# План интеграции звуковой подсистемы (Music & SFX)

Цель: расширить архитектурный план ADDON\_STATES\_RESEARCH\_AUTOMATION\_PLAN.md, добавив полноценную звуковую подсистему. Описаны интеграция звуковых эффектов (SFX) и музыки, с привязкой к слоям архитектуры (Presentation, Infrastructure, Application) и учетом специальных требований (off-screen симуляция, событийная модель, сохранение состояния звука).

## Оглавление

1. Обзор звуковой подсистемы
2. Интеграция звуковых эффектов (SFX)  
   2.1. SFX: Архитектура и слои  
   2.2. Аудиомикшер SFX: группы и параметры  
   2.3. Событийная система для SFX (EventBus)  
   2.4. Класс SfxManager (Presentation) – воспроизведение звуков  
   2.5. Импорт и классификация аудио-ассетов (Infrastructure)
3. Интеграция музыкальной подсистемы  
   3.1. Режимы музыки и переходы состояний  
   3.2. Класс MusicDirector и аудио-снапшоты (Presentation)  
   3.3. Управление музыкой из Application-слоя (события и состояния)  
   3.4. Темы, слои и стили музыки (ambient, build-up, stinger)
4. Off-screen симуляция и поведение аудио
5. Сохранение и восстановление состояния аудио
6. Глоссарий терминов
7. Технические требования и рекомендации

## 1. Обзор звуковой подсистемы

Звуковая подсистема состоит из двух основных частей: звуковые эффекты (SFX) и музыкальное сопровождение. Эти компоненты интегрируются в архитектуру игры с соблюдением принципов слоя: логика игры (Application) остается независимой от Unity API, а всё воспроизведение звука сосредоточено в Presentation- и Infrastructure-слоях[[1]](https://github.com/mrsokol2552-source/Game/blob/59b3aa7254ed979296ecd24d2d389cb586acb53a/docs/AUDIO_INTEGRATION.md#L7-L9).

Основные задачи интеграции звука: - Организация аудио-ресурсов (файлы SFX и музыки, аудиомикшеры, группы) и их оптимальный импорт.  
- Реализация менеджеров звука, отвечающих за воспроизведение эффектов и музыки.  
- Привязка звуковых событий к игровым событиям через EventBus (событийную шину) для декуплинга.  
- Обеспечение поддержки различных режимов музыки (спокойный, напряженный, военный) с плавными переходами и ducking (автоприглушением) музыки под важные SFX.  
- Учет особых случаев: отсутствие визуализации (off-screen или фоновая симуляция) и сохранение/восстановление аудиосостояния при паузе или загрузке игры.

Ниже рассмотрены по отдельности интеграция SFX и музыки, а затем общие моменты (off-screen, сохранение состояния).

## 2. Интеграция звуковых эффектов (SFX)

### 2.1. SFX: Архитектура и слои

Presentation (Unity): В слое представления реализуется класс SfxManager (MonoBehaviour), который управляет проигрыванием звуковых эффектов. Он содержит привязки к аудиомикшеру и группам, пулы аудио-источников и методы для воспроизведения SFX разных типов (UI, мир, бой и т.д.). Все Unity-специфичные элементы (AudioSource, AudioMixer) инкапсулированы здесь.

Application (Game Logic): Логика игры генерирует события (например, “пользователь кликнул кнопку”, “юнит выстрелил”, “исследование завершено” и т.п.), не вызывая аудио напрямую. Вместо этого события публикуются в EventBus, на который подписан SfxManager. Application-слой оперирует абстракциями событий, не зная о конкретных звуковых клипах.

Infrastructure (Assets & Data): Этот слой содержит собственно аудио-ассеты (файлы звуков), настройки их импорта, аудиомикшеры с группами и параметры, а также, при необходимости, конфигурационные данные для сопоставления событий и аудиоклипов. Например, JSON/CSV-таблицы идентификаторов звуков и путей в проекте могут использоваться для каталога SFX (опционально)[[2]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L39-L48). Кроме того, Addressables могут применяться для организации загрузки/выгрузки больших банков звуков по меткам (Labels)[[3]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L267-L274).

### 2.2. Аудиомикшер SFX: группы и параметры

Все звуковые эффекты разделены на категории, каждая привязана к своей группе аудиомикшера. Предусмотрены следующие группы микшера для SFX[[4]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L129-L134):

* SFX\_UI – звуки интерфейса (нажатия кнопок, уведомления UI).
* SFX\_World – звуки окружения и мира (двери, механика, окружение, толпа).
* SFX\_Combat – боевые эффекты (выстрелы, взрывы, попадания).
* SFX\_Voice – голосовые и радио эффекты (радио-переговоры, системные голосовые оповещения).
* SFX\_Ambience – фоновые амбиент-звуки (шумы окружения, ветер, дневной/ночной фон и т.д.).

Все эти группы сведены в иерархии микшера под общую шину SFX/Main, которая затем идет на мастер-шину Master[[4]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L129-L134). Такая структура позволяет раздельно регулировать категории SFX и применять эффекты/фильтры к ним группами.

Exposed параметры микшера: для автоматизации микширования SFX вводятся три глобальных параметра, экспонированных в AudioMixer[[5]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L130-L134):

* SFX\_Tension (0..1) – степень напряженности: используется для добавления яркости/резкости звука (например, повышением высоких частот или транзиент-шейпером) при повышении напряжения ситуации[[6]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L132-L134). Может управляться игровыми состояниями (например, плавно увеличиваться при росте опасности).
* SFX\_War (0..1) – “военный режим”: при значении 1 усиливает громкость боевых звуков (~+1–2 dB) и одновременно несколько ослабляет музыку[[6]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L132-L134). Используется для акцента на SFX во время интенсивного боя/войны.
* OcclusionLPF (0..1) – коэффициент окклюзии/Low-Pass фильтра: глобально снижает верхние частоты (LPF) для приглушения звуков, которые находятся за препятствиями или вне экрана (off-screen)[[6]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L132-L134). Повышение этого параметра сопровождается более сильным фильтром, имитируя приглушение звука стенами или удаленностью.

