МОДЕЛИРОВАНИЕ ЗВУКА В СИСТЕМАХ ВИРТУАЛЬНОГО ОКРУЖЕНИЯ

И. П. Саблин[1,а], **М. В. Михайлюк**[1,б], **Д. В. Омельченко**[1,в], **Д. А. Кононов**[2,г]

[1] *Федеральное государственное автономное учреждение «Федеральный научный центр Научно-исследовательский институт системных исследований Национального исследовательского центра «Курчатовский институт», г. Москва, Российская Федерация*

[2] *Федеральное государственное бюджетное учреждение науки Институт проблем управления им. В. А. Трапезникова Российской академии наук, г. Москва, Российская Федерация*

[а] ✍ *sablinivan97@gmail.com*

[б] *ORCID: http://orcid.org/0000-0002-7793-080X, mix@niisi.ras.ru*

[в] *omelchenko_dv@mail.ru*

[г] *ORCID: http://orcid.org/0000-0002-6059-5590, dmitrykon52@gmail.com*

Аннотация: в работе представлена технология и методы реализации подсистемы звука в системах виртуального окружения. Основное внимание уделено вопросам внедрения акустических компонентов в 3D-сцены и разработке программного обеспечения для управления звуком. В качестве решения предлагается метод подготовки объектов типа «Источник звука» в системе трехмерного моделирования 3DS MAX с последующим экспортом в целевую платформу. Для реализации звука в реальном времени разработано программное обеспечение, обеспечивающее динамический расчет позиций источников звука и слушателя. Предложена классификация источников звука на стационарные (гудок, сирена, рупор и т.д.), динамические (работа лифта, холостой ход и звук двигателя робота под нагрузкой и др.) и природные (например, дождь или град различной интенсивности), а также методы определения моментов звукового сопровождения и проигрывания звуковых файлов. Для стационарных источников предложена методика создания виртуальных пультов управления с функциями включения и выключения звука, регулировки громкости и паузы. Апробация предлагаемых методов и подходов была проведена в созданном комплексе виртуального окружения VirSim. Результаты апробации показали адекватность предлагаемых решений и их применимость для моделирования звука в системах виртуального окружения.

Ключевые слова: система виртуального окружения, источник звука, пульт управления, библиотека Miniaudio.

Благодарности: публикация выполнена в рамках государственного задания НИЦ «Курчатовский институт» — НИИСИ по теме № FNEF-2024-0002 «Математическое моделирование многомасштабных динамических процессов и системы виртуального окружения».

Для цитирования: Саблин И. П., Михайлюк М. В., Омельченко Д. В., Кононов Д. А. Моделирование звука в системах виртуального окружения. *Успехи кибернетики*. 2025;6(2):92–99.

Поступила в редакцию: 29.04.2025. *В окончательном варианте: 15.05.2025.*

SOUND MODELING IN VIRTUAL ENVIRONMENT SYSTEMS

I. P. Sablin[1,a], **M. V. Mikhailyuk**[1,b], **D. V. Omelchenko**[1,c], **D. A. Kononov**[2,d]

[1] *Scientific Research Institute for System Analysis of the National Research Centre "Kurchatov Institute", Moscow, Russian Federation*

[2] *V. A. Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russian Federation*

[a] ✍ *sablinivan97@gmail.com*

[b] *ORCID: http://orcid.org/0000-0002-7793-080X, mix@niisi.ras.ru*

[c] *omelchenko_dv@mail.ru*

[d] *ORCID: http://orcid.org/0000-0002-6059-5590, dmitrykon52@gmail.com*

Abstract: we presented technologies and methods for implementing a sound subsystem within virtual environment systems. We focused on integrating acoustic components into 3D scenes and developing sound management software. As a solution, we proposed a method for preparing Sound Source objects in the 3DS MAX three-dimensional modeling system, followed by export to the target platform. To enable real-time sound implementation, we developed software that dynamically calculates the positions of sound sources and

the listener. We introduced a classification of sound sources into immovable (e.g. horn, siren, megaphone), dynamic (e.g. elevator operation, idling or loaded robot engine sounds), and ambient (e.g. varying rain or hail intensity), and outlined methods for determining the timing of sound accompaniment and playback of sound files. For stationary sources, we proposed a method for creating virtual control panels with functions such as sound activation, volume control, and pause. We tested the proposed methods and approaches within the virtual environment system VirSim. The results confirmed the adequacy of the proposed solutions and their applicability to sound modeling in virtual environments.

