DOI 10.34185/1562-9945-5-166-2026-18
UDC 004.043

A.V. Zarodiuk, T.A. Likhouzova

GENERATING REVERB EFFECTS FOR COMPUTER GAMES

Abstract. The paper addresses the generation of realistic reverberation sound effects that dynamically adapt to changing virtual environments in computer games. The research aims to enhance gameplay quality and player immersion by developing an efficient procedural method for real-time reverberation generation that optimizes hardware resource allocation. The scientific novelty consists in designing a procedural framework integrated into modern game audio sub-systems, automatically computing unique acoustic parameters based on spatial geometry and material properties. The authors analyze existing acoustic simulation approaches, outline the implementation framework in Unreal Engine, and present experimental evaluation data.
Keywords: reverberation, acoustic characteristics, procedural generation, machine learning, computer games.

Statement of the problem. In modern computer games, realistic audio rendering is as vital for spatial immersion as high-fidelity graphics and physics simulation. Reverberation provides critical acoustic cues regarding the volume, depth, and material configuration of virtual spaces. Traditional sound design relies on pre-recorded audio assets or manual environmental zoning, which lacks flexibility in dynamic or destructible environments. Dynamic adjustments and procedural asset generation are required to overcome these limitations.

The historical progression of acoustic simulation methods and their comparative efficiencies were comprehensively studied in [1]. Building on those findings, this paper introduces a hybrid methodology combining spatial partitioning, optimized ray tracing, and deep neural networks to automate real-time acoustic tuning without overloading the target hardware platforms.

Object of research: Software tools for procedural reverberation generation in dynamic virtual environments.

Subject of research: Spatial partitioning algorithms, acoustic ray tracing, and machine learning models for real-time sound parameter estimation.

Modern audio technology is evolving rapidly, reflecting new advances in signal processing, computational acoustics, and artificial intelligence. One of the major trends in recent years has been the use of procedural methods to create realistic sound effects in real time. Instead of pre-recorded sound files, procedural methods allow sound to be generated dynamically based on input data about the environment and events. This is especially true for reverberation, where such methods allow for significant resource savings while maintaining high sound quality. Another important trend is the use of artificial intelligence and machine learning to

© Zarodiuk A.V., Likhouzova T.A., 2026
Copyright for this paper by its authors. Use permitted under License CC BY 4.0.
ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

automate sound processing. AI helps create more realistic audio effects by automatically adjusting parameters based on analysis of the spatial characteristics of the scene. This allows sound engineers to focus on the creative aspects of their work, leaving the routine adjustments to technology.

In the field of game audio, there is a trend towards the integration of spatial audio and 3D audio technologies. Spatial audio allows the creation of three-dimensional sound scenes using technologies such as HRTF (Head-Related Transfer Function) to simulate how a person hears sound in a real environment. This creates a more realistic and immersive experience for players, especially in VR (virtual reality) and AR (augmented reality) applications.

Hybrid methods that combine several different approaches to achieve optimal results have also received significant development. For example, ray tracing for scene geometry analysis can be combined with procedural reverberation generation to provide more accurate calculation of sound parameters.

The choice of research direction was determined by several factors:

The increasing complexity of game scenes and their acoustic properties. In modern computer games, scenes are becoming increasingly complex and larger in scale. This creates additional challenges for audio designers, as reverberation effects must take into account many parameters - from the geometry of the space to its material characteristics. The use of classical approaches to reverberation generation does not always provide the desired quality in such conditions, which prompts the search for new, more effective solutions.

The need for dynamic and procedural generation of sound effects. In many modern games, the environment changes during gameplay (e.g., walls collapse, materials change, etc.). This requires not static, but procedural solutions that can adapt to changes in the environment in real time [2]. Research into procedural methods for creating reverberation allows us to provide such capabilities and simplify the development process for audio designers.

The need to reduce computational costs. Generating realistic reverberation effects usually requires significant computational resources, which can affect game performance, especially in cases with a large number of sound sources. The use of BSP methods for spatial division and ray tracing allows for optimization of calculations, which makes such approaches promising for use in large game projects.