Snapshots (снапшоты): В аудиомикшере настраиваются снапшоты под ключевые состояния игры: Calm, Tense, War. Снапшоты фиксируют уровни групп и значения параметров (в т.ч. вышеописанных) для соответствующего состояния. Переходы между снапшотами происходят плавно (кросс-фейд ~0.25–0.4 с)[[7]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L135-L137), обеспечивая незаметное для игрока изменение аудиобаланса при смене ситуации. Например, снапшот Calm — базовый баланс (музыка на нормальном уровне, SFX нормальные), Tense — слегка повышенная яркость SFX и включен ducking музыки, War — максимальная громкость SFX, музыка приглушена, ducking отключён или минимален (см. ниже про ducking).

Ducking (саунд-даккинг): В спокойном и напряженном режимах вводится автоматическое приглушение музыки при громких SFX с помощью сайдчейна (Music канал "слушает" пики SFX)[[7]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L135-L137). Это означает, например, что взрыв или выстрел немного приглушит фоновую музыку, чтобы эффект был более заметен. Для режима War ducking ослабляется либо вовсе отключается[[7]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L135-L137), так как в тот момент и музыка, и эффекты должны звучать полно для ощущения хаоса. Реализация ducking на уровне микшера: на музыкальной группе компрессор с side-chain от группы SFX; параметрами или снапшотом War этот эффект снижается.

### 2.3. Событийная система для SFX (EventBus)

Звуковая подсистема SFX реагирует на события игрового мира. В архитектуре применяется событийная шина (EventBus) для декуплинга: все важные игровые события публикуются в EventBus, а SfxManager на уровне Presentation подписывается на них[[8]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L141-L149).

Таким образом, логика Application генерирует события вроде OnUiClick, OnResearchCompleted, OnExplosion и т.д., не заботясь о том, как именно будет проигран звук. SfxManager же содержит логику маппинга событий на конкретные звуковые клипы и выполняет воспроизведение.

Пример соответствия событий и эффектов из дизайн-документа SFX[[9]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L144-L152):

* OnUiClick → проиграть один из клипов SFX\_UI\_Click\_\* (случайный вариант).
* OnResearchCompleted → SFX\_UI\_ResearchDone\_\* (звук завершения исследования).
* OnConvoyDepart → SFX\_World\_ConvoyDepart\_\* (звук отправления конвоя).
* OnExplosionSmall → SFX\_Combat\_Explosion\_S\_\* (небольшой взрыв, вариант из набора).
* OnOutbreak → SFX\_World\_InfectionSurge\_\* (звук вспышки инфекции).

Полный список событий и их сопоставленных звуков ведется в аудио-документации и будет поддерживаться в актуальном состоянии[[10]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L144-L153)[[11]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L274-L283). Такая таблица помогает придерживаться консистентности звукового дизайна.

На практике, подписка реализуется либо через прямое подключение к глобальному EventBus (например, в Awake менеджера подписаться на нужные события), либо через внедрение зависимостей (если EventBus абстрагирован). При получении события SfxManager извлекает нужный аудио-клип (или выбирает случайный из группы вариантов) и вызывает соответствующий метод воспроизведения (например, PlayUI для UI-звуков, PlayWorldAt для 3D-звуков в мире и т.д.).

### 2.4. Класс SfxManager (Presentation) – воспроизведение звуков

SfxManager – центральный компонент воспроизведения SFX в Unity. Он существует как Singleton (например, статическое поле SfxManager.I) или как объект на основной сцене. Ниже приведён упрощённый пример реализации SfxManager на C#, демонстрирующий ключевые аспекты (маршрутизация по группам, пространственный звук, циклы и пул источников):

using UnityEngine;  
using UnityEngine.Audio;  
using System.Collections.Generic;  
  
public class SfxManager : MonoBehaviour  
{  
 public static SfxManager I;  
  
 [Header("Routing")]  
 public AudioMixer mixer;  
 public AudioMixerGroup sfxUI, sfxWorld, sfxCombat, sfxVoice, sfxAmbience;  
  
 [Header("Banks")] // Списки клипов для различных событий  
 public List<AudioClip> uiClick;  
 public List<AudioClip> buildDone;  
 public List<AudioClip> rifleBurst;  
 // ... другие банки звуков  
  
 [Range(0f, 0.05f)] public float randPitch = 0.03f;  
 [Range(0f, 1.5f)] public float randVolDb = 1.0f;  
  
 void Awake() { I = this; }  
  
 // Проиграть UI-звук (2D)  
 public void PlayUI(AudioClip clip)  
 {  
 var src = GetOneShotSource(sfxUI);  
 SetupRandom(src);  
 src.PlayOneShot(clip);  
 }  
  
 // Проиграть звук в мире (3D) в позиции pos  
 public void PlayWorldAt(AudioClip clip, Vector3 pos)  
 {  
 var src = GetOneShot3DSource(sfxWorld, pos);  
 SetupRandom(src);  
 src.PlayOneShot(clip);  
 }  
  
 // Проиграть звуковой луп (цикл) в мире, вернуть AudioSource для остановки  
 public AudioSource PlayLoopAt(AudioClip clip, Vector3 pos, AudioMixerGroup grp)  
 {  
 var src = GetPooledSource();  
 src.outputAudioMixerGroup = grp;  
 src.transform.position = pos;  
 src.loop = true;  
 src.spatialBlend = (grp == sfxUI ? 0f : 1f);  
 src.clip = clip;  
 src.Play();  
 return src;  
 }  
  
 // Остановить луп и вернуть AudioSource в пул  
 public void StopLoop(AudioSource src)  
 {  
 if (!src) return;  
 src.Stop();  
 ReturnToPool(src);  
 }  
  
 // --- Вспомогательное: пул источников и рандомизация параметров ---  
  
 private Queue<AudioSource> pool = new Queue<AudioSource>();  
  
 // Получить готовый AudioSource из пула (или создать новый)  
 AudioSource GetPooledSource()  
 {  
 if (pool.Count > 0)  
 return pool.Dequeue();  
 // Создаем новый источник, подвешиваем к SfxManager  
 var go = new GameObject("SFX\_AudioSource");  
 go.transform.parent = transform;  
 var src = go.AddComponent<AudioSource>();  
 src.rolloffMode = AudioRolloffMode.Custom; // используем заданные кривые затухания  
 return src;  
 }  
  
