ITUПубликации
Рекомендации

Международный союз электросвязи
Сектор радиосвязи

Рекомендация МСЭ-R BS.2127-1

(11/2023)

Серия BS: Радиовещательная служба (звуковая)

Рендерер модели определения аудиофайла для усовершенствованных звуковых систем

Предисловие

Роль Сектора радиосвязи заключается в обеспечении рационального, справедливого, эффективного и экономичного использования радиочастотного спектра всеми службами радиосвязи, включая спутниковые службы, и проведении в неограниченном частотном диапазоне исследований, на основании которых принимаются Рекомендации.

Всемирные и региональные конференции радиосвязи и ассамблеи радиосвязи при поддержке исследовательских комиссий выполняют регламентарную и политическую функции Сектора радиосвязи.

Политика в области прав интеллектуальной собственности (ПИС)

Политика МСЭ-R в области ПИС излагается в общей патентной политике МСЭ-T/МСЭ-R/ИСО/МЭК, упоминаемой в Резолюции МСЭ-R 1. Формы, которые владельцам патентов следует использовать для представления патентных заявлений и деклараций о лицензировании, представлены по адресу: http://www.itu.int/ITU-R/go/patents/ru, где также содержатся Руководящие принципы по выполнению общей патентной политики МСЭ-T/МСЭ-R/ИСО/МЭК и база данных патентной информации МСЭ-R.

Серии Рекомендаций МСЭ-R

(Представлены также в онлайновой форме по адресу: http://www.itu.int/publ/R-REC/ru.)

Серия	Название
BO	Спутниковое радиовещание
BR	Запись для производства, архивирования и воспроизведения; пленки для телевидения
BS	**Радиовещательная служба (звуковая)**
BT	Радиовещательная служба (телевизионная)
F	Фиксированная служба
M	Подвижные службы, служба радиоопределения, любительская служба и относящиеся к ним спутниковые службы
P	Распространение радиоволн
RA	Радиоастрономия
RS	Системы дистанционного зондирования
S	Фиксированная спутниковая служба
SA	Космические применения и метеорология
SF	Совместное использование частот и координация между системами фиксированной спутниковой службы и фиксированной службы
SM	Управление использованием спектра
SNG	Спутниковый сбор новостей
TF	Передача сигналов времени и эталонных частот
V	Словарь и связанные с ним вопросы

Примечание. – *Настоящая Рекомендация МСЭ-R утверждена на английском языке в соответствии с процедурой, изложенной в Резолюции МСЭ-R 1.*

Электронная публикация
Женева, 2024 г.

РЕКОМЕНДАЦИЯ МСЭ-R BS.2127-1*

Рендерер модели определения аудиофайла для усовершенствованных звуковых систем

(Вопрос МСЭ-R 135-2/6)

(2019-2023)

Сфера применения

В настоящей Рекомендации определен эталонный рендерер для использования, в том числе для обмена программами, с усовершенствованными звуковыми системами, которые описаны в Рекомендации МСЭ-R BS.2051-2, и относящимися к звуковому сигналу метаданными, которые описаны в модели определения аудиофайла (ADM) в Рекомендации МСЭ-R BS.2076-1. Звуковой рендерер преобразует набор звуковых сигналов с соответствующими метаданными в иную конфигурацию звуковых сигналов и метаданных, основываясь на предоставленных метаданных контента и локальных метаданных окружающей среды.

ПРИМЕЧАНИЕ. – В настоящее время разрабатываются руководящие указания, объясняющие использование рендерера.

Ключевые слова

ADM, модель определения аудиофайла, метаданные, рендерер, AdvSS, усовершенствованная звуковая система, звук на основе канала, звук на основе объекта, звук на основе сцены, многоканальный звук.

Ассамблея радиосвязи МСЭ,

учитывая,

a) что в Рекомендации МСЭ-R BS.1909 "Требования к рабочим характеристикам перспективной многоканальной стереофонической звуковой системы, предназначенной для использования с сопровождающим изображением и без него" определены требования к рабочим характеристикам перспективных звуковых систем с сопровождающим изображением и без него;

b) что в Рекомендации МСЭ-R BS.2051 "Усовершенствованная звуковая система для производства программ" определены усовершенствованная звуковая система, представляющая собой систему с конфигурацией воспроизведения звука, помимо тех, что определены в Рекомендации МСЭ-R BS.775, или систему с любой конфигурацией воспроизведения звука, которая может поддерживать входные сигналы на основе канала, объекта или сцены либо их комбинацию с метаданными;

c) что в Рекомендации МСЭ-R BS.2076 "Модель определения аудиофайла" определена структура модели метаданных, позволяющая достоверно описывать формат и содержимое аудиофайлов;

d) что в Рекомендации МСЭ-R BS.2094 "Общие определения для модели определения аудиофайла" содержится набор общих определений для модели определения аудиофайла;

e) что в Рекомендации МСЭ-R BS.2125 "Последовательное представление модели определения аудиофайла" определен формат метаданных на основе модели определения аудиофайла в разбивке на временны́е ряды кадров;

f) что для воспроизведения через усовершенствованные звуковые системы требуется рендеринг метаданных, связанных со звуковыми сигналами, для представления контента в одной из конфигураций громкоговорителей в соответствии с Рекомендацией МСЭ-R BS.2051;

g) что пользователи усовершенствованных звуковых систем должны иметь свободу выбора метода рендеринга;

---

* Настоящую Рекомендацию следует довести до сведения ИСО, МЭК, SMPTE и ЕТСИ.

h) что желательно иметь открытую спецификацию единого эталонного метода рендеринга, который можно использовать для программ усовершенствованных звуковых систем;

i) что единый эталонный рендерер должен позволять производителям контента и радиовещательным организациям осуществлять контроль управления качеством во время производства контента, проверять использование метаданных и обеспечивать взаимодействие с другими элементами производственной цепочки,

рекомендует

1 использовать методы рендеринга, описанные в Приложении 1, в качестве эталона для интерпретации метаданных ADM, описанных в Рекомендации МСЭ-R BS.2076-1, и сопутствующих звуковых сигналов;

2 считать приводимое ниже Примечание 1 частью настоящей Рекомендации.

ПРИМЕЧАНИЕ 1. – Соблюдение настоящей Рекомендации носит добровольный характер. Вместе с тем настоящая Рекомендация может содержать некоторые обязательные положения (например, для обеспечения функциональной совместимости или применимости), и в таком случае соблюдение Рекомендации достигается при выполнении всех этих обязательных положений. Для выражения требований используется слово "должен" (shall) или некоторые другие обязывающие выражения, такие как "обязан" (must), а также их отрицательные формы. Употребление таких слов не означает, что от какой-либо стороны требуется полное или частичное соблюдение положений настоящей Рекомендации.

Приложение 1

Спецификации рендерера ADM для усовершенствованных звуковых систем

СОДЕРЖАНИЕ

Стр.

1 Введение

В настоящей Рекомендации описывается звуковой рендерер, обеспечивающий полную интерпретацию метаданных модели определения аудиофайла (ADM), определенных в Рекомендации МСЭ-R BS.2076-1. Метаданные ADM рекомендуется применять для описания аудиоформатов, используемых при производстве программ для усовершенствованных звуковых систем (AdvSS), также называемых звуковыми системами следующего поколения (NGA). Этот рендерер способен воспроизводить звуковые сигналы для всех конфигураций громкоговорителей, указанных в Рекомендации МСЭ-R BS.2051-2.

Настоящая спецификация сопровождается эталонной реализацией с открытым исходным кодом для обработки файлов ADM, написанной на языке Python, которая доступна по адресу: https://www.itu.int/dms_pub/itu-r/oth/0a/07/R0A0700003E0001ZIPE.zip.

Настоящая спецификация служит описанием указанного кода.

1.1 Сокращения/глоссарий

ADM	Audio definition model	Модель определения аудиофайла
BMF	Broadcast metadata exchange format	Формат обмена радиовещательными метаданными
BW64	Broadcast wave 64 format	Формат аудиофайлов "радиовещательная волна 64 бита"
BWF	Broadcast wave format	Формат аудиофайлов "радиовещательная волна"
HOA	Higher-order ambisonics	Амбиофония высшего порядка
NGA	Next generation audio	Звуковые системы следующего поколения
PSP	Point source panner	Точечный панораматор
VBAP	Vector base amplitude panning	Векторное амплитудное панорамирование
XML	Extensible markup language	Расширяемый язык разметки

2 Соглашения

2.1 Условные обозначения

В настоящей Рекомендации используются следующие соглашения:

– текст, выделенный курсивом, относится к элементам, подэлементам, параметрам или атрибутам ADM из Рекомендации МСЭ-R BS.2076-1 – *audioObject*;

– текст с моноширинным шрифтом относится к исходному коду (переменным, функциям, классам) эталонной реализации – `core.point_source.PointSourcePanner`. Следует отметить, что для удобства чтения префикс `iar.` опущен;

– набранные жирным шрифтом заглавные буквы используются для обозначения матриц – **X**;

– набранные жирным шрифтом строчные буквы используются для обозначения векторов – **x**;

– нижний индекс в виде x_n означает n-й элемент вектора **x**;

– разделы текста с моноширинным шрифтом, выделенные цветом, используются для описания структур данных:

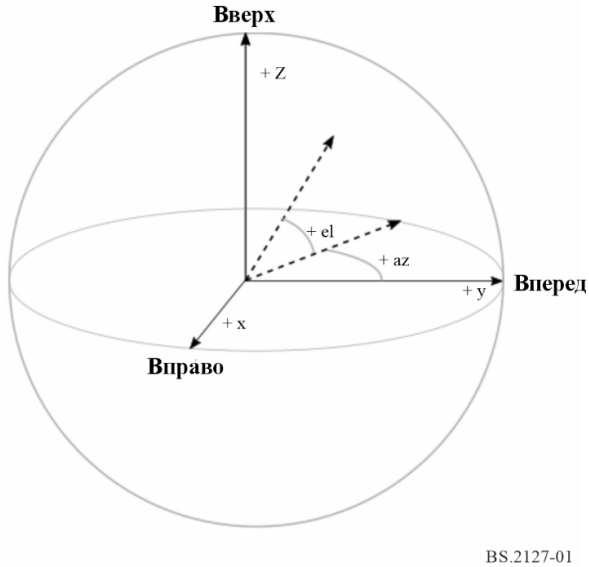
```
struct PolarPosition : Position {
  float azimuth, elevation, distance = 1;
};
```

2.2 Система координат

В настоящей Рекомендации используются как декартовы, так и полярные координаты.

РИСУНОК 1

Система координат



Полярные координаты определены в соответствии с Рекомендацией МСЭ-R BS.2076-1 следующим образом:

– азимут, обозначаемый φ, – это угол в горизонтальной плоскости со значением 0° в направлении вперед и положительными значениями в направлении против часовой стрелки;

– угол места, обозначаемый θ, – это угол над горизонтальной плоскостью со значением 0° в направлении вперед и положительными значениями в направлении вверх.

Декартовы координаты определены в соответствии с Рекомендацией МСЭ-R BS.2076-1 следующим образом:

– положительная ось *Y* указывает вперед;

– положительная ось *X* указывает вправо;

– положительная ось *Z* указывает вверх.

Для декодера HOA, описанного в разделе 9, используется система координат HOA и условные обозначения, приведенные в Рекомендации МСЭ-R BS.2076-1, где:

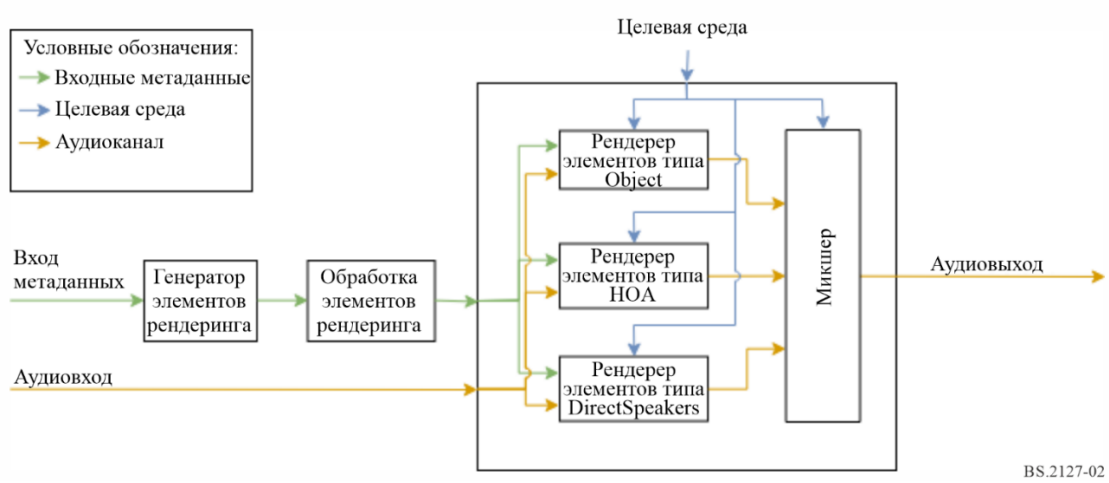
– угол места, обозначаемый θ, – это угол в радианах от положительной оси *Z*;

– азимут, обозначаемый φ, – это угол в радианах в горизонтальной плоскости со значением 0 в направлении вперед и положительными значениями в направлении против часовой стрелки.

3 Структура

РИСУНОК 2

Общий обзор архитектуры



Общая архитектура состоит из нескольких основных компонентов и этапов обработки, описание которых приводится в следующих разделах настоящего документа.

– Преобразование данных ADM в набор элементов, подлежащих рендерингу, описывается в пункте 5.2.

– Дополнительная обработка для использования уровня важности и эмуляции преобразования применяется к элементам рендеринга, как описано в пункте 5.3.

– Сам рендеринг делится на подкомпоненты в зависимости от типа элемента (*typeDefinition*):
  - рендеринг объектно-ориентированного контента описывается в разделе 7;
  - рендеринг прямых сигналов на громкоговорители описывается в разделе 8;
  - рендеринг элементов HOA описывается в разделе 9;
  - общие части для всех компонентов описываются в разделе 6.

Обработка типа *Matrix* на диаграмме не показана, поскольку этот тип обрабатывается при создании элементов рендеринга и в рамках рендереров других типов.

3.1 Поведение целевой среды

При инициализации пользователь может выбрать расположение громкоговорителей из схем расположения, указанных в Рекомендации МСЭ-R BS.2051.

Номинальное расположение каждого громкоговорителя (`polar_nominal_position`) определено в Рекомендации МСЭ-R BS.2051. `M+SC` и `M−SC` имеют номинальные значения азимута 15° и −15°.

Реальное расположение каждого громкоговорителя (`polar_position`) может быть определено пользователем. В противном случае используется номинальное расположение. Заданное реальное расположение проверяется по диапазонам, приведенным в Рекомендации МСЭ-R BS.2051; если оно выходит за пределы диапазона, выдается сообщение об ошибке. Кроме того, абсолютный азимут обоих громкоговорителей `M+SC` и `M−SC` должен находиться между 5° и 25° или между 35° и 60°.

4 Интерфейс ADM-XML

ADM – это общая модель метаданных, которую можно соответственно представить в виде XML-документа. В следующих подразделах описывается способ преобразования ADM во внутренние структуры данных. Эти структуры применяются в настоящей Рекомендации и соответствуют структурам данных, используемым в эталонной реализации.

Следует отметить, что хотя XML является типичной и распространенной формой представления метаданных ADM, рендерер не ограничивается этим представлением.

Преобразование между ADM и внутренними структурами данных следует набору простых правил, описанных ниже. Как всегда имеются некоторые исключения из правил, которые описываются в следующих подразделах.

– Все основные элементы ADM должны быть представлены как подкласс, производный от `ADMElement`, с сигнатурой:

```
class ADMElement {
  string id;
  ADM adm_parent;
  bool is_common_definition;
};
```

– Каждый класс элементов ADM должен расширяться всеми атрибутами и подэлементами ADM, преобразованными в атрибуты класса.