Keywords: virtual environment system, sound source, control panel, Miniaudio librar.

Acknowledgements: this is a part of the FNEF-2024-0002 Simulation of Multiscale Dynamic Processes and Virtual Environments government contract granted to the Scientific Research Institute for System Analysis of the National Research Centre “Kurchatov Institute”.

Cite this article: Sablin I. P., Mikhailyuk M. V., Omelchenko D. V., Kononov D. A. Sound Modeling in Virtual Environment Systems. *Russian Journal of Cybernetics*. 2025;6(2):92–99.

Original article submitted: 29.04.2025. *Revision submitted: 15.05.2025*.

Введение

Одним из важных направлений в современных исследованиях является создание и усовершенствование систем виртуального окружения. Данные системы применяются во многих сферах, в том числе при создании тренажеров (медицинских, авиационных, строительных, архитектурных и др.), для создания игр, виртуальных выставок и т.д. Для более глубокого погружения в виртуальную реальность и большей реалистичности в ней необходимо влиять на различные органы чувств человека. Следовательно, систему виртуальной реальности удобно разделить на подсистемы, отвечающие за каждый орган соответственно. В данной статье описывается разработка одной из этих подсистем — подсистемы звука.

К решению данной задачи может быть несколько подходов. В работах [1] и [2] рассматривалась возможность синтеза звука непосредственно во время работы системы виртуальной реальности. Данный метод помогает сэкономить на общем объеме системы, так как при синтезе звука отпадет необходимость в хранении самих аудиофайлов, но, с другой стороны, такой подход увеличит время расчета каждого шага моделирования. Другим подходом может быть разработка подсистемы звука на основе уже существующих библиотек для работы с аудиофайлами. Например, в статье [3] делается обзор аудио плагинов различных разработчиков, использующих игровой движок Unity. В диссертации [4] описывается создание сервера 3D звука для системы виртуальной реальности на основе аудио библиотеки OpenAL. Но библиотеки, использовавшиеся в двух последних работах, имеют ограничения по применению, поэтому в наших разработках используется бесплатная библиотека Miniaudio.

Miniaudio — это кроссплатформенная библиотека пространственного аудио, предназначенная для эффективного рендеринга многоканального позиционного аудио, написанная на языке программирования C. Miniaudio оформлена в виде одного файла с расширением h, содержащего в себе код, а также комментарии к нему. Для установки данной библиотеки необходимо скачать этот файл с аккаунта ее разработчиков в GitHub [5] и подключить к проекту C++, прописав в одном из файлов с расширением cpp следующие строчки:

```
#define MINIAUDIO_IMPLEMENTATION
#include "miniaudio.h"
```

Более подробная документация и примеры использования возможностей данной библиотеки находятся на сайте разработчиков [6].

Система виртуального окружения VirSim

Система VirSim [7] состоит из оболочки, а также подсистем управления, визуализации, динамики и звука (см. рис. 1).

Виртуальная сцена создается в системе трехмерного моделирования 3DS MAX и загружается в подсистемы динамики и визуализации. Оператор управляет с помощью реальных или виртуальных пультов управления динамическими объектами и эффектами, и эти управляющие сигналы передаются через оболочку в подсистемы динамики и визуализации. Подсистема динамики вычисляет новые координаты и ориентации объектов сцены и передает их через оболочку в подсистему визуализации, которая выводит на монитор оператора вид сцены после изменений. Этот цикл занимает не более 40