 // Вернуть AudioSource в пул  
 void ReturnToPool(AudioSource s)  
 {  
 s.clip = null;  
 s.loop = false;  
 pool.Enqueue(s);  
 }  
  
 // Для одноразового 2D-звука  
 AudioSource GetOneShotSource(AudioMixerGroup grp)  
 {  
 var s = GetPooledSource();  
 s.outputAudioMixerGroup = grp;  
 s.spatialBlend = 0f;  
 return s;  
 }  
  
 // Для одноразового 3D-звука  
 AudioSource GetOneShot3DSource(AudioMixerGroup grp, Vector3 pos)  
 {  
 var s = GetPooledSource();  
 s.outputAudioMixerGroup = grp;  
 s.transform.position = pos;  
 s.spatialBlend = 1f;  
 return s;  
 }  
  
 // Настроить случайные отклонения громкости/тона для вариативности  
 void SetupRandom(AudioSource s)  
 {  
 s.pitch = 1f + Random.Range(-randPitch, randPitch);  
 float volDb = Random.Range(-randVolDb, randVolDb);  
 s.volume = Mathf.Pow(10f, volDb / 20f);  
 }  
}

Пример: Когда игрок нажимает кнопку UI, игровой код вызывает событие OnUiClick. SfxManager, подписанный на это событие, выбирает случайный клип из списка uiClick и вызывает PlayUI(clip), что создает AudioSource, направляет его в группу SFX\_UI, применяет небольшую случайную вариацию громкости/тона и проигрывает звук[[12]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L187-L195)[[13]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L248-L254). Для звуков в мире (например, взрыв в координатах) используется PlayWorldAt, который устанавливает пространственное положение и включает 3D-режим (spatialBlend=1). Длительные звуки (например, шум работающего генератора или стройки) могут запускаться через PlayLoopAt – звук начинает играть в цикле, и позже, когда событие завершится (генератор выключен), игровой код через EventBus сигнализирует об остановке, и SfxManager.StopLoop прекращает цикл.

Пул аудио-источников: Чтобы не создавать/destroy AudioSource каждый раз, SfxManager использует пул объектов. Метод GetPooledSource() либо берет неактивный AudioSource из очереди, либо создает новый (GameObject с AudioSource)[[14]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L222-L230). После проигрывания лупа, StopLoop возвращает источник в пул для повторного использования[[15]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L230-L238). Это улучшает производительность при большом количестве повторяющихся звуков.

Пространственный звук: В примере, PlayUI использует GetOneShotSource с spatialBlend = 0 (чисто 2D звук, без позиционирования)[[16]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L232-L240). А вот PlayWorldAt берет GetOneShot3DSource, который устанавливает позицию звука в мире и spatialBlend = 1 для полного 3D эффекта[[17]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L239-L247). Кроме того, по умолчанию используется Custom Rolloff кривая затухания (настраивается в проекте) для более точного контроля радиуса слышимости звука[[18]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L224-L230). Минимальная дистанция источника настраивается так, чтобы близкие звуки не были чрезмерно громкими[[19]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L260-L263).

Вариативность звуков: Чтобы избежать «эффекта пулемета» при повторном проигрывании одного и того же звука, в SetupRandom вводится небольшая случайность: питч ±(0.0–0.05) и громкость ±1 dB (в дБ шкале)[[13]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L248-L254). Это соответствует требованию иметь вариативность ~±3–5% по тону и ~1.5 dB по уровню между разными воспроизведениями[[20]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L12-L15).

### 2.5. Импорт и классификация аудио-ассетов (Infrastructure)

Структура файлов и нейминг: В проекте Unity все звуковые файлы организованы по папкам и названы согласно конвенции[[21]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L22-L31)[[22]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L36-L44):

Assets/Audio/  
 Mixers/ # аудиомикшеры (.mixer) и снапшоты (Calm/Tense/War)  
 Music/ # музыкальные файлы  
 SFX/  
 UI/ # звуки интерфейса  
 World/ # окружение, механика, толпа  
 Combat/ # оружие, взрывы, боевые  
 Voice/ # голосовые оповещения, радио  
 Ambience/ # фоновые амбиенты (день/ночь, шумы)  
 Addressables/ # (опционально) каталоги для адресной загрузки

Файлы SFX именуются шаблоном SFX\_<Category>\_<Event>\_<Var##>.wav[[22]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L36-L44). Например: SFX\_UI\_Click\_V03.wav – третий вариант звука клика UI, SFX\_Combat\_Explosion\_M\_V02.wav – второй вариант среднего взрыва. Наличие нескольких вариаций (Var01–Var08) на одно событие соответствует требованию 5–8 вариантов для вариативности[[20]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L12-L15).

Формат и импорт: Мастер-файлы хранятся в несжатом качестве 48 kHz / 24-bit WAV[[23]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L9-L16). При импорте в Unity для каждого звука выставляются настройки в зависимости от его типа (короткий одноразовый, средний цикл, длинный амбиент):

* Короткие One-shot эффекты (UI, короткие звуки до ~1–2 с): Load Type = Decompress on Load, формат сжатия PCM (без потерь) либо Vorbis ~0.7 качества[[24]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L16-L18)[[5]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L130-L134). Такие клипы загружаются сразу в память и быстро проигрываются без задержек. Рекомендуется Force To Mono = On для эффектов, где стерео не критично (например, одиночные клики)[[25]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L72-L75).
* Средние по длине лупы (шаги, механические шумы ~1–2 с): Load Type = Compressed In Memory, сжатие Vorbis ~0.6–0.7[[24]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L16-L18)[[26]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L128-L136). Они занимают меньше памяти, но сразу доступны для воспроизведения, подходят для часто повторяющихся фоновых циклов.
* Длительные амбиенты (фоновые шумы 6–10 с и более): Load Type = Streaming, сжатие Vorbis ~0.6[[24]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L16-L18). При потоковом воспроизведении большие файлы не занимают много RAM, данные читаются с диска по мере проигрывания. Важно включить опцию Loop для зацикливания длительных амбиентов[[27]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L83-L87).

