Т. 12. №7 2026
https://doi.org/10.33619/2414-2948/128

УДК 004.424 https://doi.org/10.33619/2414-2948/128/11

МЕТОД ДИНАМИЧЕСКОГО СИНТЕЗА АУДИО ИЗ БАЙТОВЫХ ПОТОКОВ В СРЕДЕ РАЗРАБОТКИ UNITY

*©**Абляев М. Р.**, ORCID: 0009-0008-2920-9496, Крымский инженерно-педагогический университет им. Ф. Якубова, г. Симферополь, Россия, ablyaev.marlen@gmail.com*

METHOD OF DYNAMIC AUDIO SYNTHESIS FROM BYTE STREAMS IN THE UNITY DEVELOPMENT ENVIRONMENT

*©**Ablyaev M.**, ORCID: 0009-0008-2920-9496, Crimea Crimean Engineering and Pedagogical University named after F. Yakubov, Simferopol, Russia, ablyaev.marlen@gmail.com*

Аннотация. Рассматривается метод динамического синтеза аудио из байтовых потоков в среде Unity, ориентированный на загрузку, преобразование и воспроизведение звуковых данных без обязательного использования заранее подготовленных аудиофайлов. Актуальность исследования обусловлена ростом числа приложений и игровых проектов, в которых требуется гибкая работа со звуком, снижение объема хранимого контента и поддержка кроссплатформенного функционирования. Цель работы заключается в описании последовательности преобразования бинарных данных в аудиосигнал, пригодный для воспроизведения средствами Unity, а также в выявлении технических условий и ограничений применения данного подхода. В рамках исследования последовательно рассмотрены этапы получения байтовых данных из сети, локальной файловой системы или памяти приложения, анализ структуры аудиоформата, преобразование массива byte[] в массив float[] и создание объекта AudioClip для последующего воспроизведения через AudioSource. Особое внимание уделено работе с параметрами аудиоданных, включая частоту дискретизации, число каналов, глубину битности и особенности обработки WAV-заголовка. Дополнительно описаны механизмы сохранения результата в стандартный аудиофайл и возможные варианты потоковой подгрузки данных в процессе проигрывания. Показано, что предложенный метод позволяет повысить гибкость аудиосистемы приложения, сократить требования к предварительному хранению контента и реализовать адаптивное звуковое сопровождение интерактивных сценариев. Вместе с тем установлено, что практическое применение подхода связано с затратами ресурсов на преобразование данных, задержками при потоковой загрузке и необходимостью корректной синхронизации асинхронных операций. Полученные результаты подтверждают, что метод может быть использован в мобильных, сетевых и игровых решениях, где важны динамическая генерация звука и оптимизация использования памяти.

Abstract. Examines a method for dynamically synthesizing audio from byte streams in the Unity environment. This method enables loading, converting, and playing audio data without the need for pre-prepared audio files. The relevance of this research is driven by the growing number of applications and game projects requiring flexible audio processing, reduced storage space, and cross-platform support. The goal of this paper is to describe the process of converting binary data into an audio signal suitable for playback in Unity, as well as to identify the technical requirements and limitations of this approach. This paper sequentially examines the steps involved in obtaining byte data from the network, local file system, or application memory, analyzing the audio format structure, converting a byte[] array to a float[] array, and creating an AudioClip object for subsequent playback via an AudioSource. Particular attention is paid to working with audio data parameters, including

Т. 12. №7 2026
https://doi.org/10.33619/2414-2948/128

sampling frequency, number of channels, bit depth, and WAV header processing. Additionally, mechanisms for saving the result to a standard audio file and possible options for streaming data during playback are described. It is demonstrated that the proposed method improves the flexibility of an application's audio system, reduces the requirements for pre-storing content, and enables adaptive audio for interactive scenarios. However, practical application of the approach is associated with resource costs for data conversion, delays during streaming downloads, and the need for correct synchronization of asynchronous operations. The obtained results confirm that the method can be used in mobile, network, and gaming solutions where dynamic sound generation and memory optimization are important.

Ключевые слова: Unity, байтовые потоки, AudioClip, PCM, звуковые данные.

Keywords: Unity, byte streams, AudioClip, PCM, audio data.

Современные приложения и игровые проекты все чаще используют динамически генерируемые аудиоданные – например, для синтеза речи, генерации звуков эффектов или потокового сопровождения интерактивных сценариев. Традиционные методы предполагают работу с готовыми аудиофайлами, что ограничивает гибкость разработки и требует значительных объемов памяти для хранения контента.

Метод преобразования байтовых потоков в аудио прямо в среде Unity позволяет загружать и воспроизводить звуковую информацию по запросу – из сети, памяти устройств или генерировать ее на лету. Это особенно актуально для кроссплатформенных проектов, где важны оптимизация ресурсов и адаптивность контента.

Метод динамического синтеза аудио из байтовых потоков в среде Unity представляет собой последовательность операций по получению, интерпретации и преобразованию бинарных данных в формат, пригодный для воспроизведения через аудиосистему движка. Данный процесс требует понимания как структуры аудиоданных, так и внутренних механизмов работы Unity с аудио.

1. *Получение байтовых данных*. На первом этапе происходит загрузка исходного набора байтов. Источник данных может варьироваться: удаленный сервер, локальная файловая система устройства, оперативная память (например, данные, полученные от другого модуля программы). В Unity для загрузки данных по сети используется класс UnityWebRequest. При работе с аудиоданными возможно использование как специализированных методов (UnityWebRequestMultimedia.GetAudioClip), так и универсального способа получения «сырых» байтов (https://707.su/cq2J): DownloadHandlerBuffer – позволяет получить массив byte[] без предварительной интерпретации; поддержка асинхронной загрузки, что важно для недопущения блокировки основного потока. Для локальных данных применяются стандартные средства: File.ReadAllBytes(path) – чтение файла целиком; потоковое чтение через FileStream – при работе с большими объемами данных.

