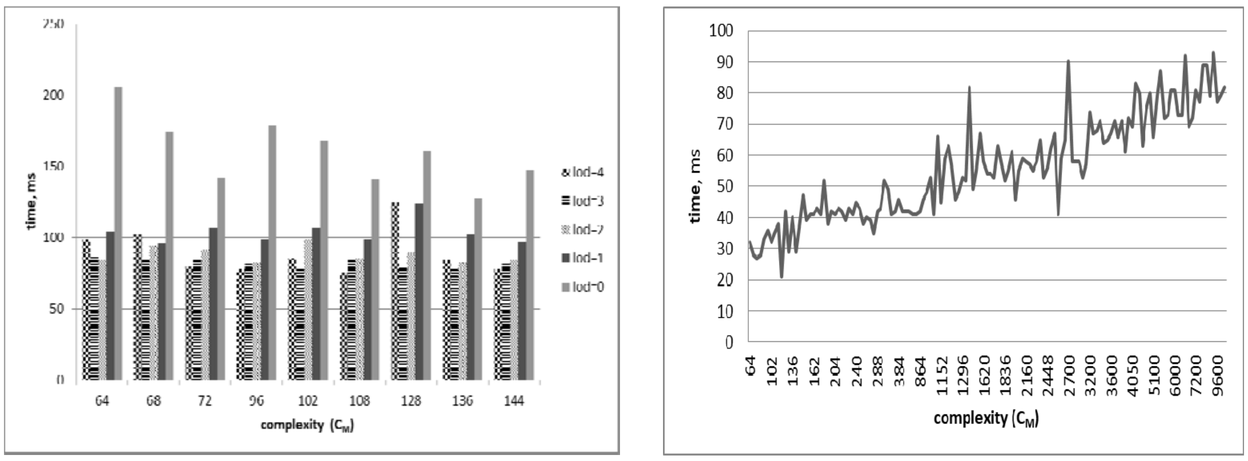
Max LOD Distance: 0

Submit

Export

Figure 9 – Examples of visualization:
a – anthropogenic and natural objects in the LandscapeGen program window; b – relief of Bezier surfaces; c – visualization of the urban landscape adapted to a relief (16 curves per segment, Perlin noise smoothing factor 500)

© Levus Ye. V., Morozov M. Yu., Moravskyi R. O., Pustelnyk P. Ya., 2022
DOI 10.15588/1607-3274-2022-2-15



a b

Figure 10 – Generation time dependency on the complexity of objects:
a – for one segment without the use of caching; b – for one or more segments with RAM caching

6 DISCUSSION

The obtained results of the LandscapeGen software system confirm the effectiveness of the developed method of visualization using containerization.

The average rendering time for hard disk caching experiments is about seven times shorter than the rendering time without caching. At the same time, using cache in RAM reduces rendering time by about 500 times compared to hard disk caching.

Most time is spent on non-caching cases for a minimum level of detail. Moreover, for insignificant values of complexity (up to 144), caching gives an advantage of 4 times the display time (Fig. 7, a, b).

A dramatic increase in the complexity of the visualization model does not significantly affect the visualization time of one segment (Fig. 7, b). In the case of using caching, increasing the complexity of the visualization model by 150 times increases the time only by three. Therefore, the system scales well with different complexity of visualization models.

The proposed method efficiently stores and loads three-dimensional objects, which is especially important to consider the trends of increasing the volume of three-dimensional scenes and growing requirements for detailing objects.

CONCLUSIONS

A unique feature of the developed software system LandscapeGen is the generation of three-dimensional objects with different levels of detail based on software agents, taking into account the characteristics of the landscape. The architecture of the system allows expanding the functionality of the system further.

The scientific novelty of the obtained results lies in developing a method for landscapes surfaces generation, which allows controlling the quantitative and qualitative indicators of modelling, resulting in increased realism of the display result and productivity of the visualization process at different levels of detail.