– Если подэлемент содержит несколько значений, он сам по себе является классом. Например, подэлемент *jumpPosition* – это класс с сигнатурой:

```
class JumpPosition {
  bool flag;
  float interpolationLength;
};
```

– В ходе анализа XML ссылки на другие элементы ADM хранятся в виде простых идентификаторов с использованием имени подэлемента в качестве имени атрибута (например, `AudioObject.audioPackFormatIDRef`). Чтобы упростить последующий доступ, эти ссылки затем разрешаются на следующем этапе, где разрешенные элементы добавляются непосредственно в каждую структуру данных (`AudioObject.audioPackFormats`).

Следуя этим правилам полную сигнатуру элемента `AudioContent` можно представить следующим образом:

```
class AudioContent : ADMElement {
  string audioContentName;
  string audioContentLanguage;
  LoudnessMetaData loudnessMetadata;
  int dialogue;
  vector<AudioObject*> audioObjects;
  vector<string> audioObjectIDRef;
};
```

Основные элементы ADM и их выделенные классы реализуются в элементах `fileio.adm.elements.main_elements`. Разрешение ссылок реализуется в каждом классе (в ADM и в каждом из основных элементов ADM) как метод `lazy_lookup_references`.

Анализ и запись ADM реализуются в модуле `fileio.adm.xml`.

4.1 Элемент AudioBlockFormat

Элемент *audioBlockFormat* отличается от других элементов ADM, поскольку его подэлементы и атрибуты различаются в зависимости от определения типа (*typeDefiniton)*. Для отражения этого `AudioBlockFormat` разбивается на несколько классов, по одному на каждое поддерживаемое значение *typeDefinition:* `AudioBlockFormatObjects`, `AudioBlockFormatDirectSpeakers` и `AudioBlockFormatHoa`.

Они реализуются в структурах `fileio.adm.elements.block_formats`.

4.2 Подэлементы расположения

Расположение представлено в ADM несколькими подэлементами *position*. Для упрощения внутренней обработки значения этих подэлементов объединяются в один атрибут в составе представления `AudioBlockFormat`.

Для *typeDefinition==Objects* это либо `ObjectPolarPosition`, либо `ObjectCartesianPosition` в зависимости от используемой системы координат. Для *typeDefinition==DirectSpeakers* это `DirectSpeakerPolarPosition` либо `DirectSpeakerCartesianPosition`.

4.3 Атрибут TypeDefinition

Атрибуты *typeDefinition* и *typeLabel* описывают всего одно свойство. По этой причине для их представления во внутренней структуре должен использоваться только один объект:

```
enum TypeDefinition {
  DirectSpeakers = 1;
  Matrix = 2;
  Objects = 3;
  HOA = 4;
  Binaural = 5;
};

enum FormatDefinition {
  PCM = 1;
};
```

5 Элементы рендеринга

`RenderingItem` – это представление элемента ADM, подлежащего рендерингу, которое содержит всю необходимую для этого информацию. Таким образом, этот элемент должен представлять один формат *audioChannelFormat* или группу форматов *audioChannelFormats*. Поскольку к каждому типу *typeDefinition* предъявляются разные требования, необходимо иметь разные структуры метаданных для каждого значения *typeDefinition*, чтобы адаптироваться к его конкретным требованиям.

В следующем подразделе используемые структуры метаданных описываются более подробно.

5.1 Структуры метаданных

`RenderingItems` построены на следующих базовых классах:

- `TypeMetadata` для хранения всех (возможно, изменяющихся во времени) параметров, необходимых для рендеринга элемента;
- `MetadataSource` для хранения ряда объектов `TypeMetadata`; и
- `RenderingItem` для связи `MetadataSource` с источником выборок звукового сигнала и дополнительной информацией, не обязательно требуемой рендереру.

Поскольку у каждого типа *typeDefinition* свои требования, классы `TypeMetadata` и `RenderingItem` должны быть разделены на подклассы для каждого значения *typeDefinition*, чтобы адаптироваться к его конкретным требованиям. Класс `MetadataSource` не зависит от *typeDefinition*. В структуре `ExtraData` объединены общие данные:

```
struct ExtraData {
  optional<duration> object_start;
  optional<duration> object_duration;
  ReferenceScreen reference_screen;
  Frequency channel_frequency;
};
```

Данные о важности должны храниться в структуре `ImportanceData`:

```
struct ImportanceData {
  optional<int> audio_object;
  optional<int> audio_pack_format;
};
```

Ссылки на входные выборки звукового сигнала должны инкапсулироваться в структуры `TrackSpec`, чтобы обеспечить возможность спецификации беззвучных дорожек и матричной обработки.

DirectTrackSpec указывает, что выборки должны считываться непосредственно с указанной входной дорожки. SilentTrackSpec указывает, что все выборки должны быть нулевыми:

```
struct TrackSpec {};

struct DirectTrackSpec : TrackSpec {
  int track_index;
};

struct SilentTrackSpec : TrackSpec {
};
```

Для поддержки *typeDefinition==DirectSpeakers* предусмотрены два типа TrackSpec. MatrixCoefficientTrackSpec указывает, что параметры, приведенные в coefficient (из элемента coefficient матрицы *audioBlockFormat*), применяются к выборкам input_track, а MixTrackSpec – что выборки из нескольких TrackSpec микшируются вместе:

```
struct MatrixCoefficientTrackSpec : TrackSpec {
  TrackSpec input_track;
  MatrixCoefficient coefficient;
};

struct MixTrackSpec : TrackSpec {
  vector<TrackSpec> input_tracks;
};
```

Это реализуется в модуле core.utils.metadata_input. В следующих подразделах подробно описываются конкретные реализации для каждого значения *typeDefinition*.

5.1.1 DirectSpeakers

Для *typeDefinition==DirectSpeakers* класс TypeMetadata должен содержать *audioBlockFormat*, список элементов *audioPackFormat*, ведущих к содержанию *audioChannelFormat*, плюс общие данные, собранные в ExtraData:

```
struct DirectSpeakersTypeMetadata : TypeMetadata {
  AudioBlockFormatDirectSpeakers block_format;
  vector<AudioPackFormat> audioPackFormats;
  ExtraData extra_data;
};
```

Поскольку каждый *audioChannelFormat*, для которого *typeDefinition==DirectSpeakers* может обрабатываться независимо, RenderingItem содержит только один TrackSpec:

```
struct DirectSpeakersRenderingItem : RenderingItem {
  TrackSpec track_spec;
  MetadataSource metadata_source;
  ImportanceData importance;
};
```

5.1.2 Matrix

typeDefinition==Matrix при рендеринге элементов других типов должен поддерживаться с использованием механизма TrackSpec, поэтому явные классы MatrixTypeMetadata или MatrixRenderingItem не требуются.

5.1.3 Objects

Класс ObjectTypeMetadata должен содержать *audioBlockFormat* плюс общие данные, собранные в ExtraData:

```
struct ObjectTypeMetadata : TypeMetadata {
  AudioBlockFormatObjects block_format;
  ExtraData extra_data;
};
```

Поскольку каждый *audioChannelFormat*, для которого *typeDefinition==Objects* может обрабатываться независимо, `RenderingItem` должен содержать только один `TrackSpec`:

```
struct ObjectRenderingItem : RenderingItem {
  TrackSpec track_spec;
  MetadataSource metadata_source;
  ImportanceData importance;
};
```

5.1.4 HOA

Для *typeDefinition==HOA* ситуация отличается от случаев *typeDefinition==DirectSpeakers* и *typeDefinition==Objects*, поскольку пакет объектов *audioChannelFormat* должен обрабатываться как одно целое. Поэтому `HOATypeMetadata` не содержит `audioBlockFormat` плюс `ExtraData`, а необходимая информация извлекается из *audioBlockFormat* и сохраняется непосредственно в `HOATypeMetadata`:

```
struct HOATypeMetadata : TypeMetadata {
  vector<int> orders;
  vector<int> degrees;
  optional<string> normalization;
  optional<float> nfcRefDist;
  bool screenRef;
  ExtraData extra_data;
  optional<duration> rtime;
  optional<duration> duration;
};
```

По той же причине отличается ситуация для `HOARenderingItem`. Здесь `HOARenderingItem` содержит не просто один `TrackSpec`, а вектор `TrackSpecs`:

```
struct HOARenderingItem : RenderingItem {
  vector<TrackSpec> track_specs;
  MetadataSource metadata_source;
  vector<ImportanceData> importances;
};
```

5.1.5 Binaural

Поскольку *typeDefinition==Binaural* не поддерживается, классы `BinauralTypeMetadata` или `BinauralRenderingItem` отсутствуют.

5.2 Определение элементов рендеринга

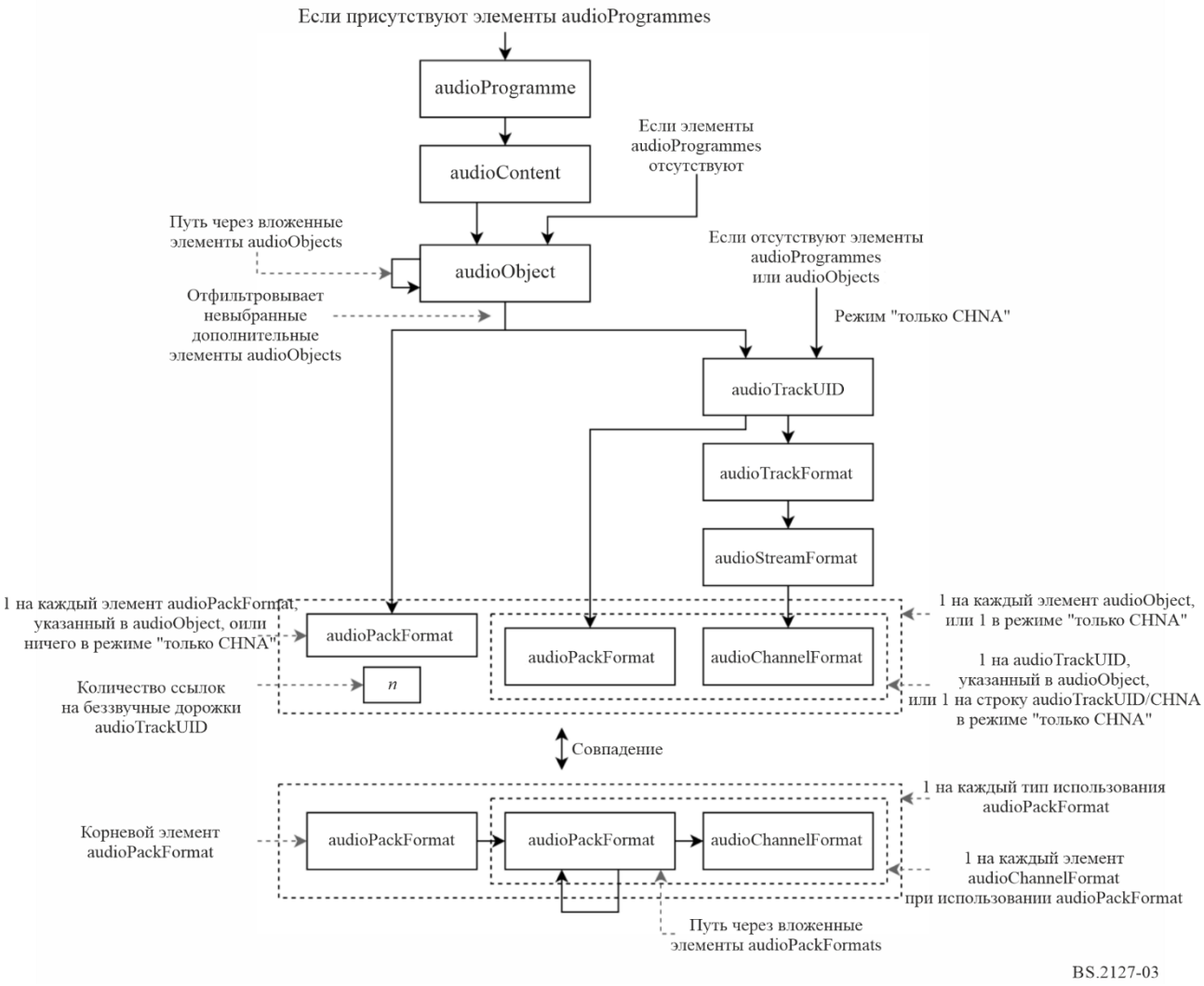
Для определения элементов `RenderingItems` необходимо проанализировать структуру ADM. На рисунке 3 иллюстрируется выбранный путь.

Состояние процесса выбора элемента передается между различными компонентами в одном объекте, называемом "состояние выбора элемента", который при полном заполнении представляет все компоненты, составляющие один элемент `RenderingItem`. Каждый компонент принимает одно состояние выбора элемента и возвращает его копии (от нуля до множества) с дополнительными заполненными записями. Эти шаги составляют `select_rendering_items`, вложенный цикл перебора состояний, когда они изменяются каждым компонентом по очереди.

Это реализуется в модуле `core.select_items`.

РИСУНОК 3

Прохождение структуры ADM для определения элементов `RenderingItems`



5.2.1 Отправная точка

Выбор элементов рендеринга может начинаться из нескольких точек в структуре ADM в зависимости от включенных в файл элементов.

Если присутствуют элементы *audioProgramme*, то выбирается один из них; или же, если присутствуют элементы *audioObject*, должны выбираться все элементы *audioObject*; в противном случае выбираются все элементы *audioTrackUID* (строки CHNA) (это называется режимом "только CHNA").

5.2.2 Выбор элемента audioProgramme

Выбирается только один элемент *audioProgramme*. Используемую программу может выбрать пользователь. Если ни одна звуковая программа не выбрана, должна выбираться программа с наименьшим идентификатором.

5.2.3 Выбор элементов audioContent

Выбираются все элементы *audioContent*, указанные в выбранном элементе *audioProgramme*.

5.2.4 Выбор элементов audioObject

Элементы `audioObject` должны пройти все возможные пути иерархии *audioObject* по очереди, начиная с выбранного элемента *audioContent* (по ссылкам *audioObject*).

5.2.5 Обработка дополнительных элементов *audioObject*

Ссылки *audioComplementaryObject* должны интерпретироваться как определяющие группы элементов *audioObject*, из которых будет воспроизводиться только один такой элемент.

Группа описывается ссылками *audioComplementaryObject* из элемента *audioObject* по умолчанию на все элементы *audioObject* этой группы, не являющиеся элементами по умолчанию. Пользователь может предоставить на выбор набор элементов *audioObject*, переопределяющих элементы по умолчанию. Исходя из этого определяется набор игнорируемых элементов *audioObject*, и если какие-либо элементы *audioObject*, встречающиеся на пути *audioObject*, присутствуют в этом наборе, их состояние сбрасывается.

5.2.5.1 Выбор дополнительных игнорируемых элементов *audioObject*

Прежде всего набор выбранных пользователем элементов *audioObject* должен дополняться элементами по умолчанию для каждой группы: для каждого корневого элемента *audioObject* (*audioObject* со ссылками *audioComplementaryObject*), если в группе, определенной корневым элементом *audioObject* этой группы, в наборе не присутствует ни один из элементов *audioObject*, должен добавляться корневой элемент *audioObject* (по умолчанию).

Затем набор игнорируемых элементов *audioObject* составляется из набора всех дополнительных элементов *audioObject* (то есть элементов *AudioObject* со ссылкой *audioComplementaryObject* и элементов *audioObject*, на которые указывает ссылка *audioComplementaryObject*) минус расширенный набор элементов *audioObject*, выбранных пользователем.

Если выбраны элементы *audioObject*, не принадлежащие к какой-либо дополнительной группе, или несколько элементов *audioObject* выбраны в одной группе *audioObject* (по ошибке пользователя либо в результате наложения групп), то возникает состояние ошибки.

5.2.6 Сопоставление элементов *audioPackFormat*

На следующем шаге должно производиться сопоставление информации в элементах *audioObject* (список элементов *audioPackFormat, audioTrackUID* и количества беззвучных дорожек или просто список всех элементов *audioTrackUID* в режиме "только CHNA") со структурами *audioPackFormat* и *audioChannelFormat*.

Это определяется как задача сопоставления/поиска, а не как особый путь через структуры ссылок, которые нужно разрешить, поскольку с обеих сторон имеется несколько элементов, которые для формирования правильного решения должны совпадать и не конфликтовать.