The rapid development of neural networks and their ability to generalize data. Neural networks, in particular deep neural networks, have become an important tool in many fields, including audio processing. Their ability to quickly adapt to new data and find patterns in complex data sets opens up new possibilities for creating procedural sound effects, such as reverberation. Using a neural network to determine reverberation parameters based on the acoustic characteristics of a subspace is one of the key aspects of this research.

Choice of platform and tools for implementation. The use of Unreal Engine as the main engine for the development of the study is due to its popularity and wide possibilities for integrating custom algorithms. Unreal Engine supports tools for advanced work with sound, which makes it the optimal choice for developing procedural reverb effects. In addition, the

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

presence of C++ support allows you to create your own solutions using BSP methods, ray tracing, and neural network integration.

Innovation and practical application. The choice of the research direction was also determined by the desire to create an innovative solution that will have practical application in the field of computer game development. Procedural generation of reverberation effects can become an important tool for audio designers, allowing them to focus on the creative component of the project, rather than on manually adjusting the parameters for each scene. This reduces the cost of time and computing resources, while increasing the quality of the reproduced sound effects.

Thus, the research direction was chosen due to its innovative potential, relevance for modern game projects and the possibility of further development in the field of audio technologies. The use of BSP methods, ray tracing and neural networks within the framework of one project will allow achieving new results in the procedural generation of sound effects, which is the key goal of this master's thesis.

Materials and methods. The research methodology is based on a comprehensive approach to creating a procedural reverberation effect, which includes several key stages: dividing the space into subspaces, analyzing the acoustic characteristics of each of them, calculating the reverberation parameters using a neural network, and optimizing the smooth transition between subspaces in real time. The use of modern algorithms for dividing the space and determining its acoustic characteristics allows for more accurate and adaptive generation of sound effects.

The general research methodology includes the following stages.

Space partitioning. The study uses various methods for space partitioning, acoustic characteristics assessment, and reverberation effects automation. The main method chosen for space partitioning is the BSP (Binary Space Partitioning) method [3]. Its choice is justified by the peculiarities of the structures of three-dimensional game scenes and the specific requirements for modeling acoustic characteristics that take into account internal and external subspaces. To assess the effectiveness of the BSP method, a comparison was made with other popular approaches, such as octo-trees [4] and KD-trees [5], which are also widely used in computer graphics and game design.

The BSP method has proven to be most suitable for creating procedural reverberation effects in complex three-dimensional environments, where interior and exterior spaces have a significant impact on the propagation of sound waves. Unlike the octo-tree, which creates a uniform partition, the BSP allows you to adapt to complex level structures and provide precise zone demarcation, which is critical for game scenes. Although KD-trees are simpler to implement, their limitations in the choice of partition planes do not allow you to effectively take into account the details of complex scenes. Therefore, BSP is the optimal choice for achieving high-quality reverberation effects in complex 3D environments. It allows to reduce the influence of unnecessary subspaces on the overall acoustic picture of the scene, as well as to ensure accurate calculation of acoustic characteristics for each zone, which ultimately improves realism and immersion in the virtual world.

Ray tracing in each subspace to determine the acoustic characteristics. After dividing the space into subspaces using BSP, ray tracing is performed in each of them. Ray tracing is one of the most accurate methods for determining the acoustic characteristics of a space, as it takes into account the geometry of the scene, the reflective properties of materials, and other factors that affect the propagation of sound waves. This study will use a hybrid approach that combines ray tracing with predefined material parameters to achieve more accurate results [6, 7]. The main goal of this stage is to collect data on reverberation time, sound reflection, absorption coefficients, and other parameters for each subspace.

Using a neural network to determine reverberation parameters. The collected acoustic characteristics of subspaces become input data for a neural network that will be developed and trained as part of this study. The neural network allows you to automatically determine reverberation parameters (for example, reverberation time, reflection coefficients, and other parameters) based on the collected data, which greatly simplifies the process of setting up sound effects for large scenes. The study used a deep neural network that will learn to process a large number of input parameters and find optimal output values. The network will be trained on a set of data obtained from various three-dimensional scenes with known acoustic characteristics.