Эти настройки могут применяться пакетом через скрипт или пресет импортёра. В документации приведён пример Editor-скрипта, который автоматически назначает правильные параметры импортируемым файлам на основе их папки (UI, Ambience и т.п.)[[26]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L128-L136)[[28]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L110-L118).

Addressables (адресная загрузка): При увеличении количества контента предусмотрено использование Unity Addressables для управления памятью. Аудиофайлы могут быть помечены метками по категориям (например, SFX\_UI, SFX\_World и т.д.)[[29]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L267-L275), и при необходимости целые группы могут выгружаться, когда не используются (например, при смене сцены/региона игры)[[30]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L268-L271). На ранних этапах разработки Addressables не обязательны, но архитектура готова к их добавлению позднее[[31]](https://github.com/mrsokol2552-source/Game/blob/59b3aa7254ed979296ecd24d2d389cb586acb53a/docs/AUDIO_INTEGRATION.md#L52-L55).

Наконец, все группы микшера и параметры должны быть заранее созданы в ресурсе AudioMixer (см. шаги плана интеграции). SFX-менеджер в сцене связывается с соответствующими AudioMixerGroup (через инспектор или программно) и готов проигрывать звуки сразу после инициализации.

## 3. Интеграция музыкальной подсистемы

### 3.1. Режимы музыки и переходы состояний

Музыкальное сопровождение динамически реагирует на обстановку в игре. Выделено три основных режима музыки для игрового процесса: Calm (спокойствие), Tense (напряжение) и War (бой)[[32]](https://github.com/mrsokol2552-source/Game/blob/59b3aa7254ed979296ecd24d2d389cb586acb53a/docs/AUDIO_INTEGRATION.md#L13-L15). Каждый режим имеет свою тему или вариацию фоновой музыки, отражающую настроение:

* Calm: спокойная фоновая музыка, когда на базе или в мире относительно тихо и безопасно. Минимум напряжения.
* Tense: тревожная, нарастающая музыка, когда обстановка накаляется (например, приближается опасность, волна врагов на подходе).
* War: активная, агрессивная музыкальная тема, звучащая во время прямых боевых столкновений, осады базы или крупной битвы.

Переходы между этими состояниями осуществляются плавно с помощью AudioMixer Snapshots и/или кросс-фейда треков. В Unity настроены три снапшота микшера (Calm, Tense, War), которые могут переключаться методом AudioMixer.TransitionToSnapshots() за ~0.25–0.40 с[[32]](https://github.com/mrsokol2552-source/Game/blob/59b3aa7254ed979296ecd24d2d389cb586acb53a/docs/AUDIO_INTEGRATION.md#L13-L15). Снапшоты управляют как уровнями самой музыки, так и параметрами микшера (например, отключение ducking в War, см. разд. 2.2). Это обеспечивает бесшовное изменение: например, при переходе Calm → Tense музыка плавно перейдет на более напряженную тему и микшер подстроит эффекты (чуть ярче SFX, слегка приглушая музыку при громких звуках).

Помимо основных режимов, система поддерживает и другие музыкальные состояния, если они понадобятся: например, MainMenu/WorldMap (меню или карта мира), Stealth (скрытное прохождение), Aftermath/Ruins (состояние после боя или разрушенный мир). Эти дополнительные состояния упоминаются в планах на будущее[[33]](https://github.com/mrsokol2552-source/Game/blob/59b3aa7254ed979296ecd24d2d389cb586acb53a/docs/AUDIO_INTEGRATION.md#L11-L15), но в рамках текущей интеграции фокус на трех главных режимах игры (Calm/Tense/War) для боевой части.

### 3.2. Класс MusicDirector и аудио-снапшоты (Presentation)

Аналогично SfxManager для эффектов, в Presentation-слое вводится менеджер музыки – условно назовем его MusicDirector (MonoBehaviour). Он отвечает за управление музыкальными треками и состояниями. Основные функции MusicDirector:

* Проигрывание текущего трека/темы: держит активный AudioSource (или несколько для плавного перехода) с фоновым музыкальным клипом.
* Переключение состояния музыки: метод SetState(MusicState newState) изменяет режим (Calm, Tense, War), запуская соответствующий переход – в простейшем случае, меняет аудио-клип на нужный и делает кросс-фейд, а также переключает снапшот аудиомикшера под новое состояние[[34]](https://github.com/mrsokol2552-source/Game/blob/59b3aa7254ed979296ecd24d2d389cb586acb53a/docs/AUDIO_INTEGRATION.md#L8-L11).
* Управление микшером: при смене состояния вызывает AudioMixerSnapshot.TransitionTo() для одного из настроенных снапшотов (CalmSnapshot, TenseSnapshot, WarSnapshot). Снапшот автоматически применит изменения уровней (в т.ч. ducking, параметры SFX\_War и пр.). Например, calmSnapshot.TransitionTo(0.3f) – перейти к Calm за 0.3 с.
* Кросс-фейд треков: если разные состояния используют разные аудио-треки, MusicDirector обеспечивает плавное переключение. Возможны два подхода:
* Один источник с плавной сменой клипа: использовать встроенный AudioMixer переход – напр., два снапшота, в одном громкость старого трека 100% нового 0%, а в другом наоборот, и TransitionToSnapshots микширует их. Либо вручную уменьшать громкость текущего AudioSource и увеличивать другого.
* Два AudioSource одновременно: один играет старый трек, второй – новый, и через скрипт или анимацию параметра происходит перекрестное затухание. Архитектура должна позволять оба метода. На начальном этапе можно упростить: иметь один looping AudioSource и при смене состояния сразу переключать clip (с коротким фейдом).
* Инициализация и цикл: MusicDirector при старте игры может начинать с Calm (или меню) трека. Он также отслеживает, когда нужно переключиться (см. 3.3) и не позволяет повторных переключений, если новое состояние такое же как текущее.