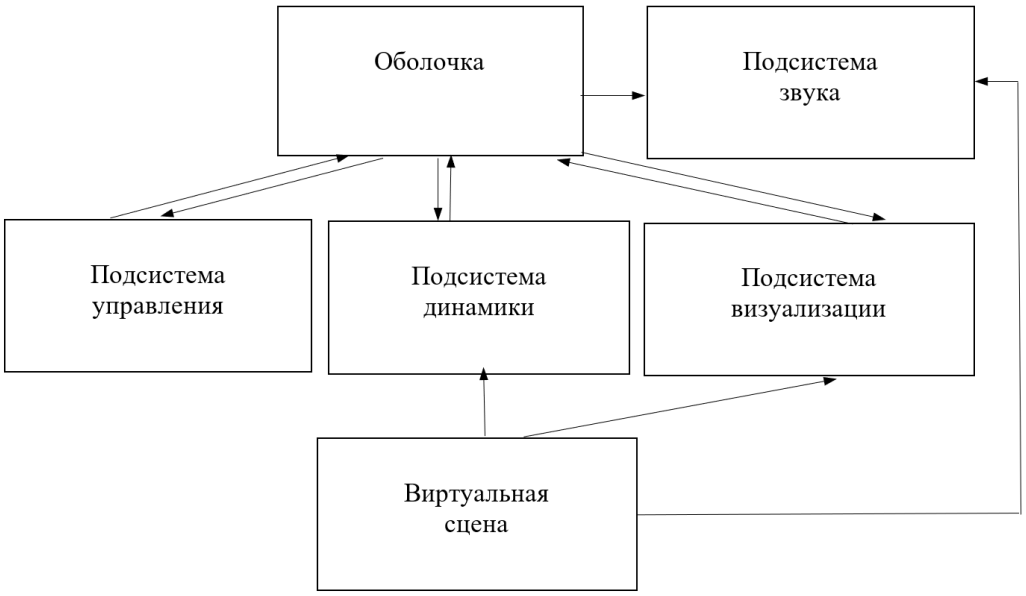

Рис. 1. *Блок-схема системы VirSim*

мсек, т. е. работа системы происходит в масштабе реального времени. Работа подсистемы звука в системе VirSim организована таким же образом. В подсистему звука загружается виртуальная сцена, а необходимые для ее работы данные от других подсистем передаются через оболочку.

Библиотека Miniaudio работы со звуком

Библиотека Miniaudio предоставляет возможность работы с низкоуровневым и высокоуровневым API. Низкоуровневый API дает прямой доступ к аппаратным ресурсам системы для обработки звука. Он позволяет управлять такими процессами, как буферизация данных, частота дискретизации, обработка многоканального звука, а также работа с конкретными устройствами вывода и ввода. Высокоуровневый аудио API предлагает решения для выполнения стандартных задач, таких как воспроизведение звуков, управление эффектами, пространственное позиционирование и так далее.

Для работы со звуком надо инициализировать ряд структур, в частности, «движок» (*engine*), который инициализируется функцией

ma_engine_init(NULL, &engine).

*Параметры источника звука***.** Для инициализации источника звука из звукового файла можно использовать функцию

ma_sound_init_from_file(&engine, имя файла, 0, NULL, NULL, &sound).

Здесь источник звука *sound* является структурой, в которую запишутся параметры файла. Остальные параметры заданы значением *NULL* — это означает, что будут использованы их значения по умолчанию.

Источник звука имеет следующие основные параметры: координаты положения в мировой системе координат (МСК), вектор направления, громкость и звуковые конусы. Положение S источника звука задается функцией

ma_sound_set_position(&sound, S_x, S_y, S_z*),*

а вектор F направления — при помощи функции

ma_sound_set_direction(&sound, F_x, F_y, F_z*).*

Громкость звука изменяется в зависимости от расстояния от источника звука. Miniaudio поддерживает несколько моделей затухания звука в пространстве, в том числе модель Inverse Distance Clamped Model, описанную в документации библиотеки OpenAL [8] для работы с аудио. Для нее операторами

ma_sound_set_min_distance(&sound, $D_{\min}$*)*

ma_sound_set_max_distance(&sound, $D_{\max}$*)*

ma_sound_set_rolloff(&sound, k)