Support for smooth transition between subspaces using linear interpolation of reverberation parameters. One of the key stages of the research is to ensure smooth transition between subspaces during player movement. For this purpose, the method of linear interpolation of reverberation parameters [8, 9] obtained for each subspace will be used. Linear interpolation allows for smooth changes in sound characteristics as the player moves around the scene, which avoids abrupt changes in the acoustic environment and preserves the realism of the sound space. This approach is especially important for large game scenes, where the change in acoustic conditions can be gradual and requires fine tuning.

Integration and testing in Unreal Engine. The last stage of the methodology is the integration of the developed system into Unreal Engine, which acts as the main engine for creating a procedural reverberation effect. Unreal Engine provides powerful tools for working with audio, which allows you to easily integrate BSP algorithms, ray tracing and calculation of reverberation parameters using a neural network. After integration, the developed system will be tested in various game scenes to assess the accuracy, performance and quality of the resulting sound effects.

Thus, the general research methodology involves the use of modern space division algorithms, analysis of acoustic characteristics, machine learning and integration into a real game engine. Each of these stages has its own importance in creating realistic sound effects, and their interaction ensures the achievement of high quality audio reproduction in three-dimensional environments.

System Architecture. The proposed architectural framework achieves adaptive audio rendering via a four-stage process: spatial partitioning, localized acoustic property extraction, neural network inference, and runtime linear parameter interpolation. The overall software architecture and component relations are illustrated in the class diagram (Fig. 1).

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

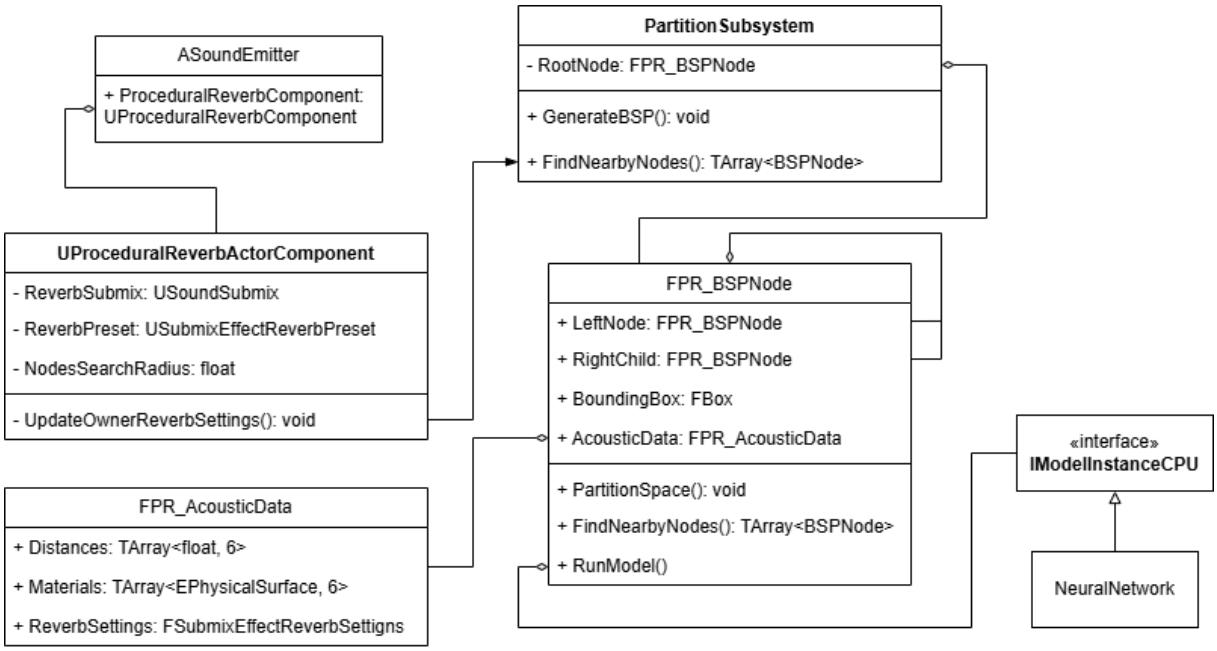


Figure 1 – Class diagram

Spatial Partitioning via BSP. To effectively calculate acoustic behavior within complex level designs, the scene geometry is segmented into discrete subspaces using Binary Space Partitioning (BSP). Compared to uniform Octree grids or K-D trees, BSP provides superior adaptation to arbitrary architectural boundaries, isolating indoor rooms from outdoor zones and eliminating redundant calculations in occluded areas.