Пример упрощенного кода MusicDirector:

public enum MusicState { Calm, Tense, War }  
  
public class MusicDirector : MonoBehaviour  
{  
 public AudioMixer musicMixer;  
 public AudioMixerSnapshot calmSnapshot, tenseSnapshot, warSnapshot;  
 public AudioSource musicSource; // основной AudioSource для музыки  
 public AudioClip calmTrack, tenseTrack, warTrack;  
  
 private MusicState currentState;  
  
 void Start() {  
 currentState = MusicState.Calm;  
 musicSource.clip = calmTrack;  
 musicSource.loop = true;  
 musicSource.Play();  
 calmSnapshot.TransitionTo(0f); // установить начальный снапшот  
 }  
  
 public void SetState(MusicState newState)  
 {  
 if (newState == currentState) return;  
 currentState = newState;  
 // Выбор нужного трека и снапшота  
 switch (newState) {  
 case MusicState.Calm:  
 musicSource.clip = calmTrack;  
 musicSource.Play();  
 calmSnapshot.TransitionTo(0.3f);  
 break;  
 case MusicState.Tense:  
 musicSource.clip = tenseTrack;  
 musicSource.Play();  
 tenseSnapshot.TransitionTo(0.3f);  
 break;  
 case MusicState.War:  
 musicSource.clip = warTrack;  
 musicSource.Play();  
 warSnapshot.TransitionTo(0.3f);  
 break;  
 }  
 // Примечание: для плавного перехода можно сначала запустить новый трек на втором AudioSource и   
 // постепенно микшировать, используя переход снапшотов или вручную анимируя volume.  
 }  
}

Комментарии: Здесь для простоты используется один AudioSource и мгновенное переключение clip при смене состояния (что подходит, если треки логически разделены по ситуациям). Реализация может быть улучшена добавлением второго фонового AudioSource для оверлапа. Важнее то, что при каждом переключении вызывается соответствующий снапшот микшера (Calm/Tense/War) с плавным переходом 0.3 с. Эти снапшоты уже настроены так, чтобы помимо самой музыки подстраивать и SFX-микшер (ducking, параметры и т.д.)[[7]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L135-L137).

В будущем MusicDirector может расшириться до более сложной логики (например, адаптивные треки). Архитектура заложена модульно: есть отдельный компонент для музыки, который можно менять, не затрагивая логику SFX.

### 3.3. Управление музыкой из Application-слоя (события и состояния)

В Application-слое музыка управляется либо напрямую через вызов метода API, либо (лучше) через события, отражающие смену состояния игры. Например, система игровых состояний может публиковать событие OnGameStateChanged(War) или специальные события: OnCombatStarted, OnCombatEnded, OnAlertLevelChanged(Tense) и т.п.

При получении такого события Presentation-слой (где работает MusicDirector) принимает решение сменить музыкальный режим. В простейшем случае можно напрямую вызвать MusicDirector.I.SetState(MusicState.War) из соответствующего контроллера, но в духе событийной архитектуры лучше также использовать EventBus. Например, MusicDirector подписывается на событие GamePhaseChanged и, проверяя фазу, переключает музыку.

Связка с Application-слоем обеспечена тем, что само состояние Calm/Tense/War определяется игровой логикой: - Calm – базовое состояние, когда нет угроз (может соответствовать статусу базы: безопасность).  
- Tense – повышенная готовность или тревога (например, обнаружен враг, но бой еще не начался).  
- War – активная боевая фаза (рейд врагов, осада и пр.).

Игровая логика знает, когда перейти между этими состояниями, и сообщает об этом. Таким образом, музыка становится частью отклика системы на те же события, что влияют и на геймплей (напр., режим War влияет не только на музыку, но и на геймплейные аспекты).

Важно, что логика не управляет аудио напрямую (не выбирает треки по имени), она лишь сообщает о смене состояния музыки. Конкретные же аудио-ресурсы (какой трек играть) и микширование определяются либо в MusicDirector, либо в конфигурации Infrastructure-слоя.

MusicConfig: В более продвинутом варианте вводится конфигурационный ресурс (например, ScriptableObject MusicConfig), мапящий состояния игры на списки аудио-клипов или адресные метки[[35]](https://github.com/mrsokol2552-source/Game/blob/59b3aa7254ed979296ecd24d2d389cb586acb53a/docs/AUDIO_INTEGRATION.md#L2-L5). Тогда MusicDirector при переходе состояния может выбирать случайный трек из списка (если для разнообразия предусмотрено несколько мелодий на одно состояние) или загружать необходимый клип через Addressables. Пока контент невелик, можно обойтись жестко заданными полями (как calmTrack, tenseTrack в примере кода). Но архитектура допускает рост: например, для режима War может быть несколько вариаций боевой музыки, выбираемых случайно или циклически.

Еще один аспект – приоритеты музыки в War режиме. В режиме War приоритет отдается SFX (битва), поэтому музыка хотя и звучит мощно, но не должна заглушать важные звуки. Мы уже рассмотрели, что ducking может быть отключен, но при этом через параметр SFX\_War музыка несколько ослаблена относительно SFX[[36]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L132-L135). Также, если ресурсно ограниченно число одновременных звуков, можно повысить лимиты для SFX в бою, чтобы музыка никогда не занимала все аудиоканалы (в Unity by default 32 voices). Однако детально управлять приоритетами микширования можно настройкой Priority у AudioSource: SFX-источникам дать более высокий приоритет, чем музыкальному, если нужно.

### 3.4. Темы, слои и стили музыки (ambient, build-up, stinger)

Архитектура музыкальной подсистемы спроектирована гибко, чтобы поддерживать разные стили и приёмы музыкального сопровождения:

* Ambient Loop (фон): основной способ – зацикленные фоновые темы для каждого состояния. Например, спокойная базовая мелодия для Calm, тревожный пульсирующий эмбиент для Tense, драйвовая боевая тема для War. Все треки сведены для зацикливания без швов (seamless loop). Это уже включено в дизайн (см. выше импорт длинных амбиентов с Loop). Смена loop-трека происходит на смене состояния.
* Layer-based Build-up: возможность послоенного наращивания музыки. Вместо переключения всей музыки, можно накладывать дополнительные слои. Архитектурно это достигается или несколькими AudioSource, привязанными к разным группам микшера, или использованием микшера с несколькими слоями: напр., базовый слой играет всегда, а при Tense добавляется второй слой (перкуссия, струнные и т.п.), при War – третий слой (ударные, хор). MusicDirector может управлять громкостями слоёв через параметры или отдельные snapshots. Такой подход дает более плавное изменение настроения, т.к. основная тема продолжает звучать, просто обогащаясь новыми элементами.
* Stingers (музыкальные акценты): короткие музыкальные фрагменты, проигрываемые поверх фоновой музыки при определённых событиях. Например, при алярме (переход Calm->Tense) может прозвучать тревожный аккорд-стингер, или при появлении босса – отдельный музыкальный фрагмент. Архитектурно стингеры могут исполняться дополнительным AudioSource, маршрутизированным в тот же микшер (но возможно в отдельную группу, чтобы иметь свой уровень). MusicDirector или отдельный компонент может подписаться на важные события (скажем, OnOutbreak, OnBossAppeared) и запускать stinger-клип, параллельно управляясь с основной музыкой (например, немного приглушая основную музыку на время стингера, что можно сделать автоматикой микшера). В рамках общей системы это расширение: уже сейчас SFX-EventBus можно использовать и для музыкальных событий – просто добавив обработчик в MusicDirector.