2. *Анализ и подготовка структуры аудиоданных*. После получения массива байтов необходимо определить, каким образом интерпретировать эти данные. В отличие от готовых аудиофайлов, здесь отсутствует автоматическое декодирование, поэтому параметры задаются вручную или извлекаются из заголовка (если он присутствует, например, в формате WAV). К ключевым параметрам относятся: Частота дискретизации – определяет количество сэмплов в секунду (например, 44100 Гц или 48000 Гц); Количество каналов – моно (1 канал) или стерео (2 канала); Глубина битности – например, 8-bit, 16-bit, 32-bit; Формат данных – PCM (наиболее распространенный), либо сжатые форматы. Если используется формат WAV, структура файла

Т. 12. №7 2026
https://doi.org/10.33619/2414-2948/128

включает заголовок (header), содержащий метаданные. В этом случае необходимо: пропустить заголовок (обычно 44 байта для PCM WAV); извлечь параметры (частота, каналы, битность); перейти к области данных (data chunk) (https://707.su/QWXB).

3. *Преобразование байтов в сэмплы.* Unity работает с аудиоданными в формате массива чисел с плавающей точкой (float). Поэтому ключевым этапом является преобразование массива byte в массив float. Алгоритм преобразования: разбить массив байтов на пары; преобразовать каждую пару в число (BitConverter.ToInt16); нормализовать значение; записать результат в массив float. При работе со стерео данные чередуются: левый канал, правый канал, левый, правый и т.д. Это необходимо учитывать при заполнении массива.

4. *Создание AudioClip.* После подготовки массива сэмплов создается объект AudioClip. В Unity это делается с помощью метода:

AudioClip.Create (name, lengthSamples, channels, frequency, stream).

Параметры: lengthSamples – общее количество сэмплов; channels – число каналов; frequency – частота дискретизации; stream – режим потоковой загрузки (важно для больших файлов).

После создания необходимо передать данные в клип: используется метод SetData(float[] data, int offsetSamples); данные записываются в буфер аудиоклипа (https://707.su/I6nR).

Если используется потоковый режим (stream = true), можно реализовать callback-функцию, которая будет подгружать данные по мере воспроизведения, что особенно важно для длинных или бесконечных аудиопотоков.

5. *Воспроизведение аудио.* Для воспроизведения создается компонент AudioSource, который является стандартным инструментом Unity для работы со звуком: аудиоклип присваивается свойству audioSource.clip; воспроизведение запускается методом audioSource.Play(). Дополнительно могут настраиваться параметры: громкость (volume); пространственное позиционирование (3D-звук); зацикливание (loop) (https://707.su/GWX6).

Это позволяет интегрировать синтезированный звук в игровую или интерактивную среду.

6. *Сохранение аудиофайла*. При необходимости данные могут быть сохранены обратно в файл. Для этого выполняется обратное преобразование: массив float[] переводится в byte[]; добавляется заголовок (например, WAV header); данные записываются на диск через File.WriteAllBytes().

Формирование WAV-заголовка включает: сигнатуру «RIFF»; указание размера файла; параметры аудио (частота, каналы, битность); блок данных «data» (https://707.su/FYSQ).

Это позволяет сохранить результат в стандартном формате, совместимом с внешними аудиоплеерами.

7. Особенности и ограничения метода. При реализации метода необходимо учитывать ряд технических аспектов: Производительность: преобразование больших массивов данных может создавать нагрузку на CPU; Задержки: при потоковой загрузке возможны задержки воспроизведения; Синхронизация: важно корректно обрабатывать асинхронные операции.

Форматы: Unity напрямую не поддерживает все аудиоформаты без декодирования.

Память: хранение больших массивов float[] требует значительных ресурсов.

Метод динамического синтеза аудио из байтовых потоков предоставляет следующие преимущества:

Гибкость: позволяет создавать или загружать аудиофайлы во время работы приложения, без предварительной подготовки контента.

Оптимизация: уменьшает объем хранимых данных, что важно для мобильных и сетевых решений.

Интерактивность: открывает возможности для адаптивных звуковых систем, реагирующих на действия пользователя или сетевые события.

Кроссплатформенность: Unity обеспечивает единый интерфейс для работы с байтовыми потоками на разных устройствах и операционных системах.

Перспективы дальнейшего развития метода связаны с внедрением алгоритмов сжатия, синтеза речи на основе нейросетей, а также реализацией потокового воспроизведения без промежуточного сохранения данных.

Источники:

(1). Unity Technologies. UnityWebRequest: Unity Manual. https://707.su/cq2J
(2). Unity Technologies. AudioClip: Scripting API. https://707.su/QWXB
(3). Unity Technologies. Audio Clip: Unity Manual. https://707.su/I6nR
(4). Library of Congress. PCM, Pulse Code Modulated Audio. https://707.su/GWX6
(5). Topher Lee. Wav (RIFF) File Format Tutorial. https://707.su/FYSQ

Поступила в редакцию
07.05.2026 г.

Принята к публикации
16.05.2026 г.

Ссылка для цитирования:

Абляев М. Р. Метод динамического синтеза аудио из байтовых потоков в среде разработки Unity // Бюллетень науки и практики. 2026. Т. 12. №7. С. 101-104. https://doi.org/10.33619/2414-2948/128/11

Cite as (APA):

Ablyaev, M. (2026). Method of Dynamic Audio Synthesis from Byte Streams in the Unity Development Environment. *Bulletin of Science and Practice, 12*(7), 101-104. (in Russian). https://doi.org/10.33619/2414-2948/128/11