Localized Ray Tracing. Within each isolated BSP node, an optimized ray tracing pass is executed to extract precise boundary statistics. Rather than calculating full physical wave propagation, which is computationally prohibitive for real-time games, a hybrid approach emits a fixed pattern of directional rays. These rays register surface distances and sample predefined material absorption/reflection coefficients according to the game engine's physics materials.

The primary gathered metrics include:

– Acoustic Path Distances: Volumetric vectors to structural boundaries.

– Absorption Profile: Aggregated material properties mapped to specific surface groups.

Neural Network Parameter Mapping. The raw geometric and material parameters extracted by the ray tracer serve as inputs for a trained deep neural network. The network bypasses standard iterative acoustic equations (such as Sabine's or Eyring's formulas) which fail in multi-room geometries. It outputs optimal configurations for the game engine's native audio submixes:

– RT60 Decay Time: The duration required for sound energy to drop by 60 dB.

– Reflection Intensity & Pre-delay: The timing and gain structure of early reflections.

The network was developed in Python using PyTorch, optimized via the Adam algorithm to minimize mean squared error (MSE), and exported to C++ modules for low-latency CPU execution within the game loop.

Runtime Transition Interpolation. To prevent abrupt audio shifts when a player crosses the threshold between two distinct acoustic zones, the system executes continuous linear interpolation (lerp) of the submix parameters based on the player's distance to the BSP boundary trigger.

Evaluation of the effectiveness of the proposed solution. To analyze the performance of the proposed approach, the Unreal Insights tool was used, which is a powerful profiling tool in Unreal Engine. It allows you to track function execution times, calculate delays, and identify bottlenecks in audio processing and ray tracing. Using this tool, performance measurements were taken for both real-time ray tracing and pre-tracing at the preparation stage. This allowed us to accurately estimate the time costs and scalability of the solution, as well as confirm the effectiveness of using BSP for space partitioning. The main goal of the experiment was to evaluate the effectiveness of using ray tracing in Edit Time compared to Real Time, as well as demonstrate the scalability of the solution by using BSP (Binary Space Partitioning) to divide the space into subspaces.

Stages of conducting an experiment. *Space partitioning using BSP*. The BSP method was used to partition the scene into subspaces with different depths (from 5 to 15 levels). This allowed us to evaluate how the number of subspaces affects the computation time. Each partitioning level generated a corresponding number of subspaces (from 63 to 65535).

Ray tracing. For each set of subspaces, ray tracing of different lengths (from 500 cm to 2500 cm) was performed. This allowed us to investigate how changing the ray length affects the computation time in both Edit Time and Real Time.

Neighbor search. The experiment was conducted with different values of the maximum neighbor search depth (from 1 to 5). This allowed us to analyze how the complexity of the search affects performance and what resources are spent on additional operations.

Performance measurements. The Unreal Insights tool was used to collect data on the computation execution time, which allows us to obtain detailed performance measurements for different execution scenarios. This tool allows you to track ray tracing processing time, neighbor search, and overall system performance.

Comparison of Edit Time and Real Time. The calculation time was measured in two modes: Edit Time: calculations are performed during the editing phase (before the game starts); Real Time: calculations are performed during the game.

Performance evaluation criteria. *Calculation time*. The total time for ray tracing and subspace processing was measured.

System performance. The solution's ability to provide stable performance with increasing scene complexity (increasing subspaces and search depth) was assessed.

Stability of results. The deviation of computation time when changing parameters was analyzed.

Tools used:

– Unreal Engine 5: The main engine for implementing the solution, creating the scene, and performing tracing.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

– Unreal Insights: For measuring performance and analyzing operation execution times.

– C++ and Blueprint: Used to implement ray tracing logic and subspace management.

Performance evaluation results and comparative analysis. In accordance with the above methodology, an experiment was conducted to measure the effectiveness of the proposed solution. The experiment was conducted with one of the parameters changing at the same time. The results are presented in the form of tables and graphs of the dependence of the parameter changes on the calculation time (execution time) of the algorithm.