можно задать минимальное ($D_{\min}$) и максимальное ($D_{\max}$) расстояния от источника звука, а также коэффициент k затухания в области между этими границами (см. рис. 2). В области от источника звука до границы $D_{\min}$громкость звука будет совпадать с громкостью V_S в точке S, т. е. коэффициент затухания $g_d = 1$. В любой точке P между этими границами, находящейся на расстоянии d от источника звука, коэффициент затухания громкости вычисляется по формуле:

$$g_d = \frac{D_{\min}}{D_{\min} + k_d(d - D_{\min})},$$

так что громкость в этой точке будет равна:

$$V_P = g_d V_S.$$

После границы $D_{\max}$коэффициент затухания в любой точке равен $g_{D_{\max}}$.

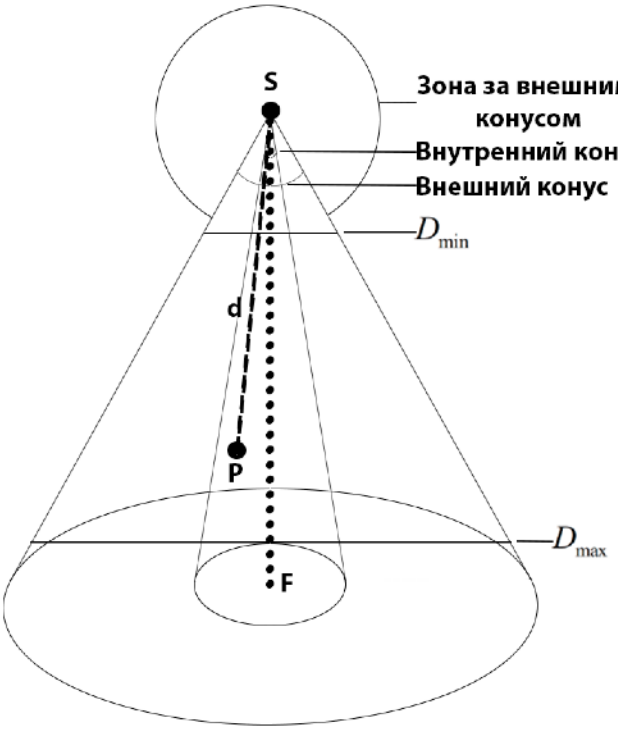

Рис. 2. *Схема затухания звука*

Затухание звука происходит также в зависимости от угла отклонения от направления источника звука. Для этого задаются углы *inAng* внутреннего и *outAng* внешнего звуковых конусов и коэффициент k_a затухания вне внешнего конуса с помощью оператора

ma_sound_set_cone(&sound, inAng, outAng, k_a).

Внутри внутреннего конуса коэффициент затухания $g_a = 1$, т. е. затухания звука по направлению не будет. Вне внешнего конуса $g_a = k_a$, а между конусами g_a будет линейно интерполироваться от 1 до k_a.

Общий коэффициент затухания равен произведению коэффициентов затухания по расстоянию и по углу, т. е.$g = g_a g_d$, а громкость звука в точке P будет вычисляться по формуле

$$V_P = g_a g_d V_S.$$

Параметры слушателя. В общем случае библиотека Miniaudio поддерживает обработку нескольких слушателей, но в нашем случае в сцене может быть только один слушатель (ему присваивается номер 0). Основными параметрами слушателя являются координаты положения в МСК, вектор направления и вектор «верха», а также звуковые конусы. Положение L слушателя задается при помощи функции

ma_engine_listener_set_position(&engine, 0, L_x, L_y, L_z).