The practical significance of obtained results lies in developing a software system for the automatic generation of natural and anthropogenic landscapes. The obtained visualizations can be used as a basis for further creation of virtual landscapes, video, photo, and game materials by landscape designers, artists, developers of virtual worlds.

Prospects for further research are the development of algorithms for the procedural generation of natural landscapes, particularly the addition of all ecosystems and appropriate algorithms for adaptation to natural landscapes.

ACKNOWLEDGEMENTS

This paper uses the results obtained by the authors in 2019–2021 at the Department of Software of the National University “Lviv Polytechnic”. The authors are grateful to the department’s teaching staff for their active participation in the discussion and support of the scientific circle “Algorithms and software of computer graphics systems”, understanding of the importance of applied research of geometric modeling theory for computer graphics systems. Practical results of the research are received within the limits of performance of complex bachelor’s qualification work by graduates of the department – Morozov M., Pustelnyk P., Moravskyi R. under the direction of the senior lecturer Levus Ye. V. The results of research by Morozov M. Yu. were taken into account for the DAAD Study Scholarship – Master Studies for All Academic Disciplines, 2021/22.

All authors state that they do not receive financial support to conduct research on this topic.

REFERENCES

1. Doran J., Parberry I. Controlled Procedural Terrain Generation Using Software Agents, *IEEE Transactions on Computational Intelligence and AI in Games*, 2010, Vol. 2(2), pp. 111–119. DOI: 10.1109/TCIAIG.2010.2049020
2. Morozov M. Yu., Levus Ye. V., Moravskyj R. O., Pustelnyk P. Ya. Heneruvania landshaftiv dlia cferychnykh poverkhon: analiz zavdannia ta variant vyrishennia, *Naukovui visnyk NLTU Ukraiiny,* 2020, Vol. 30. No. 1, pp. 136–141. DOI: 10.36930/40300124
3. Biljecki F., Kumar K., Nagel C. CityGML Application Domain Extension (ADE): overview of developments, *In: Open Geospatial Data, Software and Standards 3.1* (Dec. 2018), P. 13. DOI: 10.1186/s40965- 018 - 0055 - 6

4. Gavin S. P. Muller. The definition and rendering of terrain maps/ Gavin S. P. Muller// *ACM SIGGRAPH Computer Graphics*, 1986, Vol. 20, No. 4, pp. 39–48. DOI:10.1145/15886.15890
5. Ebner H., Reinhardt W., and Hobler R. Generation, Management and Utilization of High Fidelity Digital Terrain Models, *XVIth ISPRS Congress Technical Commission III: Mathematical Analysis of Data Supplements*, 1988, Kyoto, Japan, 27(B11), pp. 556–566.
6. Brasebin M. S. Christophe, F. Jacquinod, A. Vinesse, and H. Mahon.3D Geovisualization & Stylization to Manage Comprehensive and Participative Local Urban Plans, *In: ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences IV–2/W1*, October (2016), pp. 83–91. DOI: 10.5194/isprs-annals- IV-2- W1-83-2016.
7. Cherradi G., El Bouziri A., Boulmakoul A., and Zeitouni K. Real-Time HazMat Environmental Information System: A micro-service based architecture, *In: Procedia Computer Science,* 109.June (2017), pp. 982–987. DOI: 10.1016/j.procs.2017.05.457
8. Merino L., Fuchs J., Blumenschein M., Anslow C., Ghafari M., Nierstrasz O., Behrisch M., Keim D.A. On the impact of the medium in the effectiveness of 3D software visualizations, *IEEE Working Conference on Software Visualization (VISSOFT).* Shanghai, China, 2017, pp. 11–21. DOI: 10.1109/VISSOFT.2017.17
9. Teng E., Bidarra R. A semantic approach to patch-based procedural generation of urban road networks, *FDG '17: 12th International Conference on the Foundations of Digital Games, August 2017, proceedings*, pp. 1–10.
10. Burch M., Wallner G., Arends S. T. T. and Beri P. Procedural City Modeling for AR Applications, *24th International Conference Information Visualizations (IV),* 2020, pp. 581–586. DOI: 10.1109/IV51561.2020.00098
11. Zhang X., Zhong M, Liu S, Zheng L, Chen Y., Zhang X, Zhong M, Liu S, Zheng L, Chen Y. Template-Based 3D Road Modeling for Generating Large-Scale Virtual Road Network Environment, *ISPRS International Journal of Geo-Information*, 2019, 8(9):364. DOI: 10.3390/ijgi8090364
12. Kelly G. and McCabe H. Citygen: An interactive system for procedural city generation, *In Fifth International Conference on Game Design and Technology*, 2007, pp. 8–16.
13. Dinkov D., Vatseva R.3D modelling and visualization for landscape simulation, *Proceedings of the 6th International Conference on Cartography and GIS.* Albena, 2016, pp. 320–333.
14. Klimke J. Döllner J. Service-oriented Visualization of Virtual 3D City Models, Directionsmag.com. [Electronic resource]. Access mode: 2012. URL: https://www.directionsmag.com/article/1873
15. Object Container Streaming, *Star Citizen,* 2021. URL: https://starcitizen.tools/Object_Container_Streaming
16. Bushnaief J., Czatrowski P. Solving Visibility and Streaming in The Witcher 3: Wild Hunt with Umbra 3, *GDC Vault.* 2014. URL: www.gdcvault.com (data zvernennja: 22.2.2021).
17. Level Streaming Overview // Unreal Engine. 2021. URL: https://docs.unrealengine.com/en-US/BuildingWorlds/LevelStreaming/Overview/index.html.