Таким образом, заложенная система не жестко привязана к одному треку – она позволяет масштабировать музыкальный дизайн. На этапе базовой интеграции реализуются 3 режима с одним треком each, но в будущем не потребуется менять архитектуру, чтобы добавить, к примеру, многослойную адаптивную музыку или набор стингеров. Достаточно будет расширить MusicDirector и обновить конфигурацию (например, дополнительные AudioSource или параметры микшера).

## 4. Off-screen симуляция и поведение аудио

Off-screen в контексте нашей игры означает ситуации, когда игровая симуляция продолжается без визуализации на экране. Возможны случаи: - Игра свернута или игрок находится в меню, но симуляция мира идет в фоновом режиме. - Камера/экран переключены на другой участок, и какие-то события происходят "за кадром". - Специальный headless-режим (если планируется) для серверной логики или ускоренной симуляции без вывода.

Для звуковой подсистемы важно не тратить ресурсы впустую и не создавать дискомфорта игроку, когда событие вне поля зрения. Предусмотрено несколько уровней поддержки off-screen:

* Отключение/приглушение звука вне фокуса: Когда приложение не в фокусе (OnApplicationFocus(false)), можно глобально глушить звук или снижать громкость мастера. Unity позволяет это настроить (например, опция Mute Other Audio Sources). В зависимости от требований, мы либо полностью отключаем звук в фоне, либо оставляем только очень тихий фоновый шум. Но в целом, если игрок не смотрит, звуки можно безопасно не воспроизводить.
* Окклюзия off-screen: Если объект вне кадра, но все еще потенциально слышен (например, взрыв за границей экрана), используется параметр OcclusionLPF. Как упомянуто ранее, при off-screen (или если между камерой и источником есть стена) повышается OcclusionLPF, что накладывает низкочастотный фильтр и снижает громкость[[19]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L260-L263). Это придает реализм (приглушенный далекий звук) и одновременно смягчает нагрузку на уши игрока от событий, которых он не видит. SfxManager может получать информацию от системы видимости/камеры и устанавливать этот параметр глобально или на источник.
* LOD для звука: В больших сценах или при массовых событиях можно применять упрощение звука на дальних расстояниях. Например, вместо микроменеджмента сотни звуков толпы где-то далеко, можно свести их в один далёкий амбиентный шум. В SFX-документе предложено: “при сворачивании сектора → останавливаем дорогостоящие лупы, оставляем только агрегированные (толпа, радио) на низком уровне и с LPF”[[19]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L260-L263). Практически, это значит, что SfxManager/игровая логика должны отслеживать, какие зоны активны на экране. Если сектор становится неактивным, можно: остановить звуки шагов, стрельбы в нем, включить тихий общий loop толпы для того сектора. Когда сектор снова в кадре, наоборот, запустить индивидуальные эффекты.
* Headless режим: Если игра запускается без звуковой подсистемы (например, серверная логика или тест), архитектура не сломается: SfxManager и MusicDirector просто не будут созданы или отключены. Все события будут публиковаться в EventBus, но без подписанного обработчика ничего происходить не будет — что безопасно. Чтобы явно поддержать такой режим, можно добавить проверки if (SfxManager.I == null) return; при вызове аудио из логики или инициализировать заглушки. В Unity можно также регулировать AudioListener.volume или полностью отцепить/удалить AudioListener.

Суммарно, звуковая подсистема учитывает off-screen: важные звуки не пропадут (будут слышны приглушенно), но нагрузка и звукоряда рационально уменьшаются. Все параметры (OcclusionLPF, и др.) настраиваются, и QA-тесты должны проверить корректность: нет ли случаев, чтобы звук из офф-скрина отвлекал или, наоборот, полностью пропадал, когда должен чуть слышаться.

## 5. Сохранение и восстановление состояния аудио

Для поддержки сохранений/загрузок игры необходимо обеспечить консистентность аудиосостояния при переходах:

* Состояние музыки: Текущий музыкальный режим (Calm/Tense/War) должен сохраняться, чтобы после загрузки игры музыка соответствовала обстановке. Например, если игра была сохранена во время атаки (War), то при загрузке сразу включается боевой трек и снапшот War, а не спокойная музыка. Проще всего сохранять идентификатор состояния (например, строку или enum) в составе сохранения. При загрузке, до или сразу после восстановления игрового мира, вызывается MusicDirector.SetState(savedState). MusicDirector должен уметь применить состояние мгновенно (или очень быстро) без лишних переходов, чтобы не было рассинхрона. Позицию трека (time) можно не сохранять, обычно начинает заново, но при желании можно сохранить TimeSamples позиции и установить musicSource.timeSamples после назначения клипа.
* Активные звуковые эффекты и петли: Как правило, краткосрочные SFX не нуждаются в сохранении – к моменту загрузки они уже отзвучали. Однако, длительные loop-звуки, соответствующие длительным событиям (например, звук строительства, сирена тревоги), могут потребовать восстановления. Здесь многое зависит от логики: например, состояние "стройка в процессе" само по себе при загрузке может вызвать событие, на которое SfxManager снова запустит loop. Если же нет, можно сохранить факт активных лупов: например, что на объекте X игрался loop звук Y, и при загрузке сразу его возобновить. Но предпочтительнее положиться на игровую логику – она должна при восстановлении запустить или остановить процессы, а аудио реагирует как обычно на события.
* Параметры микшера (громкости): Если игрок изменял настройки громкости (глобальные громкости музыки/SFX), они хранятся в настройках пользователя и обычно применяются при старте игры или загрузке (например, установкой AudioMixer параметров MasterVolume, MusicVolume, SFXVolume). Это вне рамок конкретной подсистемы, но следует не забыть про них.
* Параметры среды: Некоторые параметры, как SFX\_Tension, могут зависеть от плавных динамик игры (например, скрытая метрика напряжения). Лучше воспроизводить их из геймплея. Однако, если есть длительные полу-постоянные значения, можно сохранить. К примеру, если SFX\_Tension использовался как настройка мирового состояния (0 = мирно, 1 = апокалипсис), то его можно сохранить вместе с игровым состоянием.
* Снапшот микшера: Unity не предоставляет прямого сохранения снапшота, но зная состояние (Calm/Tense/War) и, возможно, текущее значение OcclusionLPF, можно восстановить звуковую картину. При загрузке применяем нужный снапшот; параметр OcclusionLPF устанавливается в соответствии с новыми условиями (например, если загрузили игру и сразу игрок в укрытии, то EventBus-система должна вкинуть событие окклюзии или сам SfxManager проверит видимость и поставит параметр).