Вектор V направления и вектор U «верха» слушателя задаются при помощи операторов

ma_engine_listener_set_direction(&engine, 0, V_x, V_y, V_z*)*

ma_engine_listener_set_world_up(&engine, 0, U_x, U_y, U_z*).*

Как и для источника звука, для слушателя задаются звуковые конусы углами *inAL*, *ouAL* и коэффициент k_L при помощи оператора

ma_engine_listener_set_cone(&engine, 0, inAL, ouAL,k_L*).*

Звук от источника попадает к слушателю по прямой линии (направлению), соединяющей источник и слушателя. Коэффициент затухания звука g_L в зависимости от направления вычисляется по такому же принципу, как и у источника звука. А именно: если направление попадает во внутренний угол слушателя, то $g_L = 1$, между внутренним и внешним углами g_Lлинейно интерполируется от 1 до k_L, а вне внешнего угла $g_L = k_L$. Таким образом, громкость звука в точке L будет вычисляться по формуле:

$$V_L = g_L g_{a,L} g_{d,L} V_S.$$

Реализация звука в системе виртуального окружения VirSim

Создание источника звука. Для создания объекта типа «Источник звука» в системе трехмерного моделирования 3DS MAX разработан новый плагин, который позволяет через диалоговое окно (см. рис. 3) задать параметры источника звука в сцене. Параметры для всех источников звука в сцене экспортируются в бинарный файл с расширением sls, который сохраняется в папке, где находятся остальные конфигурационные файлы сцены.

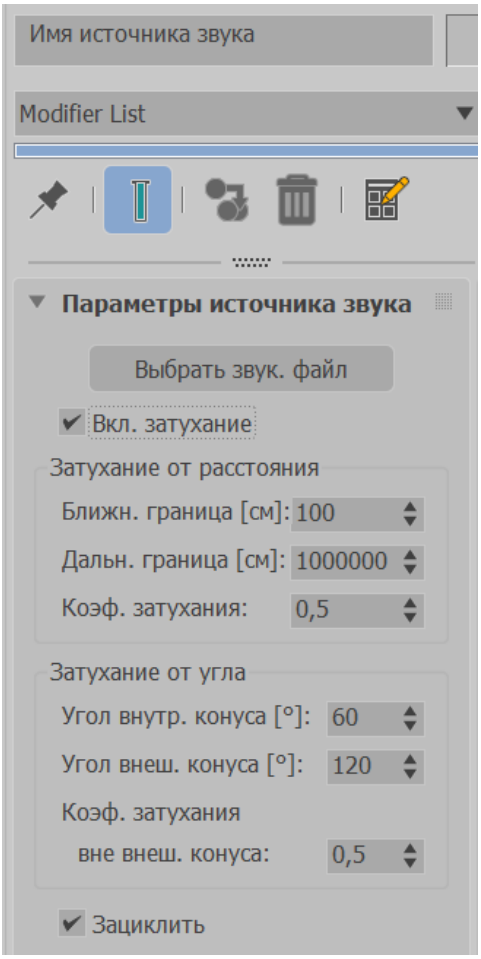

Рис. 3. *Параметры источника звука в 3DS MAX*

Также в данной папке мы будем хранить все заранее подготовленные аудиофайлы, которые будут воспроизводиться в данной сцене. Аналогично создаются и сохраняются в отдельных папках источники звука для каждого робота (звук работающего мотора, поворота манипулятора, вращения гусеницы и др.).

Создание программы обработки аудио. Программа для реализации звука в системе VirSim создана в виде динамической библиотеки (dll), в которую включены (с помощью директивы #include) библиотека Miniaudio.h, файлы Sound.h и Sound.cpp с объявлением и реализацией класса источника звука (в терминах C++), а также файлы SoundLib.h и SoundLib.cpp, содержащие объявления и тела экспортных функций, которые будут вызываться в процессе моделирования. Класс слушателя не создается, т. к. слушатель единственный.

Инициализация источников звука и слушателя. Инициализация значений всех параметров производится один раз в начале работы системы VirSim.

1. Загружаются и сохраняются параметры всех источников звука. Для этого используется экспортная функция *LoadSoundObjects*, в которую передается путь к sls-файлу, а также (для источников звука, расположенных на роботе) номер робота в сцене. В ней также создается список *SndObjectList* объектов класса *SndSource*, в каждый из которых записываются данные из sls-файла, относящиеся к одному источнику звука.