Совпадение считается действительным только в том случае, если найдено единственное решение. Если решения не найдены, то метаданные противоречивы и возникает состояние ошибки. Если найдено несколько решений, то метаданные неоднозначны и возникает состояние ошибки. В обоих случаях выполняется диагностика с отображением пользователю возможных причин ошибки.

5.2.6.1 Сопоставляемые пакеты

Спецификация элементов *audioPackFormat*, с которыми производится сопоставление, представлена в виде списка структур `AllocationPack`:

```
struct AllocationChannel {
  AudioChannelFormat channel_format;
  vector<AudioPackFormat> pack_formats;
};

struct AllocationPack {
  AudioPackFormat root_pack;
  vector<AllocationChannel> channels;
};
```

Каждая структура должна описывать корневой элемент *audioPackFormat* (`root_pack`, элемент *audioPackFormat* верхнего уровня, который указывает на все распределяемые каналы) и список каналов, с которыми производится сопоставление в этом пакете. Каждый канал представляет собой

комбинацию ссылки *audioChannelFormat* и списка возможных элементов *audioPackFormat*, с которыми может быть связан этот канал.

Для каждого пакета pack элементов *audioPackFormat*, когда *typeDefinition !=Matrix*, создается объект AllocationPack, где:

– root_pack – пакет pack;

– channels содержит по одной записи для каждого *audioChannelFormat*, доступного из pack (рекурсивное следование ссылкам *audioPackFormat*), где pack_formats содержит все элементы *audioPackFormat*, встречающиеся на пути от pack к *audioChannelFormat* (включая pack).

Хотя это некоторое упрощение структуры *audioPackFormat* и *audioChannelFormat*, преимуществом такого представления является его способность представлять структуры ссылок *audioPackFormat* и *audioChannelFormat*, используемых с содержимым *Matrix*, как описано ниже.

5.2.6.1.1 Обработка матрицы

На матрицу *audioPackFormat* можно ссылаться несколькими способами в зависимости от предполагаемого результата. Эти структуры ссылок отражены в следующих объектах AllocationPack, которые создаются для каждого пакета pack элементов audioPackFormat, для которого *typeDefinition==Matrix*.

– В случае если pack – это прямая или декодирующая матрица, то эта матрица должна применяться, если *audioObject* ссылается как на pack, так и на набор идентификаторов *audioTrackUID*, которые в свою очередь ссылаются на pack и каналы входных или кодированных выборок *audioPackFormat* из пакета pack:

  • root_pack – пакет pack;

  • channels содержит по одному значению для каждого канала *audioChannelFormat* входной выборки *audioPackFormat* пакета (либо *encodePackFormat*, либо *inputPackFormat* в зависимости от типа), где channel_format – это channel, а pack_formats – [pack].

– В случае если pack представляет собой прямую или декодирующую матрицу, то эту матрицу следует рассматривать как ранее применявшуюся к выборкам в файле, если *audioObject* ссылается как на pack, так и на набор идентификаторов *audioTrackUID*, которые в свою очередь ссылаются на пакет pack (или подпакеты) и каналы пакета pack:

  • root_pack – пакет pack;

  • channels содержит по одному значению на канал *audioChannelFormat* channel из пакета pack, где channel_format – канал, а pack_formats содержит все элементы *audioPackFormat*, встречающиеся на пути от pack к channel.

– В случае если pack является декодирующей матрицей, то может применяться ее *encodePackFormat*, за которым следует pack, если *audioObject* ссылается на pack и набор идентификаторов *audioTrackUID*, которые, в свою очередь, ссылаются на *encodePackFormat* и каналы *inputPackFormat* элемента *encodePackFormat*:

  • root_pack – пакет pack;

  • channels содержит одно значение на канал audioChannelFormat из inputPackFormat или encodePackFormat пакета, где channel_format – канал, а pack_formats содержит все элементы audioPackFormat, встречающиеся на пути от inputPackFormat до channel.

Тип матрицы *audioPackFormat* определяется по следующим правилам:

– если в ней имеются ссылки как *inputPackFormat*, так и *outputPackFormat*, то это прямая матрица;

– если в ней имеется ссылка *inputPackFormat*, но нет ссылки *outputPackFormat*, то это кодирующая матрица;

– если в ней имеется ссылка *outputPackFormat*, но нет ссылки *inputPackFormat*, это декодирующая матрица;

– если в ней нет ни ссылки *inputPackFormat*, ни ссылки *outputPackFormat*, возникает состояние ошибки.

5.2.6.2 Сопоставляемые дорожки и ссылки *audioPackFormat*

Дорожки, сопоставляемые с элементами `AllocationPack`, должны задаваться тремя значениями:

– `tracks` – список элементов `AllocationTrack`, каждый из которых представляет *audioTrackUID* (или строку CHNA):

```
class AllocationTrack {
  AudioChannelFormat channel_format;
  AudioPackFormat pack_format;
};
```

`channel_format` получается из *audioTrackUID* по ссылкам *audioTrackFormat*, *audioStreamFormat* и *audioChannelFormat*, тогда как `pack_format` указывается непосредственно идентификатором *audioTrackUID*;

– `pack_refs` – необязательный список ссылок *audioPackFormat*, присутствующих в *audioObject*;

– `num_silent_tracks` – количество беззвучных распределяемых дорожек, указанных в ссылках из *audioObject* на `ATU_00000000`.

При определении этих структур для *audioObject*:

– `tracks` содержит по одной записи для каждого (не беззвучного) *audioTrackUID*, указанного в *audioObject*;

– `pack_refs` – список ссылок *audioPackFormat*, содержащихся в *audioObject*;

– `num_silent_tracks` – количество указанных беззвучных *audioTrackUID* (соответствует ссылкам на `ATU_00000000` в *audioObject*).

В режиме "только CHNA":

– `tracks` содержит по одной записи на каждый *audioTrackUID* (или каждую строку CHNA) в файле;

– `pack_refs` имеет значение `None`;

– `num_silent_tracks` – 0.

5.2.6.3 Сопоставление

Результат сопоставления указывается в виде списка объектов `AllocatedPack`:

```
struct AllocatedPack {
  AllocationPack pack;
  vector<tuple<AllocationChannel,
               optional<AllocationTrack>>> allocation;
};
```

Каждый из них связывает каждый элемент *audioChannelFormat* из пакета `pack` с дорожкой `track` или беззвучной дорожкой, если `AllocationTrack` не указан.

Допустимое решение имеет следующие свойства.

1 Для каждого `AllocatedPack` каждый канал в `AllocationPack` встречается в `allocation` ровно один раз.

2 Каждая дорожка из `tracks` встречается на выходе ровно один раз.

3 Количество беззвучных дорожек, указанных на выходе, равно `num_silent_tracks`.

4 Для каждого канала `AllocationChannel` и соответствующей дорожки `AllocationTrack` элемент `track.channel_format` имеет значение `channel.channel_format`, а `track.pack_format` присутствует в `channel.pack_formats`.

5 Если список `pack_refs` не `None`, то для каждого пакета `AllocatedPack` имеется взаимно однозначное соответствие между `pack_refs` и значениями `pack.pack.root_pack`.

Решения, которые одинаковы, за исключением порядка элемента `AllocationPacks` или элементов `allocation` внутри него, считаются эквивалентными.

Можно использовать любой метод, позволяющий перебрать все действительные и уникальные (неэквивалентные) решения. В эталонной реализации решения находят, рассматривая вышеуказанные свойства как задачу удовлетворения ограничений и перебирая все решения с использованием поиска с возвратом.

5.2.6.3.1 Примеры

Сопоставление форматов в пакете иллюстрируется на следующих примерах.

Сначала определяются структуры, используемые в примерах. Символами `c1, c2` и т. д. и `p1, p2` и т. д. обозначены ссылки на объекты *audioChannelFormat* и *audioPackFormat* (но, возможно, и на любые другие объекты, поскольку в `allocate_packs` используется только информация из структур `Allocation...` путем сравнения этих ссылок по идентификаторам).

Монопакет и дорожка, указанная в нем:

```
ac1 = AllocationChannel(c1, [p1])
ap1 = AllocationPack(p1, [ac1])
at1 = AllocationTrack(c1, p1)
```

Двухканальный пакет с двумя парами указанных дорожек:

```
ac2 = AllocationChannel(c2, [p2])
ac3 = AllocationChannel(c3, [p2])
ap2 = AllocationPack(p2, [ac2, ac3])

at2 = AllocationTrack(c2, p2)
at3 = AllocationTrack(c3, p2)

at4 = AllocationTrack(c2, p2)
at5 = AllocationTrack(c3, p2)
```

Решение задачи для одной монодорожки в *audioObject* приводит к единственному решению, содержащему один распределенный пакет:

```
assert allocate_packs(
    packs=[ap1, ap2],
    tracks=[at1],
    pack_refs=[p1],
    num_silent_tracks=0,
) == [[AllocatedPack(pack=ap1, allocation=[(ac1, at1)])]]
```

Решение задачи для одной монодорожки в режиме "только CHNA" приводит к той же структуре:

```
assert allocate_packs(
    packs=[ap1, ap2],
    tracks=[at1],
    pack_refs=None,
    num_silent_tracks=0,
) == [[AllocatedPack(pack=ap1, allocation=[(ac1, at1)])]]
```

Решение задачи для одной беззвучной дорожки приводит к той же структуре, за исключением того, что ссылка на дорожку заменена на `None`:

```
assert allocate_packs(
    packs=[ap1, ap2],
    tracks=[],
```

```
        pack_refs=[p1],
        num_silent_tracks=1,
    ) == [[AllocatedPack(pack=ap1, allocation=[(ac1, None)])]]
```

Если дорожек больше, чем каналов в ссылках пакета, то решений не будет, поскольку правило 2 вступает в конфликт с правилом 5:

```
    assert allocate_packs(
        packs=[ap1, ap2],
        tracks=[at1],
        pack_refs=[],
        num_silent_tracks=0,
    ) == []
```

Если беззвучных дорожек больше, чем каналов в ссылках пакета, то решений не будет, поскольку правило 2 вступает в конфликт с правилом 5:

```
    assert allocate_packs(
        packs=[ap1, ap2],
        tracks=[],
        pack_refs=[ap1],
        num_silent_tracks=2,
    ) == []
```

Если имеет место несоответствие между ссылками пакета и информацией о каналах/пакетах в дорожках, решений не будет, поскольку правила 1, 4 и 5 конфликтуют между собой:

```
    assert allocate_packs(
        packs=[ap1, ap2],
        tracks=[at1, at1],
        pack_refs=[p2],
        num_silent_tracks=0,
    ) == []
```

Если в *audioObject* имеется несколько экземпляров многоканального пакета, то назначение дорожек пакетам неоднозначно, так что существует несколько решений:

```
    assert allocate_packs(
        packs=[ap1, ap2],
        tracks=[at2, at3, at4, at5],
        pack_refs=[p2, p2],
        num_silent_tracks=0,
    ) == [
        [AllocatedPack(pack=ap2, allocation=[(ac2, at2), (ac3, at3)]),
         AllocatedPack(pack=ap2, allocation=[(ac2, at4), (ac3, at5)])],
        [AllocatedPack(pack=ap2, allocation=[(ac2, at2), (ac3, at5)]),
         AllocatedPack(pack=ap2, allocation=[(ac2, at4), (ac3, at3)])],
    ]
```

5.2.6.4 Последующая обработка решения

Следует отметить, что результаты сопоставления даны в виде входных структур (`AllocationPack`, `AllocationChannel`, `AllocationTrack`), а не лежащих в их основе ссылок на структуры ADM. Это позволяет произвольно устанавливать соответствие между ссылками *audioPackFormat* и *audioChannelFormat* (в *audioObject* и *audioTrackUID*) и информацией, предоставляемой рендереру, поскольку при использовании *typeDefinition*==*Matrix* простого соответствия не существует.

Для нематричного пакета `AllocatedPack` имеет место простое соответствие. `output_pack` – это `pack.pack.root_pack`, и существует взаимно однозначное соответствие между распределениями в `pack.allocation` и реальным распределением каналов: каналу `AllocationChannel` ставится в соответствие `channel.channel_format`, дорожке `AllocationTrack` – `DirectTrackSpec` с индексом дорожки *audioTrackUID* (или строкой CHNA), связанным с этой дорожкой, а отсутствующему распределению `AllocationTrack` ставится в соответствие `SilentTrackSpec`.

Для матричного пакета AllocatedPack требуется более сложное отображение:

pack.root_pack – это всегда декодируемый или прямой пакет (см. пункт 5.2.6.1.1), поэтому output_pack – это pack.root_pack.outputPackFormat.

Выходной канал распределения дорожек содержит по одной записи на каждый *audioChannelFormat* matrix_channel в root_pack. Эти каналы находятся во взаимно однозначном соответствии с элементами *audioChannelFormat* в output_pack, заданными ссылками *outputChannelFormat*.

audioChannelFormat – это matrix_channel.block_formats[0].outputChannelFormat.

TrackSpec создается путем рекурсивного следования по ссылкам *inputChannelFormat* из matrix_channel на элементы *audioChannelFormat*, указанные в pack.allocation, вложения структур MatrixcoefficientTrackSpec и MixTrackSpec для выполнения обработки, указанной в элементах *coefficient*, и микширования нескольких входных каналов.

– Если в pack.allocation есть ссылка на matrix_channel, выдается DirectTrackSpec или SilentTrackSpec, соответствующие связанному с ними AllocationTrack (см. выше).
– В противном случае выдается MixTrackSpec, содержащий по одному MatrixcoefficientTrackSpec на каждый элемент *coefficient* c из matrix_channel.block_formats[0].matrix с применением обработки, указанной в c, к спецификации рекурсивно определенной дорожки c.inputChannelFormat.

В эталонной реализации это реализовано в двух подклассах AllocationPack, содержащих методы для запроса *audioPackFormat* и распределения каналов рендереру. Аналогичным образом поддерживается связь между AllocationTrack и соответствующими идентификаторами *audioTrackUID* с использованием подкласса AllocationTrack.

5.2.7 Выходные элементы рендеринга

После определения корневого элемента *audioPackFormat* и присвоения TrackSpec каждому из его каналов вся найденная информация преобразуется в один или несколько элементов RenderingItem.

Этот процесс зависит от типа корневого элемента *audioPackFormat*.

5.2.7.1 Общие компоненты

Некоторые данные в элементах рендеринга совместно используются разными типами и, следовательно, также получаются одинаковым способом.

5.2.7.1.1 Важность

Объект ImportanceData должен определяться на основе состояния выбора элемента со следующими значениями:

– audio_object – минимальная важность, указанная во всех элементах *audioObject* вдоль пути;
– audio_pack_format – минимальная важность, указанная в любом элементе *audioPackFormat* на пути от корневого элемента *audioPackFormat* до *audioChannelFormat*.

В обоих случаях None (важность не задана) определяется как наивысшая важность.

5.2.7.1.2 Дополнительные данные

Объект ExtraData должен определяться на основе состояния выбора элемента со следующими значениями:

– object_start – время начала следования последнего объекта *audioObject* по пути (None в режиме "только CHNA");
– object_duration – продолжительность следования последнего объекта *audioObject* по пути (None в режиме "только CHNA");

– `reference_screen` – параметр *audioProgrammeReferenceScreen* выбранной программы *audioProgramme* (`None`, если не выбрана);

– `channel_frequency` – это элемент *frequency* выбранного элемента *audioChannelFormat* (или `None`, если он не выбран, как при создании элемента рендеринга HOA).

5.2.7.2 Выходные элементы рендеринга для typeDefinition==Objects или DirectSpeakers

Процесс определения элементов рендеринга для *Objects* и *DirectSpeakers* аналогичен – различаются только используемые типы и выбор параметров.

Создается один элемент рендеринга для каждой пары *audioChannelFormat* и `track_spec` в распределении каналов.