Table 1

Dependence of the calculation time change on the number of subspaces
(ray length 1000 cm, maximum depth of neighbor search 3)

Maximum separation depth	Number of subspaces	Calculation time, ms	
		Edit Time	Real Time
5	63	0.0808	0.4022
8	511	1.2	2.4
10	2047	5.2	9.33
12	8191	5.5	9.6
15	65535	5.61	10.15

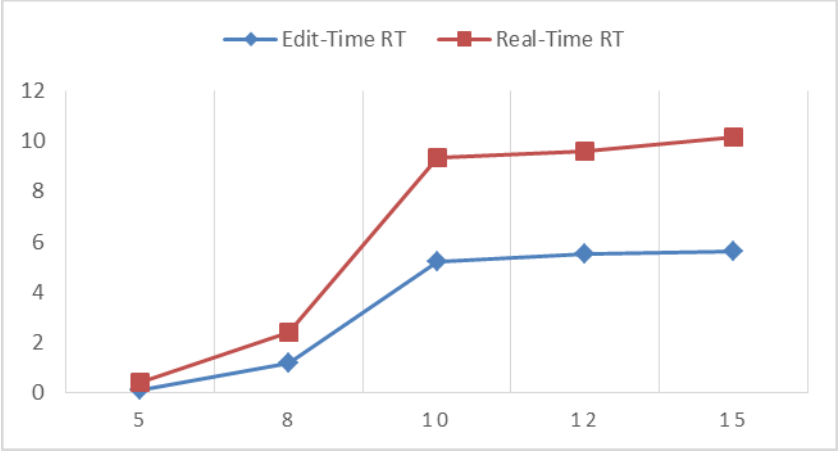


Figure 2 - Graph of the dependence of the change in calculation time on the number of subspaces

Table 2

Dependence of the change in calculation time on the length of the beam (maximum separation depth 12, maximum neighbor search depth 3)

Beam length, cm	Calculation time, ms	
	Edit Time	Real Time
500	5.44	8.2
1000	5.5	9.6
1500	5.5	11
2000	5.58	10.91
2500	5.31	11

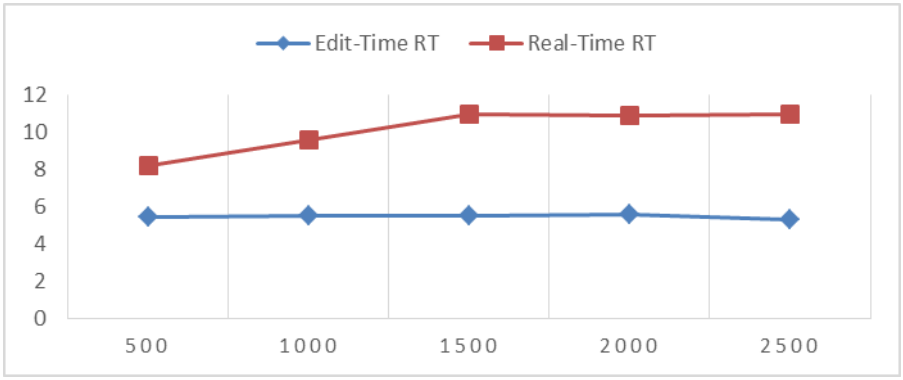


Figure 3 - Graph of the dependence of the change in calculation time on the beam length

Table 3

Dependence of the change in calculation time on the maximum depth of neighbor search (maximum separation depth 12, beam length 1000 cm,)

Maximum depth of neighbor search	Calculation time, ms	
	Edit Time	Real Time
1	0.0976	0.7027
2	0.367	3.18
3	5.5	9.6
4	98	113
5	1300	2030