Received 01.11.2021.
Accepted 16.02.2022.

УДК 004.925

АЛГОРИТМИ ТА АРХІТЕКТУРА ПРОГРАМНОЇ СИСТЕМИ АВТОМАТИЗОВАНОЇ ГЕНЕРАЦІЇ ПРИРОДНІХ ТА АНТРОПОГЕННИХ ЛАНШАФТІВ

Левус Є. В. – канд. техн. наук, доцент кафедри програмного забезпечення Національного університету «Львівська політехніка», Львів, Україна.

Морозов М. Ю. – магістрант кафедри інформатики Мюнхенського технічного університету, Мюнхен, Німеччина.

Моравський Р. О. – магістрант кафедри програмного забезпечення Національного університету «Львівська політехніка», Львів, Україна.

Пустельник П. Я. – магістрант кафедри програмного забезпечення Національного університету «Львівська політехніка», Львів, Україна.

АНОТАЦІЯ

Актуальність. Розглянуто задачу автоматизації генерування природніх та антропогенних ландшафтів. Предметом дослідження є процедурні методи генерації ландшафту, що швидко та реалістично візуалізують об’єкти з врахуванням різних рівнів деталізації. Мета роботи – підвищення якості відображення та ефективності процесу генерації ландшафту поверхонь при будь-якому рівні деталізації.

Метод. Запропонований метод візуалізації передбачає побудову природнього ландшафту з допомогою кривих та поверхонь Без’є та ручне редагування окремих сегментів; використання програмних агентів, які відповідають за окремі кроки генерації антропогенних об’єктів; адаптацію антропогенних об’єктів до характеристик природніх ландшафтів; контейнеризацію тривимірних об’єктів, що використовується на різних кроках для ефективної організації збереження та завантаження об’єктів. Для побудови поверхонь на окремих сегментах природнього ландшафту використовується згенерована карту висот на основі алгоритму шуму Перлина. Програмні агенти для обробки ландшафту застосовуються для уніфікації побудови алгоритмів створення та обробки інформації про антропогенні об’єкти. Завдяки успадкуванню конкретного інтерфейсу усіма реалізаціями агентів, гарантується коректна робота застосунку та стійкість до помилок. Ефективність деталізації відображення забезпечується контейнеризацією з дворівневим кешуванням.

Результати. Розроблений метод реалізовано програмно і досліджено його ефективність для різних варіантів вхідних даних, що у найбільшій мірі визначають складність об’єктів візуалізації.