Создается объект `MetadataSource`, который вырабатывает по одному элементу `RenderingItem` (соответствующего типа) для каждого *audioBlockFormat* из выбранного *audioChannelFormat*, в котором поле `extra_data` определяется, как указано выше, а поле `audioPackFormats` содержит все элементы *audioPackFormat*, встречающиеся на пути между корневым элементом *audioPackFormat* и элементом *audioChannelFormat*. Он помещается в объект `RenderingItem` (опять же соответствующего типа) с определенными выше параметрами `track_spec` и `importance`.

5.2.7.3 Выходные элементы рендеринга для *typeDefinition==HOA*

Создается по одному элементу `HOARenderingItem` на каждый распределенный пакет, содержащему всю информацию, которая необходима для рендеринга группы каналов, составляющих поток HOA. Эта информация распространяется по нескольким элементам *audioChannelFormat* и *audioPackFormat* (когда они вложены), которые должны быть согласованы.

Элементы *audioChannelFormat* HOA должны содержать только один элемент *audioBlockFormat*; в противном случае возникает состояние ошибки.

Создается один объект `HOATypeMetadata` с параметрами, полученными в соответствии с таблицей 1.

ТАБЛИЦА 1

Свойства параметров `HOATypeMetadata`

Параметр `HOATypeMetadata`	Параметр *audioBlockFormat*	Параметр *audioPackFormat*	Количество
`rtime`	*rtime*		Один
`duration`	*duration*		Один
`orders`	*order*		На канал
`degrees`	*order*		На канал
`normalization`	*normalization*	*normalization*	Один
`nfcRefDist`	*nfcRefDist*	*nfcRefDist*	Один
`screenRef`	*screenRef*	*screenRef*	Один

Все параметры сначала должны определяться для каждого *audioChannelFormat* в корневом элементе *audioPackFormat*. Для параметров, содержащих параметры как *audioBlockFormat*, так и *audioPackFormat*, этот параметр можно настроить на один *audioBlockFormat* в *audioChannelFormat* или на любой *audioPackFormat*, встречающийся на пути от корневого *audioPackFormat* до *audioChannelFormat*. Если для данного *audioChannelFormat* выявлены несколько копий какого-либо параметра, они должны иметь одно и то же значение, в противном случае возникает состояние ошибки. Если значения данного параметра и *audioChannelFormat* не установлены, применяется значение по умолчанию, указанное в Рекомендации МСЭ-R BS.2076-1.

После обнаружения параметра *nfcRefDist* конкретного элемента *audioChannelFormat* значение "0" должно преобразоваться в `None`, что означает, что NFC не должно применяться. Это выполняется на данном этапе (а не в ходе анализа XML), так что, например, *nfcRefDist==0.0* считается конфликтующим с *nfcRefDist==1.0*.

Для параметров, имеющих только одно значение (всех, кроме `orders` и `degrees`), параметры, определенные для всех элементов *audioChannelFormat*, должны быть равны, в противном случае возникает состояние ошибки.

`extra_data` определяется, как указано выше, для всего набора *audioPackFormat*.

`HOARenderingItem` должен создаваться с одной записью в `track_specs` и `importances` на каждый элемент в распределении каналов (как описано выше) и `MetadataSource`, содержащем только вышеуказанный объект `HOATypeMetadata`.

5.3 Обработка элементов рендеринга

Некоторые функциональные возможности рендерера реализуются путем изменения списка выбранных элементов рендеринга. В пункте 5.3.1 описано, как можно удалять контент в зависимости от заданного уровня важности, а в пункте 5.3.3 – как можно эмулировать эффекты преобразования метаданных в нисходящем направлении.

5.3.1 Эмуляция важности

Параметры *importance*, определенные в Рекомендации МСЭ-R BS.2076-1, позволяют рендереру отбрасывать элементы, важность которых ниже определенного уровня по еще не установленным причинам, зависящим от приложения.

ADM определяет три разных параметра *importance*, которые следует использовать:

– *importance* как атрибут audioObject;

– *importance* как атрибут audioPackFormat;

– *importance* как атрибут audioBlockFormat для typeDefinition==Object.

Главное различие между этими атрибутами *importance* состоит в том, что важность *audioBlockFormat* зависит от времени, то есть может изменяться со временем, а важность *audioObject* и *audioPackFormat* остается постоянной.

Для каждого атрибута *importance* может использоваться отдельный порог. Считается, что определение требуемых пороговых значений существенно зависит от приложения и сценария использования и, следовательно, оно выходит за рамки спецификации производственного рендерера. Вместо этого рендерер обеспечивает возможность моделирования влияния применения заданного порога важности к ADM. Это позволяет производителям контента исследовать влияние использования тех или иных значений параметра *importance* на рендеринг. Следовательно, эмуляция параметров *importance* не является частью фактического процесса рендеринга, а применяется как шаг постобработки объектов `RenderingItem`.

5.3.1.1 Значения важности `RenderingItem`

Каждый элемент рендеринга может иметь собственный набор действительных значений параметра *importance*, потому что элементы *audioObject* и *audioPackFormat* могут быть вложенными. Таким образом для каждого `RenderingItem` учитываются все указанные элементы *audioObject* и *audioPackFormat*, участвующие в определении этого `RenderingItem`.

Применяются следующие правила.

– Если *audioObject* имеет значение *importance* ниже порогового, все указанные элементы *audioObject* также должны отбрасываться. Для этого в качестве параметра *importance* элемента *audioObject* для данного `RenderingItem` должно использоваться наименьшее значение *importance* среди всех объектов *audioObject*, приводящих к определению `RenderingItem`.

– Если значение *importance* элемента *audioPackFormat* ниже порогового, то все указанные элементы *audioPackFormat* также должны отбрасываться. Для этого в качестве параметра *importance* элемента *audioPackFormat* для данного `RenderingItem` должно использоваться наименьшее значение *importance* среди всех элементов *audioPackFormat*, приводящих к определению `RenderingItem`.

– При определении значения *importance* объекта `RenderingItem` элемент *audioObject*, не имеющий значения *importance*, не должен учитываться.

– При определении значения *importance* объекта `RenderingItem` элемент *audioPackFormat*, не имеющий значения *importance*, не должен учитываться.

Это реализуется в модуле `fileio.utils.RenderingItemHandler`.

5.3.1.2 Обработка постоянной важности

При наличии `RenderingItem` с параметром `ImportanceData` элемент необходимо исключить из списка элементов для рендеринга, если любое постоянное значение важности (*audioObject*, *audioPackFormat*) ниже соответствующего определяемого пользователем порога:

$$\begin{aligned} &\texttt{importance.audio\_object} &&< \texttt{audio\_object\_threshold} \\ \vee\ &\texttt{importance.audio\_pack\_format} &&< \texttt{audio\_pack\_format\_threshold}. \end{aligned}$$

Это реализуется в модулях `core.importance.filter_audioObject_by_importance` и `core.importance.filter_audioPackFormat_by_importance`.

5.3.1.3 Обработка изменяющейся со временем важности

Обработка важности на уровне элемента *audioBlockFormat* (*typeDefinition==Object*) не может быть выполнена путем фильтрации объектов `RenderingItem`, поскольку этот элемент может находиться ниже порогового значения лишь в течение некоторого времени. Чтобы эмулировать отбрасывание элементов рендеринга в данном конкретном случае, `RenderingItem` необходимо эффективно заглушить на время *audioBlockFormat*. В данном контексте "заглушение *audioBlockFormat*" эквивалентно принятию `bf.gain` равным нулю для `bf` *audioBlockFormat*.

Это реализуется в модуле `core.importance.MetadataSourceImportanceFilter`.

5.3.2 Эмуляция преобразования

К элементам рендеринга может применяться эмуляция преобразования метаданных. Эмуляция преобразования может быть отключена или настроена на преобразование метаданных в форму полярных или декартовых координат.

Если эмуляция преобразования включена, выбирается соответствующая функция из раздела 10 и применяется ко всем элементам *audioBlockFormat* с атрибутом *typeDefinition==Objects* в выбранных элементах рендеринга.

6 Общие компоненты рендерера

В данном разделе содержится описание компонентов, совместно используемых субрендерерами для различных объектов typeDefinitions.

6.1 Точечный панораматор полярных координат

Ядро рендерера составляет компонент точечного панораматора; по информации о расположении громкоговорителей и направлении в трехмерном пространстве он вырабатывает коэффициенты усиления для громкоговорителей (по одному на каждый громкоговоритель), которые, будучи примененными к монофоническому/цифровому сигналу, воспроизводимому громкоговорителями, создают у слушателя впечатление звука, доносящегося из нужного направления.

Точечный панораматор используется во всех компонентах рендерера для рендеринга точечных источников, указанных метаданными объекта, а также в рамках системы пространственного рендеринга в качестве запасного варианта для рендерера *DirectSpeakers* и в процессе проектирования декодера HOA.

Точечный панораматор в этом рендерере основан на представлении VBAP [2] с несколькими усовершенствованиями, которые делают его более подходящим для использования в широковещательных системах.

– В дополнение к триплетам громкоговорителей, как в VBAP, точечный панораматор поддерживает элементарные четверки громкоговорителей. Это решает те же проблемы, что и использование виртуальных громкоговорителей в других системах, но приводит к более гладкой общей функции панорамирования.
– Триангуляция расположения громкоговорителей выполняется в номинальных положениях громкоговорителей и деформируется в соответствии с их реальным расположением, что гарантирует, что характеристики панорамирования всегда соответствует адаптациям данного расположения.
– В некоторых ситуациях для изменения рендеринга используются виртуальные громкоговорители и понижающее микширование в целях коррекции наблюдаемых эффектов восприятия и создания желаемого поведения в системах с разнесенными громкоговорителями.
– Чтобы не усложнять конструкцию ради крайне редко используемых расположений громкоговорителей, схема расположения 0+2+0 рассматривается как особый случай.

6.1.1 Архитектура

Точечный панораматор содержит список объектов с интерфейсом `RegionHandler`; каждый объект/область должен/должна отвечать за коэффициенты усиления громкоговорителя на заданном участке пространства.

Чтобы получить коэффициенты усиления для заданного направления, точечный панораматор должен опросить по очереди каждую область, которая возвращает либо вектор усиления, если она может обработать данное направление, либо нулевой результат, если не может; используется вектор усиления первой найденной области, способной обработать данное направление.

В любом действительном точечном панораматоре выполняются следующие два условия.

– Любое заданное направление может быть обработано по крайней мере одной областью.
– Все области, способные обработать данное направление, дают аналогичные коэффициенты усиления (в пределах некоторого допуска).
– В пределах любой области полученные коэффициенты усиления плавно изменяются относительно желаемого направления.

Эти свойства в совокупности гарантируют, что коэффициенты усиления, полученные от точечного панораматора, четко определены для всех направлений и всегда плавно изменяются относительно направления в пределах некоторого допуска.

В следующих подразделах описываются имеющиеся типы интерфейса `RegionHandler` и процесс настройки, используемый для создания списка областей для данной схемы расположения громкоговорителей.

Это поведение реализуется в модуле `core.point_source.PointSourcePanner`.

Кроме того, реализуется класс `PointSourcePannerDownmix` с тем же интерфейсом. Когда запрашивается определенное положение, он вызывает другой класс `PointSourcePanner`, чтобы получить вектор усиления, к которому он применяет матрицу понижающего микширования и нормализацию мощности. Это используется в пункте 6.1.3.1 для переопределения виртуальных громкоговорителей.

6.1.2 Типы областей

Большинство областей вырабатывает коэффициенты усиления для поднабора выходных каналов; отображение этого поднабора каналов на весь вектор каналов реализуется в модуле `core.point_source.RegionHandler.handle_remap`.

6.1.2.1 Триплет

Это область сферического треугольника, образованного тремя громкоговорителями, реализующими базовое VBAP.

Данная область должна инициализироваться трехмерными позициями трех громкоговорителей:

$$\mathbf{P} = [\mathbf{p}_1,\ \mathbf{p}_2,\ \mathbf{p}_3]^T.$$

Три выходных коэффициента усиления $\mathbf{g}$ для данного направления *D* таковы, что:

– $\mathbf{g} \cdot \mathbf{P} = s\mathbf{d}$ для некоторых $s > 0$ с некоторым малым допуском;

– $g_i \geq 0\ \forall\ i \in \{1,2,3\}$;

– $\| \mathbf{g} \|_2 = 1$.

Этот тип интерфейса `RegionHandler` реализован в модуле `core.point_source.Triplet`.

6.1.2.2 Область VirtualNgon

Это область, образованная *n* реальными громкоговорителями, которая разбита на треугольники с добавлением одного виртуального громкоговорителя. Каждый треугольник состоит из двух смежных реальных громкоговорителей и виртуального громкоговорителя, который микшируется с реальными громкоговорителями с помощью заданных коэффициентов понижающего микширования.

Например, если используются четыре реальные позиции громкоговорителей $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4\}$ и одна виртуальная позиция $\mathbf{p}_v$, то будут созданы следующие треугольники:

– $\{\mathbf{p}_v, \mathbf{p}_1, \mathbf{p}_2\}$;

– $\{\mathbf{p}_v, \mathbf{p}_2, \mathbf{p}_3\}$;

– $\{\mathbf{p}_v, \mathbf{p}_3, \mathbf{p}_4\}$;

– $\{\mathbf{p}_v, \mathbf{p}_4, \mathbf{p}_1\}$.

Когда у интерфейса `RegionHandler` этого типа запрашивается позиция, необходимо перебрать по очереди каждый треугольник, пока один из них не возвратит действительные коэффициенты усиления, так же как для точечных панораматоров верхнего уровня. Это приводит к вектору из *n* коэффициентов усиления для реальных громкоговорителей $\mathbf{g} = \{g_1, \ldots, g_n\}$ и коэффициента усиления для виртуального громкоговорителя g_v, который микшируется с понижением до реальных громкоговорителей с помощью заданных коэффициентов понижающего микширования $\mathbf{w}_{\mathrm{dmx}}$:

$$\mathbf{g}' = \mathbf{g} + \mathbf{W}_{\mathrm{dmx}} \quad g_v.$$

Наконец, результат нормализуется по мощности, что приводит к окончательным коэффициентам усиления:

$$\mathbf{g}'' = \frac{\mathbf{g}'}{\|\mathbf{g}'\|_2}.$$

Этот тип интерфейса `RegionHandler` реализуется в модуле `core.point_source.VirtualNgon`.

6.1.2.3 Область QuadRegion

Это область сферического четырехугольника, образованного четырьмя громкоговорителями.

Коэффициенты усиления рассчитываются для каждого громкоговорителя сначала путем разбивки позиции на две составляющие – x и y. Составляющую x можно рассматривать как положение по горизонтали внутри четырехугольника, равное 0 на левой стороне и 1 на правой, а составляющую y – как положение по вертикали, равное 0 на нижней стороне и 1 на верхней.

Значения x и y преобразуются в коэффициенты усиления для каждого громкоговорителя с использованием уравнений (1) и (2). Значения x и y (и, следовательно, коэффициенты усиления для громкоговорителей), которые приводят к заданному вектору скорости, можно определить путем решения уравнений (1) – (3).

По сложности эта задача одинакова с VBAP, и в результате ее решения получают те же коэффициенты усиления на сторонах четырехугольника, что и в случае VBAP, что позволяет использовать ее с интерфейсами `RegionHandler` других типов в одном точечном панораматоре по правилам, указанным в пункте 6.1.1.

Полученные коэффициенты усиления бесконечно дифференцируемы по отношению к положению внутри области, что дает результаты, сопоставимые с попарным панорамированием между виртуальными громкоговорителями в обычных ситуациях.

Этот тип интерфейса `RegionHandler` реализуется в модуле `core.point_source.QuadRegion`.

6.1.2.3.1 Постановка задачи

С учетом положения четырех громкоговорителей в декартовых координатах $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4]$ в направлении против часовой стрелки с позиции слушателя вычисляется вектор усиления $\mathbf{g}$ для направления на источник $\mathbf{d}$:

$$\mathbf{g}' = [(1-x)(1-y),\ x(1-y),\ xy,\ (1-x)y]; \tag{1}$$

$$\mathbf{g} = \frac{\mathbf{g}'}{\|\mathbf{g}'\|_2}, \tag{2}$$

где x и y выбраны таким образом, чтобы вектор скорости $\mathbf{g} \cdot \mathbf{P}$ имел желаемое направление $\mathbf{d}$. Величина вектора скорости r не имеет значения, поскольку коэффициенты усиления нормированы по мощности:

$$\mathbf{g} \cdot \mathbf{P} = r\mathbf{d} \tag{3}$$

для некоторого $r > 0$.