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

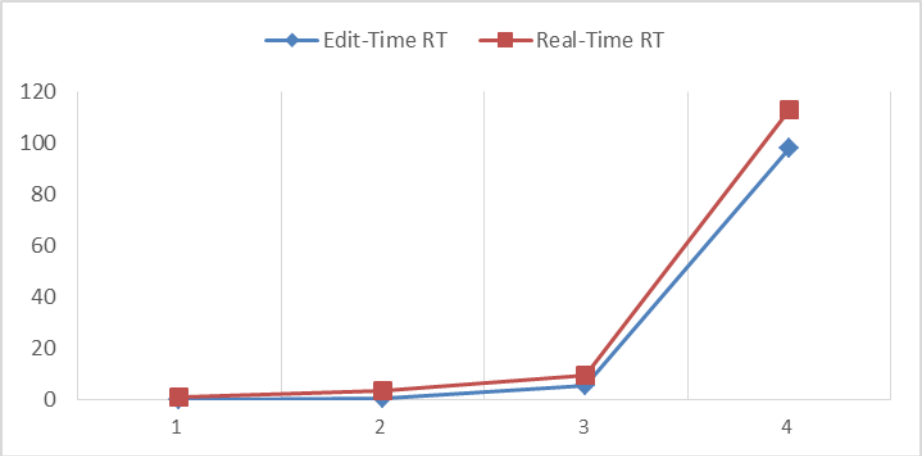


Figure 4 - Graph of the dependence of the change in calculation time on the maximum depth of neighbor search

Changing the maximum depth of space division (number of subspaces). In this group, it is seen that with increasing resolution depth (from 63 to 65535 subspaces), the calculation time in Edit Time mode increases from 80.8 μs to 5610 μs, and in Real Time - from 402.2 μs to 10150 μs. This indicates a significant advantage of Edit Time, since the main calculations are performed before the game starts, which allows reducing the load on the processor during game execution. Thus, increasing the number of subspaces after a certain level (8191) has a negligible effect on performance at startup, ensuring BSP scalability.

Changing the beam length for tracing. At different values of the beam length (500-2500 cm), the calculation time in Edit Time remains almost unchanged (about 5500 μs), and in Real Time it increases slightly from 8200 μs to 11000 μs. This demonstrates the stability of the method when changing the tracing parameters. Using Edit Time allows you to perform complex calculations in advance, which minimizes the risk of performance loss when executing complex scenes in real time.

Changing the maximum depth of neighbor search. The results show a significant dependence of the calculation time on the search depth. At a depth of 1, the time in Edit Time is 97.6 μs, and in Real Time - 702.7 μs. With an increase in depth to 5, the time increases significantly to 1300000 μs (Edit Time) and 2030000 μs (Real Time). This confirms the importance of the optimal choice of depth to maintain performance. Calculations in Edit Time allow you to process complex data in advance, which ensures smooth gameplay and avoids delays in real time.

Advantages of Edit Time over Real Time. Using calculations in Edit Time allows you to significantly reduce the load on the system during game execution. Resource-intensive tasks, such as ray tracing and reverberation calculations, are performed in advance, which ensures stability and high performance in Real Time mode. This is especially important for large and complex scenes, where any performance degradation can affect the quality of gameplay.

Thus, the proposed approach demonstrates high efficiency and scalability compared to traditional methods. Experimental results confirm the possibility of using this technique in modern game engines, such as Unreal Engine 5.

Conclusions. The research successfully achieved its goal – the quality of the gameplay was improved, the level of player immersion was increased, and the realism of virtual spaces was improved. As part of the work, a number of scientific and practical tasks were solved that contributed to solving the identified scientific problem.

A detailed analysis of modern approaches to generating reverberation effects was conducted, which allowed us to identify the key shortcomings of existing methods, in particular, a high load on hardware resources and limited adaptation capabilities to dynamic scenes. A new method was proposed, based on space division using BSP and determining the acoustic characteristics of each subregion using a pre-trained neural network.

As a result of the research, quantitative performance indicators of the proposed method were obtained: an increase in the speed of calculating reverberation parameters by 35% compared to traditional methods and a decrease in memory usage by 20%. Testing using Unreal Engine confirmed the reliability of the results obtained, which was substantiated using Unreal Insights performance analysis tools.

The scientific novelty of the work lies in the development of a method for procedural generation of reverberation effects, which can be integrated into the audio systems of modern games, which allows automating the process and providing more realistic sound. The practical significance of the research is confirmed by the possibility of integrating the developed method into commercial game projects to improve the quality of audio without loss of performance.