Висновки. Проведені експерименти підтвердили працездатність запропонованого алгоритмічного забезпечення і дозволяють рекомендувати його для використання на практиці при вирішенні задач автоматизованої генерації ландшафтів. Перспективи подальших досліджень можуть полягати у вдосконаленні та розширенні алгоритмів процедурного генерування ландшафту, функціоналу ручного опрацювання, розділенні окремих об’єктів на ієрархії контейнерів.

КЛЮЧОВІ СЛОВА: візуалізація об’єктів, сегмент, рівень деталізації, криві Без’є, програмний агент, контейнеризація.

УДК 004.925

АЛГОРИТМЫ И АРХИТЕКТУРА ПРОГРАММНОЙ СИСТЕМЫ АВТОМАТИЗИРОВАННОЙ ГЕНЕРАЦИИ ПРИРОДНЫХ И АНТРОПОГЕННЫХ ЛАНДШАФТОВ

Левус Е. В. – канд. техн. наук, доцент кафедры программного обеспечения Национального университета «Львовская политехника», Львов, Украина.

DOI 10.15588/1607-3274-2022-2-15

Морозов М. Ю. – магистрант кафедры информатики Мюнхенского технического университета, Мюнхен, Германия.

Моравский Р. О. – магистрант кафедры программного обеспечения Национального университета «Львовская политехника», Львов, Украина.

Пустельнык П. Я. – магистрант кафедры программного обеспечения Национального университета «Львовская политехника», Львов, Украина.

АННОТАЦИЯ

Актуальность. Рассмотрена задача автоматизации генерирования природных и антропогенных ландшафтов. Предметом исследования являются процедурные методы генерации ландшафта, быстро и реалистично визуализирующих объекты с учетом разных уровней детализации. Цель работы – повышение качества отражения и эффективности процесса генерации ландшафта поверхностей при любом уровне детализации.

Метод. Предложенный метод визуализации предполагает построение природного ландшафта с помощью кривых и поверхностей Безье и ручное редактирование отдельных сегментов; использование программных агентов, отвечающих за отдельные шаги по генерации антропогенных объектов; адаптацию антропогенных объектов к характеристикам природных ландшафтов; контейнеризацию трехмерных объектов, которая используется на разных шагах для эффективной организации хранения и загрузки объектов. Для построения поверхностей на отдельных сегментах природного ландшафта используется сгенерированная карта высот на основе алгоритма шума Перлина. Программные агенты для обработки ландшафта применяются для унификации построения алгоритмов создания и обработки информации об антропогенных объектах. Благодаря наследованию конкретного интерфейса всеми реализациями агентов гарантируется корректная работа приложения и устойчивость к ошибкам. Эффективность детализации отображения обеспечивается контейнеризацией с двухуровневым кэшированием.

Результаты. Разработанный метод реализован программно, исследовано его эффективность для различных вариантов входных данных, в наибольшей степени определяющих сложность объектов визуализации.

Выводы. Проведенные эксперименты подтвердили работоспособность предлагаемого алгоритмического обеспечения и позволяют рекомендовать его для использования на практике при решении задач автоматизированной генерации ландшафтов. Перспективы дальнейших исследований могут заключаться в совершенствовании и расширении алгоритмов процедурного генерирования ландшафта, функционала ручной проработки, разделении отдельных объектов на иерархии контейнеров.

КЛЮЧЕВЫЕ СЛОВА: визуализация объектов, сегмент, уровень детализации, кривые Безье, программный агент, контейнеризация.