6.1.2.3.2 Решение

При заданном значении x все векторы скорости $\mathbf{d}$ с этим значением x находятся на плоскости, образованной началом координат и двумя точками на некотором расстоянии вдоль верхней и нижней сторон четырехугольника:

$$(1-x)\mathbf{p}_1 + x\mathbf{p}_2;$$
$$(1-x)\mathbf{p}_4 + x\mathbf{p}_3.$$

Таким образом:

$$(((1-x)\mathbf{p}_1 + x\mathbf{p}_2) \times ((1-x)\mathbf{p}_4 + x\mathbf{p}_3)) \cdot \mathbf{d} = 0. \tag{4}$$

Это уравнение можно решить, чтобы найти *x* для данного направления на источник $\mathbf{d}$.

Соберем члены x:

$$[(\mathbf{p}_1 + x(\mathbf{p}_2 - \mathbf{p}_1)) \times (\mathbf{p}_4 + x(\mathbf{p}_3 - \mathbf{p}_4))] \cdot \mathbf{d} = 0.$$

Развернем векторное произведение и соберем члены:

$$\begin{aligned}&[(\mathbf{p}_1 \times \mathbf{p}_4)\\ &+x\left((\mathbf{p}_1 \times (\mathbf{p}_3 - \mathbf{p}_4)) + ((\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{p}_4)\right)\\ &+x^2\ ((\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_4))\\ &] \cdot \mathbf{d} = 0.\end{aligned}$$

Наконец, умножим на **D**:

$$\begin{aligned} & [(\mathbf{p}_1 \times \mathbf{p}_4) \cdot \mathbf{d}] \\ +x \quad & [((\mathbf{p}_1 \times (\mathbf{p}_3 - \mathbf{p}_4)) + ((\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{p}_4)) \cdot \mathbf{d}] \\ +x^2 \quad & [((\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_4)) \cdot \mathbf{d}] \\ & = 0. \end{aligned}$$

Следовательно, решением для x будет корень многочлена, который можно вычислить стандартными методами.

Заменив в вышеприведенных уравнениях **P** на $\mathbf{P}'$, можно также определить y:

$$\mathbf{P}' = [\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4, \mathbf{p}_1].$$

Затем, используя уравнения (1) и (2), можно вычислить коэффициент усиления **g**. Поскольку в уравнении (4) масштаб **d** игнорируется, можно найти решения, которые дают вектор скорости, прямо противоположный желаемому. Это можно проверить, убедившись, что

$$\mathbf{gP} \cdot \mathbf{d} > 0.$$

6.1.2.4 Метод StereoPanDownmix

Выходные сигналы точечного источника для стереосистемы (0+2+0) определяются методом, основанным на понижающем микшировании с 0+5+0 до 0+2+0. Этот метод реализуется отдельно.

Используется следующая процедура.

– Направление входного сигнала панорамируется с использованием точечного панораматора, настроенного для системы 0+5+0, чтобы получить вектор из пяти коэффициентов усиления $\mathbf{g}'$ в следующем порядке: M+030, M–030, M+000, M+110, M–110.

– Для получения стереофонических коэффициентов усиления $\mathbf{G}''$ в порядке M+030, M–030 применяется матрица преобразования формата из 0+5+0 в 0+2+0:

$$\mathbf{g}'' = \begin{bmatrix} 1 & 0 & \sqrt{\frac{1}{3}} & \sqrt{\frac{1}{2}} & 0 \\ 0 & 1 & \sqrt{\frac{1}{3}} & 0 & \sqrt{\frac{1}{2}} \end{bmatrix} \cdot \mathbf{g}'.$$

– Нормализация мощности доводит $\mathbf{g}''$ до значения, определяемого балансом между передними и задними громкоговорителями в $\mathbf{g}'$, так что источники, расположенные между позициями M+030 и M–030, не ослабляются, тогда как источники, расположенные между позициями M−110 и M+110, ослабляются на 3 дБ:

$$\begin{aligned} a_{\text{front}} &= \max\{g'_1, g'_2, g'_3\}; \\ a_{\text{rear}} &= \max\{g'_4, g'_5\}; \\ r &= \frac{a_{\text{rear}}}{a_{\text{front}} + a_{\text{rear}}}; \\ \mathbf{g} &= \mathbf{g}'' \frac{1\frac{r}{2}}{\|\mathbf{g}''\|_2}. \end{aligned}$$

Этот тип интерфейса `RegionHandler` реализуется в модуле `core.point_source.StereoPanDownmix`.

ПРИМЕЧАНИЕ. – Матрица преобразования **g** из (0+5+0) в (0+2+0) полностью совмещается с коэффициентами понижающего микширования, указанными в Рекомендации МСЭ-R BS.775, следующим образом:

$$\mathbf{g} = \begin{bmatrix} 1 & 0 & \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} & 0 \\ 0 & 1 & \sqrt{\frac{1}{2}} & 0 & \sqrt{\frac{1}{2}} \end{bmatrix}.$$

6.1.3 Процесс настройки

В процессе настройки создается точечный панораматор, содержащий вышеупомянутые типы интерфейса `RegionHandler` для данного расположения громкоговорителей. Процесс настройки принимает объект `Layout` (определенный в пункте 11.1.3) и создает объект `PointSourcePanner`.

Сначала процесс настройки выбирает поведение, определяемое атрибутом `Layout::name`. Если значение атрибута `Layout::name` равно 0+2+0, то настройка производится специальной функцией выбора конфигурации для стереосистемы, описанной в пункте 6.1.3.2. Во всех остальных случаях настройка осуществляется общей функцией, описанной в пункте 6.1.3.1.

Процесс настройки выполняется в модуле `core.point_source.configure`.

6.1.3.1 Процесс для обычных схем расположения громкоговорителей

Чтобы настроить `PointSourcePanner` для обычных схем расположения громкоговорителей, используется следующий процесс.

1 Обновить азимут номинальных положений громкоговорителей с меткой M+SC или M–SC, чтобы обеспечить правильную триангуляцию с широко разнесенными экранными громкоговорителями. Если действительный азимут (`polar_position.azimuth`) равен φ, то номинальный азимут φ_n (`polar_nominal_position.azimuth`) равен

$$\varphi_n = \mathrm{sgn}(\varphi) \times \begin{cases} 45 & |\varphi| > 30; \\ 15 & \text{в других случаях.} \end{cases}$$

2 Определить набор переопределенных виртуальных громкоговорителей, как описано ниже. Эти громкоговорители добавляются в набор громкоговорителей при какой-либо схеме расположения, так что их можно рассматривать как реальные.

3 Создать два списка позиций громкоговорителей в нормализованных декартовых координатах, которые будут использоваться на следующих шагах; один список содержит номинальные положения громкоговорителей (для триангуляции расположения громкоговорителей), а другой – реальные (для использования при создании областей). Номинальные положения громкоговорителей – это положения, указанные в Рекомендации МСЭ-R BS.2051-2, а реальные – положения, которые фактически используются в рассматриваемой системе воспроизведения.

4 Добавить к каждому списку позиций громкоговорителей один или два виртуальных громкоговорителя, которые становятся виртуальными громкоговорителями в центре многоугольника `VirtualNgon`:

- позиция $0{,}0{,}-1$ (ниже слушателя) добавляется всегда, поскольку ни в одной из схем расположения громкоговорителей, определенных в Рекомендации МСЭ-R BS.2051-2, нет громкоговорителя в этой позиции;
- позиция $0{,}0{,}1$ (выше слушателя) добавляется, если в схеме расположения нет громкоговорителя с меткой `T+000` или `UH+180`. Причина, по которой этот громкоговоритель не используется при наличии громкоговорителя `UH+180`, заключается в том, что когда он используется в схеме расположения `3+7+0`, определенной в Рекомендации МСЭ-R BS.2051-2, эта позиция может совпадать с положением виртуального громкоговорителя, создавая ступенчатый переход в функции панорамирования.

5 Составить выпуклую оболочку из номинальных положений громкоговорителей. Если этот алгоритм реализован с использованием арифметики с плавающей запятой, то ошибки могут привести к разделению некоторых граней выпуклой оболочки – грани объединяются в пределах допусков, так что результат получается таким же, как если бы алгоритм был реализован с использованием точной арифметики.

6 Создать `PointSourcePannerDownmix` со следующими областями.

- Для каждой грани выпуклой оболочки, которая не содержит ни одного из виртуальных громкоговорителей, добавленных на шаге 3:
  - если грань имеет три ребра, создать область `Triplet` из реальных позиций громкоговорителей, соответствующих вершинам грани;
  - если грань имеет четыре ребра, создать область `QuadRegion` из реальных позиций громкоговорителей, соответствующих вершинам грани.
- Для каждого виртуального громкоговорителя, добавленного на шаге 3, создать область `VirtualNgon` с реальными позициями соседних громкоговорителей (все громкоговорители, имеющие общую грань выпуклой оболочки с виртуальным громкоговорителем) по краям, позицией виртуального громкоговорителя в центре и всеми коэффициентами понижающего микширования, установленными в $\frac{1}{\sqrt{n}}$, где n – число соседних громкоговорителей.

  Следует отметить, что никакие схемы расположения, определенные в Рекомендации МСЭ-R BS.2051-2, не приводят к граням более чем с четырьмя ребрами.

  Коэффициенты понижающего микширования отображают виртуальные громкоговорители на физические громкоговорители, как описано ниже.

Это реализуется в модуле `core.point_source._configure_full`.

6.1.3.1.1 Определение виртуальных громкоговорителей с прямым понижающим микшированием

Для каждого громкоговорителя, расположенного на среднем уровне, при отсутствии на верхнем или нижнем уровнях в этой области реальных громкоговорителей на эти уровни добавляются виртуальные громкоговорители с тем же азимутом, что и у реального громкоговорителя. Эти виртуальные громкоговорители должны иметь коэффициенты понижающего микширования, которые отображают их выходной сигнал непосредственно на соответствующий громкоговоритель среднего уровня.

Как и в случае с реальными громкоговорителями, виртуальные громкоговорители имеют как реальное, так и номинальное положение, причем реальное положение определяется по реальным положениям реальных громкоговорителей, а номинальное – по номинальным положениям реальных громкоговорителей. Включение или невключение виртуального громкоговорителя зависит от номинальных положений реальных громкоговорителей, поэтому для данной схемы расположения всегда используется один и тот же набор виртуальных громкоговорителей.

Чтобы определить набор виртуальных громкоговорителей для данной схемы расположения, используется следующая процедура.

– Для каждого $i \in [1, N]$, где $N = \mathtt{len(layouts.channels)}$ – число каналов, определить:

$$\varphi_{i,r} = \mathtt{layouts.channels[i].polar\_position.azimuth}$$
$$\varphi_{i,n} = \mathtt{layouts.channels[i].polar\_nominal\_position.azimuth}$$
$$\theta_{i,r} = \mathtt{layouts.channels[i].polar\_position.elevation}$$
$$\theta_{i,n} = \mathtt{layouts.channels[i].polar\_nominal\_position.elevation}$$

– Определить три набора индексов каналов, идентифицирующих каналы на верхнем, среднем и нижнем уровнях схемы расположения:

$$S_u = \{ i \mid 30° \le \theta_{i,n} \le 70° \};$$
$$S_m = \{ i \mid -10° \le \theta_{i,n} \le 10° ;$$
$$S_l = \{ i \mid -70° \le \theta_{i,n} \le -30° \}.$$

– Виртуальные громкоговорители имеют те же номинальные и реальные азимуты, что и соответствующие реальные громкоговорители. Реальный угол места – это средний угол места реальных громкоговорителей данного уровня, если таковые имеются, или же –30° либо 30°

для нижнего и верхнего уровней. Номинальный угол места всегда составляет –30° или 30° для нижнего и верхнего уровней.

Определить два номинальных угла места:

$$\theta'_{u,n} = 30°;$$

$$\theta'_{l,n} = -30°.$$

Определить два реальных угла места:

$$\theta'_{u,r} = \begin{cases} 30° & |S_u| = 0; \\ \frac{\sum_{j \in S_u} \varphi_{j,r}}{|S_u|} & \text{в других случаях;} \end{cases}$$

$$\theta'_{l,r} = \begin{cases} 30° & |S_u| = 0; \\ \frac{\sum_{j \in S_l} \varphi_{j,r}}{|S_l|} & \text{в других случаях.} \end{cases}$$

– Громкоговорители создаются на каком-либо уровне только в том случае, если абсолютный номинальный азимут соответствующего громкоговорителя среднего уровня больше или равен максимальному абсолютному номинальному азимуту реальных громкоговорителей на этом уровне плюс 40°. Эти пределы азимута определяются следующим образом:

$$L_u = \begin{cases} 0 & |S_u| = 0; \\ \max\limits_{j \in S_u} |\varphi_{j,n}| + 40° & \text{в других случаях;} \end{cases}$$

$$L_l = \begin{cases} 0 & |S_l| = 0; \\ \max\limits_{j \in S_l} |\varphi_{j,n}| + 40° & \text{в других случаях.} \end{cases}$$

– Для каждого j в S_m:

• создать виртуальный верхний громкоговоритель $\varphi_{j,n} \geq L_u$, как указано атрибутом `channel` структуры `Channel`, со следующими значениями:

$$\begin{aligned} \texttt{channel.polar\_position.azimuth} &= \varphi_{j,r}; \\ \texttt{channel.polar\_position.elevation} &= \theta'_{u,r}; \\ \texttt{channel.polar\_nominal\_position.azimuth} &= \varphi_{j,n}; \\ \texttt{channel.polar\_nominal\_position.elevation} &= \theta'_{u,n}; \end{aligned}$$

• создать виртуальный нижний громкоговоритель, если $\varphi_{j,n} \geq L_l$, как указано атрибутом `channel` структуры `Channel`, со следующими значениями:

$$\begin{aligned} \texttt{channel.polar\_position.azimuth} &= \varphi_{j,r}; \\ \texttt{channel.polar\_position.elevation} &= \theta'_{l,r}; \\ \texttt{channel.polar\_nominal\_position.azimuth} &= \varphi_{j,n}; \\ \texttt{channel.polar\_nominal\_position.elevation} &= \theta'_{l,n}. \end{aligned}$$

В обоих случаях коэффициенты понижающего микширования направляют коэффициенты усиления от этого громкоговорителя к соответствующему громкоговорителю среднего уровня j.

Это реализуется в модуле `core.point_source.extra_pos_vertical_nominal`.

6.1.3.2 Процесс для схемы расположения `0+2+0`

Для схемы расположения 0+2+0 получается `PointSourcePanner` с одной областью `StereoPanDownmix`.

Это реализуется в модуле `core.point_source._configure_stereo`.

6.2 Определение нахождения угла в пределах допустимого диапазона

При сравнении углов с заданными угловыми диапазонами используется функция `inside_angle_range`, что позволяет определить диапазоны, в которые входит задняя часть системы координат. Она применяется в компонентах исключения зон и в объекте *DirectSpeakers* в пунктах 7.3.12.1 и 8.4.

Соответствующая сигнатура:

```
bool inside_angle_range(float x, float start, float end, float tol=0.0);
```

Это дает результат true (истинно), если угол *x* находится внутри дуги окружности, которая начинается в точке `start` и движется против часовой стрелки до точки `end`, развернувшись на величину `tol`. Все углы даны в градусах.

В общем случае, когда:

$$-180 \le \texttt{start} \le \texttt{end} \le 180,$$

эта функция эквивалентна:

$$\texttt{start} - \texttt{tol} \le x' \le \texttt{end} + \texttt{tol},$$

где $x' = x + 360 \times i$ для некоторых i, таких что $-180 < x' \le 180$.

В других случаях функция ведет себя сложнее. Например, если `start` = 90 и `end` = –90, это указывает на заднюю половину системы координат:

$$x' \le -90 \vee x' \ge 90.$$

Некоторые примеры диапазонов и эквивалентные выражения приведены в таблице 2.