2. Инициализируется «движок» библиотеки Miniaudio, как описано в разделе *Библиотека Miniaudio работы со звуком*, с применением для этого экспортной функции *SoundInitialize*. В этой функции также задаются звуковые конусы слушателя при помощи функции, описной в подразделе *Параметры слушателя*. Эти конусы не будут меняться в процессе работы системы, и для них устанавливаются следующие значения: угол внутреннего конуса — 30°, угол внешнего — 90°, коэффициент затухания за внешним конусом — 0.5.

Затем инициализируется источник звука и задаются его начальные параметры. Для этого создана функция *SetSoundSource*, которая, в свою очередь, циклично вызывается в функции *SoundInitialize*.

Моделирование работы источников звука. При моделировании звука во время работы системы VirSim мы различаем три типа источников звука: стационарные (положение и направление которых не изменяется, например, гудок, сирена, рупор и др., см. рис. 4), динамические (меняющие свои положение и направление) и природные (отличаются от стационарных только тем, что громкость их звука не зависит от затухания по расстоянию и углового затухания). На каждом шаге моделирования для любого стационарного источника звука с помощью его пульта управления можно включить, выключить, поставить на паузу или изменить громкость звука. Пример такого пульта показан на рис. 5. С помощью callback функции *PanelSignalDataHandler* эти данные передаются в оболочку, которая с помощью экспортной функции *CtrlPanelData* передает их в подсистему звука.

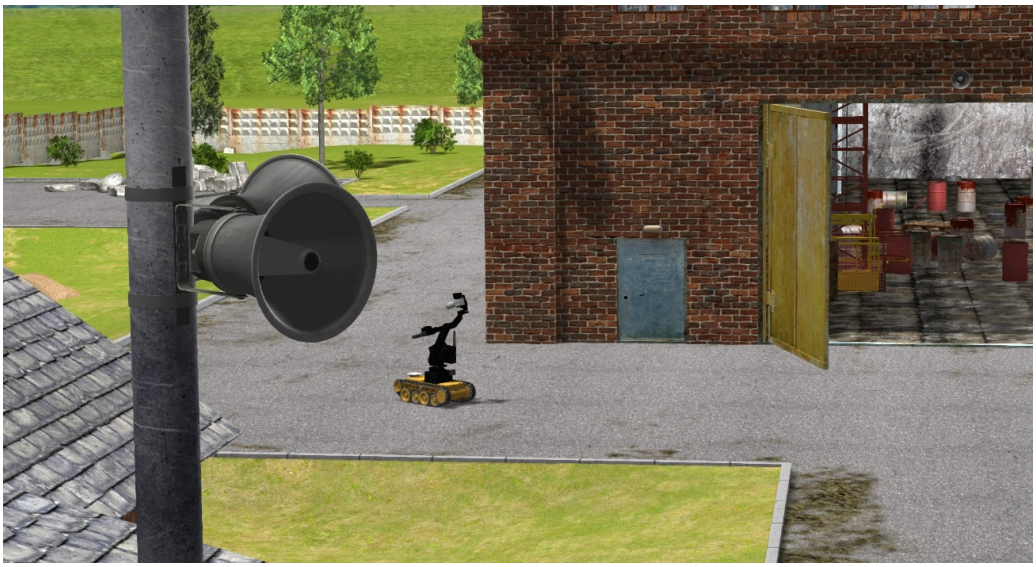

Рис. 4. *Сирена в сцене «Полигон»*

Для воспроизведения звука используются созданные ранее звуковые файлы и следующие функции, которые потребуются для работы со звуковыми файлами:

ma_sound_start(&sound) — воспроизводит звуковой файл;

Рис. 5. *Пульт управления звуком*

ma_sound_stop(&sound) — останавливает воспроизведение звукового файла;

ma_sound_seek_to_pcm_frame(&sound, N) — начинает воспроизведение аудиофайла с кадра *N*;

ma_sound_set_looping(&Sound, isLoop) — зацикливает воспроизведение звукового файла.