ЛІТЕРАТУРА / ЛИТЕРАТУРА

1. Doran J. Controlled Procedural Terrain Generation Using Software Agents / J. Doran, I. Parberry // IEEE Transactions on Computational Intelligence and AI in Games. – 2010. – Vol. 2 (2). – P. 111–119. DOI: 10.1109/TCIAIG.2010.2049020
2. Heneruvania landshaftiv dlia cferychnykh poverkhon: analiz zavdannia ta variant vyrishennia / M. Yu. Morozov, Ye. V. Levus, R. O. Moravskyj et al.] // Naukovui visnyk NLTU Ukraiiny. – 2020. – Vol. 30, № 1. – P. 136–141. DOI: 10.36930/40300124
3. Biljecki F. CityGML Application Domain Extension (ADE): overview of developments / F. Biljecki, K. Kumar, C. Nagel // In: Open Geospatial Data, Software and Standards 3.1 (Dec. 2018). – P. 13. DOI: 10.1186/s40965- 018 - 0055 - 6
4. Gavin S. P. Muller. The definition and rendering of terrain maps / Gavin S. P. Muller // ACM SIGGRAPH Computer Graphics – 1986. – Vol. 20, № 4. – P. 39–48. DOI:10.1145/15886.15890
5. Ebner H. Generation, Management and Utilization of High Fidelity Digital Terrain Models / H. Ebner, W. Reinhardt, and R. Hobler // XVIth ISPRS Congress Technical Commission III: Mathematical Analysis of Data Supplements. – 1988, Kyoto, Japan. – 27(B11). – P. 556–566.
6. 3D Geovisualization & Stylization to Manage Comprehensive and Participative Local Urban Plans / [M. Brasebin, S. Christophe, F. Jacquinod et al.] // In: ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences IV–2/W1. – October (2016). – P. 83–91. DOI: 10.5194/isprs-annals- IV-2- W1-83-2016.
7. Real-Time HazMat Environmental Information System: A micro-service based architecture / [G. Cherradi, A. El Bouziri, A. Boulmakoul, and K. Zeitouni] // In: Procedia Computer Science. – 2017. – 109.June – P. 982–987. DOI: 10.1016/j.procs.2017.05.457
8. On the impact of the medium in the effectiveness of 3D software visualizations / [L. Merino, J. Fuchs, M. Blumenschein et al.] // IEEE Working Conference on Software Visualization (VISSOFT). – Shanghai, China, 2017. – P. 11–21. DOI: 10.1109/VISSOFT.2017.17
9. Teng E. A semantic approach to patch-based procedural generation of urban road networks / Edward Teng Rafael Bidarra // FDG '17: 12th International Conference on the Foundations of Digital Games, August 2017 : proceedings – P. 1–10.
10. Procedural City Modeling for AR Applications/ [M. Burch, G. Wallner, S. T. T. Arends and P. Beri] // 24th International Conference Information Visualizations (IV). – 2020. – P. 581–586. DOI: 10.1109/IV51561.2020.00098
11. Template-Based 3D Road Modeling for Generating Large-Scale Virtual Road Network Environment / [X. Zhang, M. Zhong, S. Liu et al.] // ISPRS International Journal of Geo-Information. – 2019. – 8(9):364. DOI: 10.3390/ijgi8090364
12. Kelly G. Citygen: An interactive system for procedural city generation / G. Kelly and H. McCabe // In Fifth International Conference on Game Design and Technology. – 2007. – P. 8–16.
13. Dinkov D. 3D modelling and visualization for landscape simulation / D. Dinkov, R. Vatseva // Proceedings of the 6th International Conference on Cartography and GIS. – Albena. – 2016. – P. 320–333.
14. Klimke J. Service-oriented Visualization of Virtual 3D City Models / J. Klimke, J Döllner // Directionsmag.com. [Electronic resource]. – Access mode: 2012. URL: https://www.directionsmag.com/article/1873
15. Object Container Streaming // Star Citizen. 2021. URL: https://starcitizen.tools/Object_Container_Streaming
16. Bushnaief J. Solving Visibility and Streaming in The Witcher 3: Wild Hunt with Umbra 3 / J. Bushnaief, P. Czatrowski // GDC Vault. 2014. URL: www.gdcvault.com (дата звернення: 22.2.2021).
17. Level Streaming Overview // Unreal Engine. 2021. URL: https://docs.unrealengine.com/en-US/BuildingWorlds/LevelStreaming/Overview/index.html.

DOI 10.15588/1607-3274-2022-2-15