ТАБЛИЦА 2

Выражения, эквивалентные функции `inside_angle_range(x, start, end, tol)`

`start`	`end`	`tol`	**Эквивалентное выражение**
−90	90	0	$-90 \le x' \le 90$
−90	90	5	$-95 \le x' \le 95$
90	−90	0	$x' \le -90 \vee x' \ge 90$
90	−90	5	$x' \le -85 \vee x' \ge 85$
0	0	0	$x' = 0$
180	180	0	$x' = 180$
−180	−180	0	$x' = 180$
180	180	5	$x' \le -175 \vee x' \ge 175$
−180	180	0	true

Эта функция реализуется в модуле `core.geom.inside_angle_range`.

6.3 Определение канала LFE по его частотным метаданным

Частотные метаданные, которые могут присутствовать в качестве подэлементов *frequency* элементов *audioChannelFormat*, можно использовать для определения того, действительно ли канал является каналом LFE.

Для представления частотных метаданных используется следующая структура данных:

```
struct Frequency {
  optional<float> lowPass;
```

```
  optional<float> highPass;
};
```

Функция с сигнатурой:

```
bool is_lfe(Frequency frequency)
```

оценивает выражение:

$$\texttt{frequency.lowPass} \wedge \neg \texttt{frequency.highPass} \wedge (\texttt{frequency.lowPass} \leq 120\ \text{Hz})$$

и возвращает значение `True`, если канал признан каналом LFE, и `False` в остальных случаях.

Это реализуется в модуле `core.renderer_common.is_lfe`.

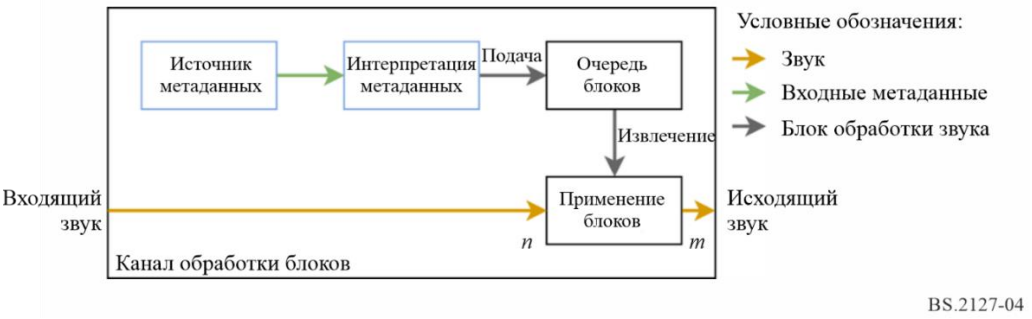
ПРИМЕЧАНИЕ. – В Рекомендации МСЭ-R BS.2127-0 значение частоты для выбора каналов LFE принимается равным 200 Гц или меньше.

6.4 Канал обработки блоков

При рендеринге хронируемых метаданных ADM требуются некоторые функциональные возможности, одинаковые для всех значений *typeDefinition*, в отношении заданного набора входных каналов между границами временны́х интервалов применяется некоторая обработка, приводящая к созданию на выходе каналов громкоговорителей.

РИСУНОК 4

Структура, используемая для обработки связанных каналов. Компоненты, обведенные голубой рамкой, предоставляются извне



На рисунке 4 показана структура, используемая для достижения этой цели. Интерфейс к этому компоненту выглядит следующим образом:

```
class BlockProcessingChannel {
    BlockProcessingChannel(MetadataSource metadata_source, Callable
interpret_metadata);
    void process(int sample_rate, int start_sample,
        ndarray<float> input_samples, ndarray<float> &output_samples);
};
```

Источник метаданных `MetadataSource` предоставляется системой как механизм подачи метаданных в рендерер. Он имеет следующий интерфейс:

```
class MetadataSource {
    optional<TypeMetadata> get_next_block();
};
```

Многократно вызывая функцию `get_next_block`, канал обработки блоков принимает последовательность блоков `TypeMetadata`, описанных в разделе 5, которые соответствуют ограниченным по времени блокам метаданных, требуемым в процессе рендеринга.

Эти блоки метаданных интерпретируются функцией `interpret_metadata`, которая предоставляется рендерером для каждого значения *typeDefintion*. Эта функция принимает данные `TypeMetadata` и возвращает список объектов `ProcessingBlock`, инкапсулирующих функции ограниченной по времени обработки звука, необходимые для реализации данного типа

`TypeMetadata`. Интерпретация для *typeDefinition==Objects* подробно описана в пункте 7.2. Для *typeDefinition==HOA* и *typeDefinition==DirectSpeakers* возвращается один объект `ProcessingBlock`.

Объекты `ProcessingBlock` имеют следующий внешний интерфейс:

```
class ProcessingBlock {
    Fraction start_sample, end_sample;
    int first_sample, last_sample;

    void process(int in_out_samples_start,
        ndarray<float> input_samples, ndarray<float> &output_samples);
}
```

Предполагается, что выборки, передаваемые в обработку, составляют поднабор выборок во входном/выходном файле, так что что `input_samples`$[i]$ и `output_samples`$[i]$ отражают глобальные выходные/выходные выборки `in_out_samples_start` $+ i$. Атрибуты `first_sample` и `last_sample` определяют диапазон глобальных номеров выборок s, на которые будет влиять процесс:

$$\texttt{first\_sample} \le s \le \texttt{last\_sample}.$$

`start_sample` и `end_sample` – дробные номера начала и конца выборки, которые используются для определения атрибутов `first_sample` и `last_sample` и могут использоваться реализациями подкласса `ProcessingBlock`.

Объекты `BlockProcessingChannel` хранят очередь блоков `ProcessingBlock`, которая пополняется путем запроса блоков из источника `metadata_source` и передачи их через `interpret_metadata`. `BlockProcessingChannel.process` применяет обрабатывающие блоки из этой очереди к передаваемым ему выборкам, используя `first_sample` и `last_sample` для определения момента перехода к следующему блоку.

Эта структура позволяет разделить компоненты рендерера; выборки аудиоданных могут обрабатываться порциями, размеры которых не зависят от размеров блока метаданных, сохраняя при этом обработку метаданных с точностью до выборки и не усложняя рендереры требованием хронирования.

Решение позволить рендереру самостоятельно извлекать (pull) блоки метаданных сохраняет интерпретацию метаданных хронирования в рендерере – если бы вместо этого метаданные подавались (push) в рендерер, то компоненту, выполняющему подачу, должно было бы быть известно, когда потребуется следующий блок, а это зависит от информации хронирования внутри него.

Эта функциональная возможность реализуется в модуле `core.renderer_common`.

6.4.1 Реализованные типы `ProcessingBlock`

Имеются три общих типа обрабатывающих блоков.

`FixedGains` принимает один входной канал и применяет n коэффициентов усиления, суммируя результат в n выходных каналов.

`FixedMatrix` принимает N входных каналов и применяет матрицу коэффициентов усиления NxM для формирования M выходных каналов.

`InterpGains` принимает один входной канал и применяет n линейно интерполированных коэффициентов усиления, суммируя результат в n выходных каналов. Предусмотрены два вектора усиления – `gains_start` и `gains_end`, которые представляют собой коэффициенты усиления, применяемые во время `start_sample` и `end_sample`. Коэффициент усиления $g(i, s)$, применяемый к каналу i в выборке s, определяется следующим образом:

$$p(s) = \frac{s - \texttt{start\_sample}}{\texttt{end\_sample} - \texttt{start\_sample}};$$

$$g(i, s) = (1 - p(s)) \times \texttt{gains\_start}[i] + p(s) \times \texttt{gains\_end}[i].$$

6.5 Общая интерпретация метаданных хронирования

Определение времени начала и окончания блока распределяется между рендererами для различных типов *typeDefinition*. Для блока объектов `TypeMetadata` используется следующий процесс.

– Время начала и окончания объекта, содержащего блок, определяется атрибутами `block.extra_data.object_start` и `block.extra_data.object_duration`. Если `object_start` имеет значение None, предполагается, что объект запускается в момент времени 0. Если время `object_duration` равно None, предполагается, что оно длится до бесконечности.

– Время начала и окончания блока определяется атрибутами `rtime` и `duration`:

  • если `rtime` и `duration` не равны `None`, то предполагается, что время начала блока – это время начала объекта плюс `rtime`, а время окончания блока – время начала блока плюс `duration`;

  • если `rtime` и `duration` равны `None`, то предполагается, что блок длится с момента начала объекта до момента окончания объекта;

  • другие сочетания значений `rtime` и `duration` считаются **ошибкой**. Для нескольких объектов *audioBlockFormat* в *audioChannelFormat* следует указывать как *rtime,* так и *duration*, тогда как для одного блока, охватывающего весь *audioObject, rtime* и *duration* указывать не нужно. В остальных случаях поведение не определено.

Время должно проверяться на согласованность. Блоки, заканчивающиеся после окончания объекта или перекрывающиеся блоки в последовательности, не должны допускаться и считаются ошибкой. Состояние ошибки означает, что реализация должна считать, что что-то не так со входными данными. Правильный порядок действий состоит в том, чтобы исправить систему, породившую это состояние. В эталонной реализации ошибки обрабатываются путем остановки процесса рендеринга и сообщения об ошибке пользователю. В других реализациях могут использоваться разные стратегии обработки ошибок в зависимости от условий их целевого применения.

Это реализуется в модуле `core.renderer_common.InterpretTimingMetadata`.

6.6 Интерпретация объектов `TrackSpec`

Ввод аудиоданных в рендерер осуществляется по многоканальной шине, которая считывает данные непосредственно из входного файла. В число входных метаданных в форме параметров `RenderingItem` входят объекты `TrackSpec`, которые представляют собой инструкции по извлечению каналов из этой шины, включая применение предварительной матричной обработки, которая смешивает несколько каналов.

Обработка каждого типа `TrackSpec` реализуется в модуле `core.track_processor`.

При наличии `TrackSpec` можно создать объект `TrackProcessor` с одной функцией `process(sample_rate, input_samples)`, который применяет указанный метод обработки к входным выборкам `input_sample` и возвращает одноканальный результат (с заданной частотой дискретизации).

6.6.1 SilentTrackSpec

Функция `process` обработки объектов `SilentTrackSpec` для *n* входных выборок возвращает *n* выборок с нулевым значением.

6.6.2 DirectTrackSpec

Функция `process` обработки объекта `DirectTrackSpec track_spec` возвращает входные выборки дорожки, указанной в `track_spec.track_index` (с использованием индексов, начиная с нуля).

6.6.3 MixTrackSpec

Функция `process` обработки объектов `MixTrackSpec` `track_spec` возвращает сумму результатов вызова `process` в `TrackProcessor` для каждой субдорожки из `track_spec.input_tracks`.

6.6.4 MatrixCoefficientTrackSpec

Функция `process` обработки объектов `MatrixCoefficientTrackSpec` `track_spec` применяет метод матричной обработки, указанный атрибутом `track_spec.coefficient` (который содержит параметры одного матричного элемента *coefficient*), к одному каналу, указанному атрибутом `track_spec.input_track`.

Если `track_spec.coefficient.gain` не равен `None`, то выборки умножаются на коэффициент усиления `gain`.

Если `track_spec.coefficient.delay` не равен `None`, то выборки задерживаются на n выборок или `delay` микросекунд (мс) с округлением до ближайшей выборки (в меньшую сторону):

$$n = \left\lceil \frac{\texttt{sample\_rate} \times \texttt{delay}}{1000} - \frac{1}{2} \right\rceil.$$

Некоторые параметры не поддерживаются. Если `gainVar`, `delayVar`, `phaseVar` или `phase` не равны `None` или `delay` имеет отрицательное значение, возникает состояние ошибки.

6.7 Относительный угол

Функция $\texttt{relative\_angle}(x, y)$ используется для нахождения эквивалентного угла y, большего или равного x. Это делается во избежание краевых случаев при работе с дугами окружности.

Функция $\texttt{relative\_angle}(x, y)$ возвращает значение $y' = y + 360n$, где n – такое наименьшее целое, что $y' \geq x$.

6.8 Преобразования координат

Определяется функция `cart` для перевода позиций из полярных координат в декартовы в соответствии с пунктом 2.2:

$$cart(\varphi, \theta, d) = \{x, y, z\},$$

где:

$$\begin{aligned} x &= \sin\left(-\frac{\pi}{180}\varphi\right)\cos\left(\frac{\pi}{180}\theta\right)d; \\ y &= \cos\left(-\frac{\pi}{180}\varphi\right)\cos\left(\frac{\pi}{180}\theta\right)d; \\ z &= \sin\left(\frac{\pi}{180}\theta\right)d. \end{aligned}$$

Также определяются обратные преобразования для получения азимута и угла места из позиции в декартовых координатах:

$$\begin{aligned} \texttt{azimuth}(\{x, y, z\}) &= -\frac{180}{\pi}\,\texttt{atan2}(x, y); \\ \texttt{elevation}(\{x, y, z\}) &= \frac{180}{\pi}\,\texttt{atan2}\left(z, \sqrt{x^2 + y^2}\right). \end{aligned}$$

Функция `local_coordinate_system` создает матрицу поворота, которая отображает позицию {0,1,0} на данные азимут и угол места:

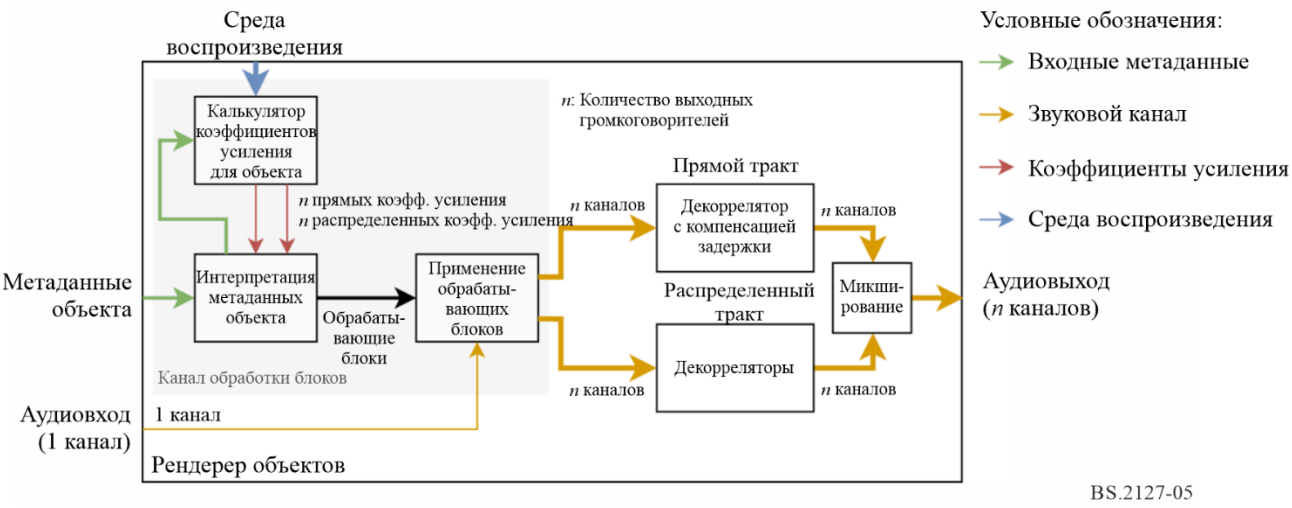
$$\texttt{local\_coordinate\_system}(\varphi,\theta) = \begin{bmatrix} \texttt{cart}(\varphi - 90,0,1) \\ \texttt{cart}(\varphi,\theta,1) \\ \texttt{cart}(\varphi,\theta + 90,1) \end{bmatrix}.$$

7 Элементы рендерера typeDefinition==Objects

7.1 Структура

РИСУНОК 5

Структура рендерера объектов



На рисунке 5 показана структура рендера при *typeDefinition==Objects*. Этот рисунок иллюстрирует процесс обработки одного элемента рендеринга; рендеринг нескольких элементов происходит так, как будто эта структура дублируется для каждого элемента, а выходные данные смешиваются.

Метаданные поступают в рендерер в форме объекта `ObjectRenderingItem`, который содержит индекс дорожки, и источника объектов `ObjectTypeMetadata`, представляющих собой ограниченные во времени параметры рендеринга для указанной дорожки.