Вывод: механизм сохранения интегрируется с игровым. От аудиоподсистемы требуется экспозиция необходимой информации (например, иметь метод GetMusicState() для сохранения, или SfxManager может регистрировать какие лупы активны). При загрузке – наоборот, вызывать методы установки состояния. Это можно сделать частью общего интерфейса сохранения/загрузки (например, AudioSubsystem.SaveState() / LoadState(data)). В текущем объеме достаточно сохранять тип музыкальной темы и возможно активность особых звуковых эвентов.

## 6. Глоссарий терминов

* SFX (Sound Effects) – звуковые эффекты, короткие звуки, сопровождающие действия (выстрелы, шаги, UI-клики и др.).
* One-shot звук – звук, проигрываемый один раз по событию (не зацикленный). Например, звук нажатия кнопки.
* Loop (луп) – зацикленный звуковой клип, продолжающийся, пока необходим эффект. Пример: гул двигателя, сирена тревоги, фоновый шум.
* Audio Mixer / Mixer Group – аудиомикшер Unity и его группы. Группы позволяют разделять потоки звука по категориям (см. SFX\_UI и др.), чтобы регулировать их громкость и эффекты независимо. Mixer-asset содержит группы и параметры, может иметь Snapshot’ы.
* Snapshot (снапшот) – сохраненное состояние AudioMixer (уровни групп, значения параметров). Несколько снапшотов позволяют мгновенно переключать микшер между разными настройками (с плавным переходом). Используются для реализации режимов Calm/Tense/War.
* Ducking (ducking) – автоматическое уменьшение громкости одной аудиодорожки при звучании другой, более приоритетной. В нашем случае – приглушение музыки при громких SFX через side-chain компрессор[[7]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L135-L137).
* Occlusion LPF (Occlusion Low-Pass Filter) – эффект низкочастотного фильтра, имитирующий приглушение звука преградами или расстоянием. Параметр OcclusionLPF в микшере контролирует силу этого эффекта[[6]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L132-L134).
* EventBus – шаблон проектирования «шина событий». Компоненты публикуют события, на которые могут подписываться любые слушатели. Позволяет отделить отправителя события от логики получателя. В контексте аудио – игра генерирует событие, аудиосистема (SfxManager/MusicDirector) получает и обрабатывает его.
* Addressables – система адресуемых ассетов Unity. Позволяет загружать и выгружать ресурсы по адресу или метке в рантайме, удобно для управления большим количеством аудиофайлов, разбивки на пакеты контента и снижения потребления памяти[[29]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L267-L275).
* Vorbis, PCM – форматы аудиосжатия. PCM – несжатый (или слабо сжатый) формат (больше размер, минимум потерь качества, быстрый старт), Vorbis – сжатый формат (меньше размер, требует декодирования). Используются при импорте звука для балансировки качества, размера и производительности[[37]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L72-L80).
* Spatial Blend (2D/3D) – параметр AudioSource, определяющий, насколько звук позиционируется в 3D пространстве. 0 = полностью 2D (стерео панорама не меняется от позиции), 1 = полностью 3D (громкость и панорама зависят от положения относительно слушателя).
* Rolloff – кривая затухания громкости звука по мере удаления от слушателя. Custom Rolloff – пользовательская кривая, которую мы настраиваем для более реалистичного затухания, вместо стандартной линейной или экспоненциальной.
* Stinger – короткий музыкальный фрагмент-акцент, обычно синхронизированный с определенным событием (выстрел оркестрового удара при появлении опасности и т.п.), накладывается поверх фоновой музыки.
* Layered music (многослойная музыка) – способ организации музыки, при котором одновременно могут играть несколько синхронизированных треков (слоев), и в зависимости от состояния какие-то слои добавляются или убираются. Например, базовая атмосфера + ритмический слой + мелодический слой, включаемые по мере возрастания напряжения.