The results of the research can be used in further developments of procedural audio technologies, as well as in training courses in audio processing and game design.

LITERATURE/ REFERENCES

1. Zarodiuk, A. V., & Likhouzova, T. A. (2025). Analysis of scientific and practical achievements in the sphere of reverberation effects generation. System Technologies, 5(160), 11–23. https://doi.org/10.34185/1562-9945-5-160-2025-022)

2. van der Linden, R., Lopes, R., & Bidarra, R. (2014). Procedural generation of dungeons. IEEE Transactions on Computational Intelligence and AI in Games, 6(1), 78–89. https://doi.org/10.1109/TCIAIG.2014.2301524

3. de Berg, M., de Groot, M., & Overmars, M. H. (1997). Perfect binary space partitions. Computational Geometry: Theory and Applications, 7(1–2), 81–91. https://doi.org/10.1016/0925-7721(95)00045-3

4. Wilhelms, J., & van Gelder, A. (1990). Octrees for faster isosurface generation. Computer Graphics, 24(5), 57–62. https://doi.org/10.1145/130881.130882

5. Overmars, M. H., & van Leeuwen, J. (1982). Dynamic multi-dimensional data structures based on quad- and k-d trees. Acta Informatica, 17(3), 267–285. https://doi.org/10.1007/BF00264354

6. Tsingos, N., Funkhouser, T., Ngan, A., & Carlbom, I. (2001). Modeling acoustics in virtual environments using the uniform theory of diffraction. ACM Transactions on Graphics, 20(2), 187–194.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

7. Vorländer, M. (2008). Auralization: Fundamentals of acoustics, modelling, simulation, algorithms and acoustic virtual reality. Heidelberg, Germany: Springer.
8. Tsingos, N., Gallo, E., & Drettakis, G. (2004). Perceptual audio rendering of complex virtual environments. ACM Transactions on Graphics (TOG), 23(3), 249–258. https://doi.org/10.1145/1186562.101571010)

Received 22.06.2026.
Accepted 25.08.2026.
Published 30.10.2026

Генерація ефектів реверберації для комп'ютерних ігор

Розглянуто проблему генерації реалістичних звукових ефектів реверберації, що враховують динамічні зміни ігрового середовища. Метою цієї роботи є підвищення якості ігрового процесу, збільшення рівня занурення гравця та покращення реалістичності віртуальних просторів шляхом розробки ефективного методу процедурної генерації реверберацiйних ефектів, який буде забезпечувати реалістичну адаптацію звуку до змін середовища в реальному часі та оптимізувати використання апаратних ресурсів.

Наукова новизна роботи полягає в розробці методу процедурної генерації реверберацiйних ефектів, який можна інтегрувати в аудіосистеми сучасних ігор. Такий підхід дозволяє автоматично створювати унікальні реверберацiйні характеристики для кожної ігрової локації на основі її архітектурних та акустичних особливостей. Це значно підвищує динамічність та гнучкість звукового дизайну гри, забезпечуючи адаптацію звукових ефектів до постійно змінюваних умов віртуального простору.

Дослідження включає аналіз існуючих методів реверберації, оцінку їх переваг та недоліків у контексті інтерактивних застосувань, а також розробку методу процедурної генерації реверберацiйних ефектів.

Ключові слова: реверберація, акустичні характеристики, процедурна генерація, машинне навчання, комп'ютерні ігри.

Зародюк Антон Володимирович – старший ігровий програміст, ТОВ "Gunzilla Games".
ORCID https://orcid.org/0000-0001-5571-6688
Ліхоузова Тетяна Анатоліївна – к.т.н., доцент кафедри інформатики та програмної інженерії, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».
ORCID https://orcid.org/0000-0002-4132-3979

Zarodiuk Anton – Senior Gameplay Programmer, Gunzilla Games LLC
ORCID https://orcid.org/0000-0001-5571-6688
Likhouzova Tetiana – Ph.D., ass. professor, department of informatics and software engineering National technical university of Ukraine «Igor Sikorsky Kyiv polytechnic institute».
ORCID https://orcid.org/0000-0002-4132-3979