К каждому объекту `ObjectTypeMetadata` применяется метод, описанный в пункте 7.2; он интерпретирует метаданные хронирования и вычисляет векторы усиления с использованием калькулятора коэффициентов усиления, описанного в пункте 7.3. Это приводит к созданию объектов `ProcessingBlock`, которые применяют ограниченные по времени операции обработки сигналов ко входному звуковому сигналу, создавая прямую и распределенную шины, каждая из которых содержит по одному каналу на громкоговоритель. Этот подход и инкапсулирующий его класс `BlockProcessingChannel` описаны в пункте 6.4.

Распределенная шина проходит через банк фильтров декорреляции для каждого канала, а прямая шина задерживается для согласования перед микшированием в целях формирования выходного сигнала. Фильтры декорреляции и задержки описаны в пункте 7.4.

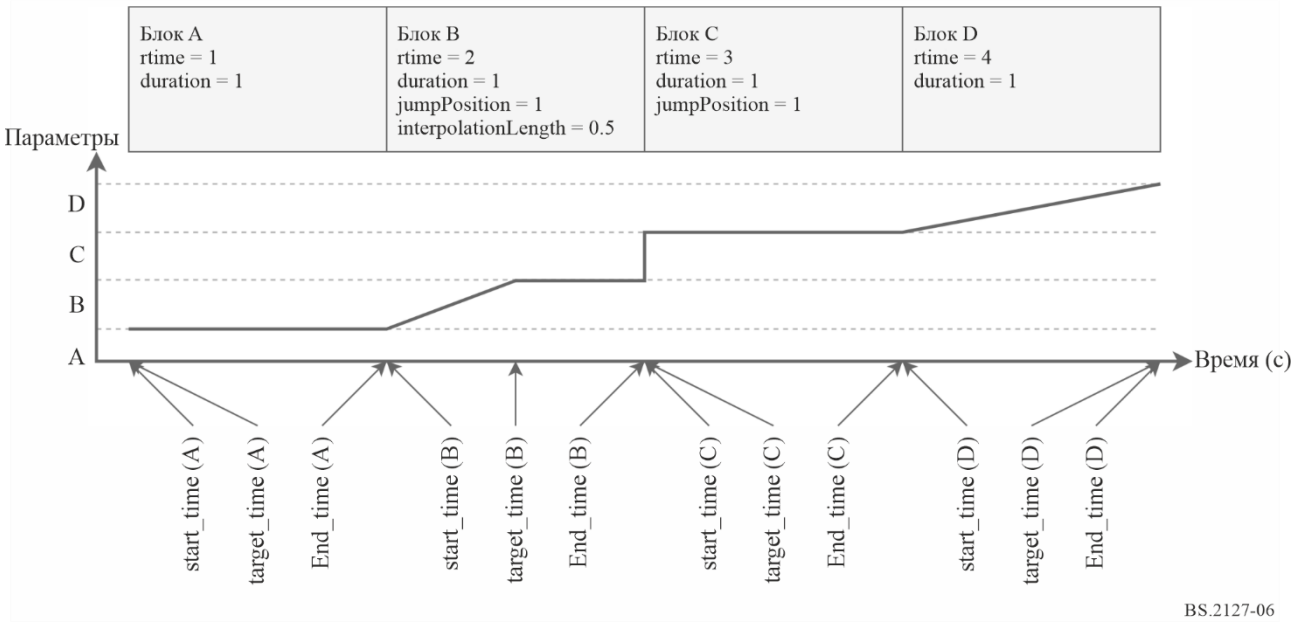
Эта структура реализуется в модуле `core.objectbased.renderer.ObjectRenderer`.

7.2 Класс InterpretObjectMetadata

Метаданные хронирования объектов интерпретируются в классе `InterpretObjectMetadata`, который входит в структуру канала обработки блоков.

РИСУНОК 6

Пример объектов audioBlockFormat и интерпретируемых кривых интерполяции



Для каждого входного объекта `ObjectTypeMetadata` используется следующий процесс.

– Определяется время начала и окончания блока `start_time` и `end_time` в соответствии с пунктом 6.5.

– Время, когда интерполяция в этом блоке заканчивается, `target_time` определяется согласно следующим случаям, которые иллюстрируются соответствующими блоками на рисунке 6.

A

Если это первый блок или если `end_time` предыдущего блока меньше, чем `start_time` текущего, то:

$$\texttt{target\_time} = \texttt{start\_time}.$$

B

Если установлен флаг bf.jumpPosition.flag и значение bf.jumpPosition.interpolationLength не равно None, то:

$$\texttt{target\_time} = \texttt{start\_time} + \texttt{bf.jumpPosition.interpolationLength}.$$

C

Если установлен флаг bf.jumpPosition.flag и значение bf.jumpPosition.interpolationLength равно None, то:

$$\texttt{target\_time} = \texttt{start\_time}.$$

D

Если флаг `bf.jumpPosition.flag` не установлен, то осуществляется интерполяция по всему блоку:

$$\texttt{target\_time} = \texttt{end\_time}.$$

– Вектор усиления `interp_to` рассчитывается с использованием экземпляра `GainCalculator` для текущего блока. `interp_from` – это вектор усиления, рассчитанный для предыдущего блока.

– Если start_time < target_time, создается объект InterpGains ProcessingBlock, который интерполирует от interp_from до interp_to в период между start_time и target_time.

– Если target_time < end_time, создается объект FixedGains ProcessingBlock, который применяет interp_to в период между start_time и target_time.

Это реализуется в модуле core.objectbased.renderer.InterpretObjectMetadata.

7.3 Калькулятор коэффициентов усиления

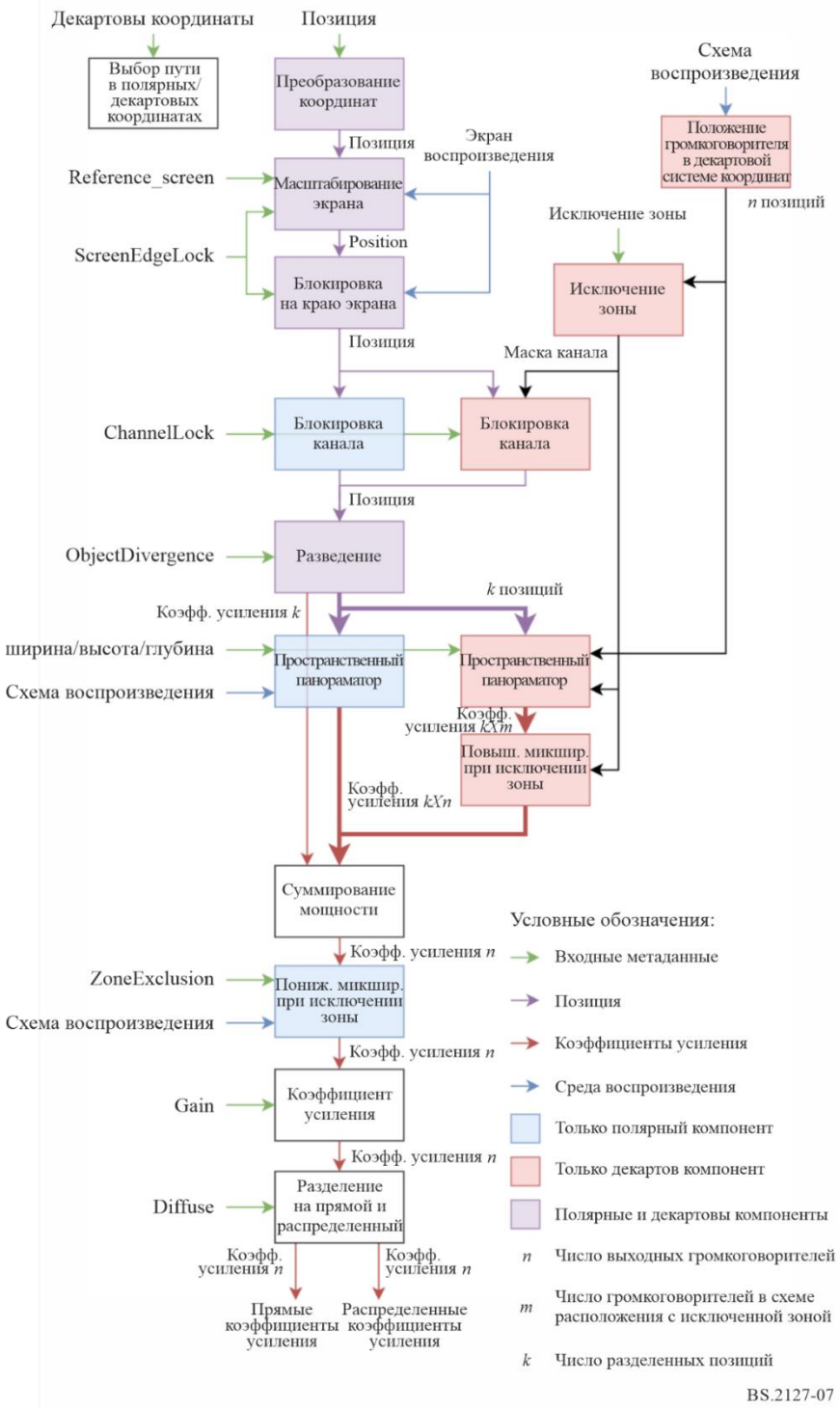
Если имеется объект ObjectTypeMetadata, этот объект рассчитывает коэффициенты усиления для каждого громкоговорителя в прямом и распределенном трактах. Интерфейс для этого компонента:

```
struct      DirectDiffuseGains {
    vector<float> direct;
    vector<float> diffuse;
};

class GainCalc {
    GainCalc(Layout layout);

    DirectDiffuseGains render(ObjectTypeMetadata otm);
};
```

7.3.1 Структура

РИСУНОК 7

Структура калькулятора коэффициентов усиления при *typeDefintion==Objects*



Этот компонент состоит главным образом из подкомпонентов, перечисленных в настоящем разделе. Схема прохождения сигнала между этими компонентами показана на рисунке 7. Поведение объекта `ObjectTypeMetadata otm`, содержащего атрибут `block_format bf`, является следующим.

– К параметру `bf.position` применяется преобразование координат, описанное в пункте 7.3.2, для получения позиции объекта `CartesianPosition position`.

– Применяется масштабирование экрана с использованием метода, описанного в пункте 7.3.3, с параметрами `position`, `bf.screenRef`, `otm.extra_data.reference_screen` и `bf.cartesian` и обновлением значения `position`. Этот компонент инициализируется с параметрами экрана воспроизведения (`layout.screen`) и схемы воспроизведения (`layout`).

– Блокировка на краю экрана применяется с использованием метода, описанного в пункте 7.3.4, с параметрами `position`, `bf.position.screenEdgeLock` и `bf.cartesian` и обновлением значения `position` в соответствии с результатом. Этот компонент инициализируется с параметрами экрана воспроизведения (`layout.screen`) и схемы воспроизведения (`layout`).

– Если `bf.cartesian`, то:

  • аллоцентрическое положение каждого громкоговорителя в схеме расположения `layout.without_lfe` определяется в соответствии с пунктом 7.3.9, что дает массив `allo_channel_positions`;

  • алгоритм исключения зоны, описанный в пункте 7.3.5, применяется к `allo_channel_positions` и `bf.zone_exclusion`, давая булеву маску громкоговорителей для исключения `excluded`;

  • применяется блокировка канала в аллоцентрической конфигурации, описанная в пункте 7.3.6, с параметрами `position`, `bf.channelLock` и `excluded`, обновляющая значение `position`;

  в противном случае:

  • применяется блокировка канала в эгоцентрической конфигурации, описанная в пункте 7.3.6, с параметрами `position` и `bf.channelLock`, обновляющая значение `position`.

– Применяется разведение с помощью метода, описанного в пункте 7.3.7, с параметрами `position`, `bf.objectDivergence` и `bf.cartesian`. В результате получается до трех разнесенных источников с коэффициентами усиления и позициями, которые хранятся в переменных `diverged_gains` и `diverged_positions`.

– Если `bf.cartesian`, то:

  • пространственный панораматор, описанный в пункте 7.3.11, применяется к каждой позиции `p` в `diverged_positions` с параметрами `channel_positions`, `p`, `bf.width`, `bf.height`, `bf.depth`, в результате чего получают векторы усиления для массива неисключенных громкоговорителей. `channel_positions` – это список позиций неисключенных каналов, выбранных из `allo_channel_positions[i]`, где `excluded[i]` имеет значение `False`;

  • эти векторы усиления микшируются с повышением в соответствии с атрибутом `excluded`, что дает коэффициент усиления для каждого громкоговорителя `i`, когда значение `excluded[i]` равно `False`, и ноль, когда значение `excluded[i]` равно `True`, и сохраняются в переменной `gains_for_each_pos`;

  в противном случае:

  • пространственный панораматор, описанный в пункте 7.3.8, применяется к каждой позиции `p` в `diverged_positions` с параметрами `p`, `bf.width`, `bf.height`, `bf.depth`, что дает вектор усиления для каждого громкоговорителя, который сохраняется в переменной `gains_for_each_pos`.

– Коэффициенты усиления в переменной `gains_for_each_pos` микшируются вместе, при этом мощность определяется значением `diverged_gains`:

$$\texttt{gains}[i] = \sqrt{\sum_j \texttt{diverged\_gains}\,[j] \times \texttt{gains\_for\_each\_pos}[j,i]^2}.$$

– Если `bf.cartesian` не задано, то к коэффициентам усиления `gains` и `bf.zoneExclusion` применяется исключение зоны, как описано в пункте 7.3.12, что приводит к новому вектору `gains`. Этот компонент инициализируется со значением `layout.without_lfe`.

– Вектор `gains` дополняется коэффициентами усиления канала LFE со значением 0, что дает `gains_full` с одним значением на каждый громкоговоритель в схеме расположения `layout`.

– `gains_full` делится на прямой и распределенный векторы для управления прямым и распределенным трактами в зависимости от значения параметра `bf.diffuse`. Эти векторы возвращаются в виде `DirectDiffuseGains` с атрибутами:

$$\begin{aligned} \texttt{direct} &= \texttt{gains\_full} \times \sqrt{1 - \texttt{bf.diffuse}}; \\ \texttt{diffuse} &= \texttt{gains\_full} \times \sqrt{\texttt{bf.diffuse}}. \end{aligned}$$

7.3.1.1 Обсуждение (информационное)

На структуру калькулятора коэффициентов усиления влияют следующие два принципа.

– Если параметры применяются редко (то есть используется лишь небольшое число возможных полей метаданных), желательно сохранить очевидную интерпретацию этих параметров.

– Когда используются комбинации параметров, выбирается тот вариант, который дает пользователю наибольшее количество возможностей для различных полезных действий.

Например:

– Блокировка канала реализуется как изменение позиции – если блокировка канала используется сама по себе (с соответствующим значением *maxDistance*), то источник канала блокируется из-за поведения точечного панораматора, однако блокировку канала также можно использовать с параметрами расширения, например, для создания протяженного источника с центром в позиции определенного громкоговорителя.

– Распределенность не связана с расширением – путем установки соответствующих параметров расширения можно получить полностью распределенный источник, но они также позволяют использовать декорреляционную фильтрацию с неполными значениями расширения.

7.3.2 Преобразование координат

В модуле `core.objectbased.gain_calc.coord_trans`, который используется для преобразования входных позиций в стандартную декартову систему координат, реализовано простое преобразование координат. Он имеет следующую сигнатуру:

```
CartesianPosition coord_trans(ObjectPosition position);
```

Позиция `position` сначала преобразуется в вектор **p** в декартовых координатах.

Если это `ObjectCartesianPosition`, то элементы вектора **p** предварительно усекаются до диапазона $[-1,1]$:

$$\texttt{clip}(\mathbf{p}, -1, 1),$$

в противном случае **p** возвращается неизменным.