Динамическими источниками звука являются источники, меняющие свое положение или ориентацию. Примерами таких источников являются звук работы лифта, звук холостого хода двигателя робота и под нагрузкой. В них громкость зависит не только от расстояния и угла, но также, например, от скорости движения. Для динамических источников звука подсистема динамики с помощью callback функции *DynDataHandler* передает в оболочку имя источника звука, его позицию и направление. Эти данные оболочка передает в подсистему звука с помощью экспортной функции *SndSourceCoords*. В ней работает цикл по объектам списка *SndObjectList*, в котором для каждого изменившего параметры объекта записываются его новые положение и направление.

Примером природного источника звука является дождь. Отличие звука дождя от статических и динамических звуков состоит в том, что его громкость не будет зависеть от каких-либо параметров затухания, т. к. он не локализован в одной точке. Для того чтобы отключить затухания звука в зависимости от расстояния и угла, при инициализации аудиофайла дождя в функции *ma_sound_init_from_file()* в качестве 3-го параметра необходимо передать флаг *MA_SOUND_FLAG_NO_SPATIALIZATION*.

В каждом шаге моделирования необходимо передать в подсистему звука также положение слушателя. В системе VirSim его положение и ориентация совпадают с положением и ориентацией виртуальной камеры, с помощью которой оператор наблюдает виртуальную сцену. Эти данные на каждом шаге моделирования подсистема визуализации передает с помощью callback функции *CameraDataHandler* в оболочку, которая, в свою очередь, при помощи экспортной функции *ListenerCoords* передает их в подсистему звука.

Заключительным шагом моделирования является установка и ориентация динамических объектов в соответствии с новыми данными, а также запуск звуковых файлов в соответствии с их текущими параметрами.

После окончания моделирования вызывается экспортная функция *DestroySoundObjects*, в которой все объекты списка *SndObjectList* очищаются в цикле при помощи функции *ma_sound_uninit(&sound)* и удаляются при помощи оператора delete, а сам список очищается при помощи оператора clear. «Движок» библиотеки Miniaudio, в свою очередь, очищается при помощи функции *ma_engine_uninit(&engine)*.

Заключение

В данной работе рассматривается реализация подсистемы звука в системах виртуального окружения. Предлагается механизм подготовки источников звука для виртуальной сцены, способ вычисления позиции слушателя, определения моментов активизации и проигрывания звуковых файлов с использованием библиотеки Miniaudio. Разработанные методы объединены в рамках подсистемы работы со звуком. Апробация этих методов в системе виртуального окружения VirSim показала их адекватность поставленным задачам.

ЛИТЕРАТУРА

1. Russ M. *Sound Synthesis and Sampling*. Focal Press; 2012. 464 p. DOI: 10.4324/9780080481043.
2. Cornelis Pieter van den Doel. *Sound Synthesis for Virtual Reality and Computer Games*. Ph.D. thesis, University of British Columbia; 1998. 186 p. DOI: 10.14288/1.0051161.
3. Nuora J. *Introduction to Sound Design for Virtual Reality Games: a Look into 3D Sound, Spatializer Plugins and Their Implementation in Unity Game Engine*. Bachelor's thesis, Tampere University of Applied Sciences; 2018. 47 p.
4. Schreier M. *Audio Server for Virtual Reality Applications*. A dissertation submitted in partial fulfilment of the requirements for the degree of Master of Science. Brunel; 2002. 75 p.
5. *Github Miniaudio*. Режим доступа: https://github.com/mackron/miniaudio?tab=readme-ov-file#readme.
6. *Документация Miniaudio*. Режим доступа: https://miniaud.io/docs/.
7. Михайлюк М. В., Мальцев А. В., Тимохин П. Ю., Страшнов Е. В., Крючков Б. И., Усов В. М. Система виртуального окружения VirSim для имитационно-тренажерных комплексов подготовки космонавтов. *Пилотируемые полеты в космос*. 2020;37(4):72-95. DOI: 10.34131/MSF.20.4.72-95.
8. *Документация OpenAL*. Режим доступа: https://openal.org/documentation/.