## 7. Технические требования и рекомендации

* Формат аудио-контента: Исходные мастер-файлы звуков – 48 kHz / 24-bit WAV, с peak не выше -1 dBTP[[38]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L9-L12). При экспорте в игру используются сжатые форматы (Vorbis OGG) или PCM для критичных звуков; моно предпочтительнее для источников, где стерео не даёт выигрыш (шаги, одиночные эффекты)[[23]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L9-L16).
* Организация и нейминг ассетов: Соблюдать файловую структуру Assets/Audio/... и соглашение имен SFX\_<Category>\_<Event>\_Var## для унификации[[22]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L36-L44). Это облегчает поиск и автоматизацию (скрипты могут по имени определять категорию, вариант и т.д.).
* Вариативность эффектов: Для каждого важного события иметь по 5–8 вариаций звука[[20]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L12-L15). Менеджер SFX должен выбирать случайный вариант и слегка варьировать pitch/volume, чтобы избежать монотонности.
* Настройки импорта в Unity: Применять правильный тип загрузки: Decompress on Load для коротких (<1 с) клипов, Compressed In Memory для средних лупов (~1–2 с), Streaming для длинных амбиентов (>5 с)[[24]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L16-L18). Использовать Vorbis с качеством ~0.6–0.7 для сжатия, кроме случаев, когда нужен PCM (очень частые короткие звуки). Убедиться, что длинные клипы помечены как Loop при импорте.
* AudioMixer и Snapshots: Создать в микшере группы SFX\_UI, SFX\_World, SFX\_Combat, SFX\_Voice, SFX\_Ambience и объединить их под шину SFX/Main[[4]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L129-L134). Экспонировать параметры SFX\_Tension, SFX\_War, OcclusionLPF и настроить их влияния (EQ, дакуинг, фильтр)[[39]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L130-L137). Настроить три снапшота: Calm, Tense, War – с подходящими уровнями групп и параметров, проверить плавность переходов (время ~0.3 с, без щелчков)[[7]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L135-L137).
* EventBus интеграция: Все новые игровые события, требующие звука, должны быть добавлены в EventBus и обработаны в SfxManager (или MusicDirector). Список маппинга событий на звуки вести в документации (или в конфиге)[[9]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L144-L152). Проверить, что подписка SfxManager охватывает нужные категории (UI, Combat, World, Voice, Ambience, Infection и т.д. по необходимости)[[40]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L300-L305).
* Performance и пул источников: Ожидается большое количество коротких звуков (особенно в бою), поэтому реализован пул AudioSource. Необходимо профилировать производительность: при массовой стрельбе, взрывах и толпе не должно быть пропусков или клиппинга[[41]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L287-L295). Пул должен иметь разумный размер или динамический рост, а удаленные источники – переиспользоваться.
* Off-screen оптимизации: Настроить пороговые расстояния затухания (MinDistance, Custom Rolloff) так, чтобы дальние звуки не сохраняли полную громкость[[19]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L260-L263). Убедиться, что OcclusionLPF параметр применяется при реальных сценариях (юнит за стеной, события вне экрана)[[19]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L260-L263). Оптимизировать лупы: дальние циклы отключаются или агрегируются.
* Addressables (опционально): Пометить аудио-ассеты метками по категориям[[29]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L267-L275). Настроить Addressables Groups (например, по сценам или по категориям звуков). В будущем – реализовать выгрузку неиспользуемых звуков при длительной игре или переходе между сценами (Release by label)[[30]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L268-L271).
* Сохранение состояния: Интерфейс сохранения игры должен сохранять текущее музыкальное состояние (и, при необходимости, активные долгие аудио-события). После загрузки эти состояния необходимо восстановить вызовом методов MusicDirector/SfxManager до возобновления геймплея, чтобы аудио сразу соответствовало ситуации.
* Документация и тесты: Обновлять документацию (аудиокаталог, маппинг событий) по мере добавления нового контента[[42]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L302-L305). Провести калибровку громкостей: UI-звуки не должны заглушать бой, бой не должен полностью забивать музыку[[41]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L287-L295) – настроить относительные уровни групп. Протестировать бесшовность лупов и корректность переходов снапшотов (Calm↔Tense↔War)[[43]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L288-L292), а также стресс-тест (много звуков одновременно) на отсутствие аудио-артефактов[[44]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L287-L294).

Со внедрением описанной звуковой подсистемы архитектура игрового проекта станет богаче, обеспечивая погружение через звук, при этом сохраняя принцип разделения ответственностей и масштабируемости. Дизайн звука интегрирован с игровыми механиками (через события и состояния), что позволит легко расширять аудио-составляющую вместе с ростом проекта. [[8]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L141-L149)[[45]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L129-L137)

[[1]](https://github.com/mrsokol2552-source/Game/blob/59b3aa7254ed979296ecd24d2d389cb586acb53a/docs/AUDIO_INTEGRATION.md" \l "L7-L9) [[31]](https://github.com/mrsokol2552-source/Game/blob/59b3aa7254ed979296ecd24d2d389cb586acb53a/docs/AUDIO_INTEGRATION.md#L52-L55) [[32]](https://github.com/mrsokol2552-source/Game/blob/59b3aa7254ed979296ecd24d2d389cb586acb53a/docs/AUDIO_INTEGRATION.md#L13-L15) [[33]](https://github.com/mrsokol2552-source/Game/blob/59b3aa7254ed979296ecd24d2d389cb586acb53a/docs/AUDIO_INTEGRATION.md#L11-L15) [[34]](https://github.com/mrsokol2552-source/Game/blob/59b3aa7254ed979296ecd24d2d389cb586acb53a/docs/AUDIO_INTEGRATION.md#L8-L11) [[35]](https://github.com/mrsokol2552-source/Game/blob/59b3aa7254ed979296ecd24d2d389cb586acb53a/docs/AUDIO_INTEGRATION.md#L2-L5) AUDIO\_INTEGRATION.md

<https://github.com/mrsokol2552-source/Game/blob/59b3aa7254ed979296ecd24d2d389cb586acb53a/docs/AUDIO_INTEGRATION.md>

[[2]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L39-L48) [[3]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L267-L274) [[4]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L129-L134) [[5]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L130-L134) [[6]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L132-L134) [[7]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L135-L137) [[8]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L141-L149) [[9]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L144-L152) [[10]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L144-L153) [[11]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L274-L283) [[12]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L187-L195) [[13]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L248-L254) [[14]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L222-L230) [[15]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L230-L238) [[16]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L232-L240) [[17]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L239-L247) [[18]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L224-L230) [[19]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L260-L263) [[20]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L12-L15) [[21]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L22-L31) [[22]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L36-L44) [[23]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L9-L16) [[24]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L16-L18) [[25]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L72-L75) [[26]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L128-L136) [[27]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L83-L87) [[28]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L110-L118) [[29]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L267-L275) [[30]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L268-L271) [[36]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L132-L135) [[37]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L72-L80) [[38]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L9-L12) [[39]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L130-L137) [[40]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L300-L305) [[41]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L287-L295) [[42]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L302-L305) [[43]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L288-L292) [[44]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L287-L294) [[45]](https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md#L129-L137) SFX\_INTEGRATION.md

<https://github.com/mrsokol2552-source/network/blob/f1be0a71b895eb8a1527142abe63dcd84f80133d/G/SFX_INTEGRATION.md>