Функция `clip` определяется для действительных значений следующим образом:

$$\texttt{clip}(x, a, b) = \begin{cases} a & x \le a; \\ x & a \le x \le b; \\ b & b \le x \end{cases}$$

и тривиально применяется к каждому элементу вектора:

$$\texttt{clip}(\{x, y, z\}, a, b) = \{\texttt{clip}(x, a, b), \texttt{clip}(y, a, b), \texttt{clip}(z, a, b)\}.$$

7.3.3 Масштабирование экрана

Компонент масштабирования экрана деформирует позиции источников, компенсируя различия в геометрии экрана между производственной средой и средой воспроизведения. Интерфейс для этого компонента:

```
class ScreenScaleHandler {
  ScreenScaleHandler(Screen reproduction_screen);
  CartesianPosition handle(
    CartesianPosition position,
    bool screenRef,
    Screen reference_screen,
    bool cartesian
  );
};
```

Используются два определения экрана.

Эталонный экран

– это экран `AudioProgrammeReferenceScreen`, параметры которого указаны в элементе `audioProgramme`, или размер экрана по умолчанию в полярных координатах, если они не указаны. Эта геометрия экрана использовалась при создании метаданных.

Экран воспроизведения

– это геометрия экрана в среде воспроизведения, в которой будет прослушиваться выходной сигнал рендерера.

Позиции на эталонном экране деформируются так, чтобы они оказались в соответствующих позициях на экране воспроизведения.

7.3.3.1 Внутреннее представление экрана

Информация об обоих экранах может быть представлена либо в полярных, либо в декартовых координатах (объекты `PolarScreen` или `CartesianScreen`). В отличие от исходных положений объекта, между ними нет очевидной эквивалентности, но для упрощения реализации требуется единое представление экрана, с помощью которого можно было бы представлять экраны обоих типов. Для этого служит структура `PolarEdges`, в которой хранятся значения азимута левого и правого краев экрана, а также высоты верхнего и нижнего краев:

```
struct PolarEdges {
    float left_azimuth;
    float right_azimuth;
    float bottom_elevation;
    float top_elevation;
};
```

Объект `PolarEdges` создается из данного объекта `PolarScreen` или `CartesianScreen` путем преобразования экрана в центральную позицию декартовой системы координат и в два вектора (вдоль направлений x и z), определяющих поверхность экрана, с последующим определением азимута и высоты каждого края.

Для экрана `PolarScreen`, когда:

$$\begin{aligned} \varphi &= \texttt{screen.centrePosition.azimuth;} \\ \theta &= \texttt{screen.centrePosition.elevation;} \\ d &= \texttt{screen.centrePosition.distance;} \\ w &= \texttt{screen.widthAzimuth;} \\ a &= \texttt{screen.aspectRatio,} \end{aligned}$$

используется следующая процедура.

– Центральная позиция представляет собой простое декартово преобразование центральной позиции:

$$centre = cart(\varphi, \theta, d).$$

– Рассчитываются ширина и высота в декартовых координатах:

$$\begin{aligned} \texttt{width} &= d \cdot \tan\left(\frac{\pi}{180}\frac{w}{2}\right); \\ \texttt{height} &= \frac{\texttt{width}}{a}. \end{aligned}$$

– Для нахождения векторов x и z экрана используется элемент `local_coordinate_system`:

$$\begin{aligned} \begin{bmatrix} l_x \\ l_y \\ l_z \end{bmatrix} &= \texttt{local\_coordinate\_system}(\varphi, \theta); \\ v_x &= \texttt{width} \times l_x; \\ v_z &= \texttt{height} \times l_z. \end{aligned}$$

Для экрана `CartesianScreen screen`, когда:

$$\begin{aligned} w &= \texttt{screen.widthX}; \\ a &= \texttt{screen.aspectRatio}, \end{aligned}$$

используется следующая процедура.

– Центральная позиция используется непосредственно:

$$centre = screen.centrePosition.$$

– Вычисляются ширина и высота:

$$\begin{aligned} \texttt{width} &= \frac{w}{2}; \\ \texttt{height} &= \frac{\texttt{width}}{a}. \end{aligned}$$

– Определяются векторы экрана x и z:

$$\begin{aligned} v_x &= \{\texttt{width}, 0, 0\}; \\ v_z &= \{0, 0, \texttt{height}\}. \end{aligned}$$

Затем можно создать объект `PolarEdges` для экранов обоих типов:

$$\begin{aligned} \texttt{left\_azimuth} &= \texttt{azimuth}(\texttt{centre} - v_x); \\ \texttt{right\_azimuth} &= \texttt{azimuth}(\texttt{centre} + v_x); \\ \texttt{bottom\_elevation} &= \texttt{elevation}(\texttt{centre} - v_z); \\ \texttt{top\_elevation} &= \texttt{elevation}(\texttt{centre} + v_z). \end{aligned}$$

7.3.3.2 Компенсация положения

В некоторых схемах вывода при `cartesian==true` вертикальное панорамирование перед слушателем может быть деформировано. Это компенсируется использованием функции `core.screen_common.compensate_position`:

$$\texttt{compensate\_position}(\varphi, \theta, \texttt{layout}) = \begin{cases} \{\varphi', \theta\} & \text{"U + 045"} \in \texttt{layout.channel\_names}; \\ \{\varphi, \theta\} & \text{в других случаях,} \end{cases}$$

где:

– φ_r образуется путем кусочно-линейной интерполяции θ от

$$\{-90, 0, 30, 90\}$$

до

$$\left\{30{,}30{,}30\,\frac{30}{45}, 30\right\};$$

– φ' образуется путем кусочно-линейной интерполяции φ от

$$\{-180, -30{,}30{,}180\}$$

до

$$\{-180, -\varphi_r, \varphi_r, 180\}.$$

7.3.3.3 Деформация направления

Деформация позиций определяется в функции `core.screen_scale.PolarScreenScaler.scale_az_el`, которая независимо деформирует значения азимута и угла места. С учетом атрибутов `PolarEdges ref` эталонного экрана и `PolarEdges rep` экрана воспроизведения это работает следующим образом.

– К азимуту применяется кусочно-линейная интерполяция, преобразующая значения:

$$\{-180, \texttt{ref.right\_azimuth}, \texttt{ref.left\_azimuth}, 180\}$$

в

$$\{-180, \texttt{rep.right\_azimuth}, \texttt{rep.left\_azimuth}, 180\}.$$

– К углу места применяется кусочно-линейная интерполяция, преобразующая значения:

$$\{-90, \texttt{ref.bottom\_elevation}, \texttt{ref.top\_elevation}, 90\}$$

в

$$\{-90, \texttt{rep.bottom\_elevation}, \texttt{rep.top\_elevation}, 90\}.$$

Результат помещается в функцию `core.screen_scale.PolarScreenScaler.scale_position`, которая применяет метод `scale_az_el` к компонентам азимута и угла места декартового вектора, оставляя расстояние неизменным.

7.3.3.4 Интерпретация метаданных

Если установлен параметр `screenRef` и представлен экран воспроизведения, то позиция передается посредством функции `PolarScreenScaler.scale_direction` с эталонным экраном и экраном воспроизведения. В противном случае позиция возвращается без изменений.

Если параметр `screenRef` не задан или экран воспроизведения не указан, то позиция возвращается без изменений. В противном случае поведение зависит от флага `cartesian`.

– Если установлен флаг `cartesian`, то применяется масштабирование в полярных координатах и компенсация путем преобразования, описанного в пункте 10.1, что дает новую позицию $\{x', y', z'\}$:

$$\begin{aligned}\{\varphi, \theta, d\} &= \texttt{point\_cart\_to\_polar}(\texttt{position.x}, \texttt{position.y}, \texttt{position.z});\\ \{\varphi_s, \theta_s\} &= \texttt{scale\_az\_el}(\varphi, \theta);\\ \{\varphi_{sc}, \theta_{sc}\} &= \texttt{compensate\_position}(\varphi_s, \theta_s, \texttt{layout});\\ \{x', y', z'\} &= \texttt{point\_cart\_to\_polar}(\varphi_{sc}, \theta_{sc}, d).\end{aligned}$$

– В противном случае к компонентам азимута и угла места позиции применяется метод `scale_az_el`.

7.3.4 Блокировка на краю экрана

Компонент блокировки на краю экрана изменяет позиции источников так, чтобы разместить источник на указанном краю экрана. Он имеет следующий интерфейс:

```
class ScreenEdgeLockHandler {
  ScreenEdgeLockHandler(Screen reproduction_screen);
```

```
    CartesianPosition handle_vector(
      CartesianPosition position,
      ScreenEdgeLock screen_edge_lock,
      cartesian=False
    );

    tuple<float, float> handle_az_el(
      float azimuth,
      float elevation,
      ScreenEdgeLock screen_edge_lock
    );
  };
```

При инициализации этот компонент преобразует `reproduction_screen` в объект `PolarEdges polar_edges`, как указано в пункте 7.3.3.1.

Метод `handle_az_el` независимо изменяет азимут и угол места, что приводит к новым значениям азимута и угла места.

– Если `screen_edge_lock.horizontal` имеет значение `LEFT`, то устанавливается значение азимута `polar_edges.left_azimuth`; если `RIGHT`, то устанавливается значение азимута `polar_edges.right_azimuth`; в противном случае азимут остается неизменным.

– Если `screen_edge_lock.vertical` имеет значение `TOP`, то устанавливается значение угла места `polar_edges.top_elevation`; если `BOTTOM`, то устанавливается значение угла места `polar_edges.bottom_elevation`; в противном случае угол места остается неизменным.

Если флаг `reproduction_screen` не установлен, то изменение положения не происходит.

Обработка производится в полярной области, поэтому позиции в декартовых координатах следует предварительно преобразовать. Если вместо метода `handle_az_el` используется метод `handle_vector`, то применяется преобразование туда и обратно.

– Если установлен флаг `cartesian`, то применяется масштабирование в полярных координатах и компенсация путем преобразования, описанного в пункте 10.1, что дает новую позицию $\{x', y', z'\}$:

$$\begin{aligned}\{\varphi, \theta, d\} &= \texttt{point\_cart\_to\_polar}(\texttt{position.x}, \texttt{position.y}, \texttt{position.z});\\ \{\varphi_s, \theta_s\} &= \texttt{handle\_az\_el}(\varphi, \theta, \texttt{screen\_edge\_lock});\\ \{\varphi_{sc}, \theta_{sc}\} &= \texttt{compensate\_position}(\varphi_s, \theta_s, \texttt{layout});\\ \{x', y', z'\} &= \texttt{point\_cart\_to\_polar}(\varphi_{sc}, \theta_{sc}, d).\end{aligned}$$

– В противном случае к компонентам азимута и угла места позиции применяется метод `handle_az_el`.

Этот компонент реализуется в модуле `core.screen_edge_lock.ScreenEdgeLockHandler`.

7.3.5 Исключение декартовой зоны

Алгоритм исключения декартовой зоны начинается с полной схемы воспроизведения в виде `channel_positions` и обрабатывает объекты `ExclusionZone`, чтобы определить, какие громкоговорители следует исключить – соответствующий алгоритм описан в пункте 7.3.12.1. Каждый громкоговоритель, который, как установлено, находится в любой из областей, указанных объектом `ExclusionZone`, исключается, и если это приводит к уменьшению ряда громкоговорителей (с одинаковыми координатами y и z) до одного громкоговорителя, то все громкоговорители этого ряда исключаются, так что основные свойства, определенные для точечного панораматора в пункте 7.3.10, сохраняются.

Если процесс применения исключения зоны мог бы привести к исключению всех громкоговорителей, то громкоговорители не исключаются.

Наконец, создается матрица повышающего микширования, которая преобразует каналы в сокращенной схеме расположения в исходные каналы полной схемы с единичным усилением.

Это реализуется в модуле `core.allocentric.apply_zone_exclusion`.

7.3.6 Блокировка канала

Блокировка канала реализуется как преобразование позиций. Если установлен элемент *channelLock* и громкоговоритель находится в диапазоне, указанном в элементе *maxDistance*, то позиция преобразуется в позицию громкоговорителя, ближайшую к исходной. При отсутствии метаданных divergence, extent, zone exclusion и diffuse источник будет воспроизводиться непосредственно выбранным громкоговорителем.

Класс `objectbased._gain_calc.ChannelLockHandlerBase` имеет следующую сигнатуру:

```
class ChannelLockHandlerBase {
  ChannelLockHandlerBase(Layout layout);
  CartesianPosition handle(
    CartesianPosition position,
    optional<ChannelLock> channelLock,
    vector<bool> excluded,
  );
};
```

`excluded` – это маска исключения каналов, указывающая, какие громкоговорители следует игнорировать, которая используется только в аллоцентрическом тракте, поскольку только там после исключения зоны выполняется блокировка канала.

Для эгоцентрического тракта `ChannelLockHandlerBase` настраивается в классе `core.objectbased._gain_calc.EgoChannelLockHandler`.

Для аллоцентрического тракта `ChannelLockHandlerBase` настраивается в классе `core.objectbased._gain_calc.AlloChannelLockHandler`.

В эгоцентрическом режиме рассматриваемые позиции громкоговорителей представляют собой нормализованные реальные позиции громкоговорителей в `layout`, а в аллоцентрическом – это позиции в соответствии со значением `core.allocentric.positions_for_layout (layout)`, как описано в пункте 7.3.9.

Для применения метаданных блокировки канала используется следующая процедура.

– Если значение `excluded` не равно `None`, то на следующих шагах громкоговорители, для которых `excluded[n]==True` (`n` – это *n*-й громкоговоритель), не рассматриваются.

– Если `channelLock` имеет значение `None`, возвращается исходная `позиция`.

– Если `channelLock.maxDistance` не равно `None`, рассчитывается расстояние ℓ_2 между позицией каждого громкоговорителя и значением `position` и определяются в качестве возможных громкоговорителей все громкоговорители, расстояние до которых (в пределах некоторого допуска) меньше `channelLock.maxDistance`.

– Если возможных громкоговорителей не выявлено, возвращается `position`.

– В наборе возможных громкоговорителей определяются те из них, которые находятся ближе всего к точке `position`. В эгоцентрической конфигурации используется расстояние ℓ_2 между `position` и каждым громкоговорителем, а в аллоцентрической – взвешенное расстояние между `position` и каждым громкоговорителем. Взвешенное расстояние рассчитывается следующим образом:

$$dw_i = \sqrt{w_x \times \left(x_o - x_{spkr_i}\right)^2 + w_y \times \left(y_o - y_{spkr_i}\right)^2 + w_z \times \left(z_o - z_{spkr_i}\right)^2},$$

где:

$$w_x = \frac{1}{16};$$
$$w_y = 4;$$
$$w_z = 32.$$

– При отсутствии ближайшего громкоговорителя (в пределах некоторого допуска) из набора ближайших громкоговорителей выбирается громкоговоритель с наивысшим приоритетом. Порядок приоритетности громкоговорителей определяется путем лексикографического сравнения кортежа:

$$\{|\theta|, \theta, |\varphi|, \varphi\},$$

где φ и θ – действительные азимут и угол места громкоговорителя. Чем ниже значение кортежа, тем выше приоритет – наивысший приоритет имеют громкоговорители с малым абсолютным углом места, далее следует угол места, затем абсолютный азимут и, наконец, азимут.

– Возвращается позиция выбранного громкоговорителя.

7.3.7 Разведение

Разведение реализуется путем добавления двух дополнительных позиций источников $\mathbf{p}_l$ и $\mathbf{p}_r$ слева и справа от исходного положения источника $\mathbf{p}_c$. Каждая позиция источника связана со значением коэффициента усиления g_l, g_c и g_r.

Метаданные разведения интерпретируются функцией `core.objectbased.gain_calc.diverge` со следующей сигнатурой:

```
tuple<vector<float>, vector<CartesianPosition>> diverge(
  CartesianPosition position,
  ObjectDivergence objectDivergence,
  bool cartesian
);
```

Эта функция принимает трехмерную позицию (в данном случае выход функции блокировка канала) и применяет метаданные разведения, введенные в `objectDivergence`. Создаются три позиции источников с соответствующими коэффициентами усиления, каждая из которых передается в пространственный панораматор для рендеринга.

Расчет этих коэффициентов усиления и позиций описан ниже.

7.3.7.1 Расчет коэффициентов усиления

Для данного значения x атрибута `objectDivergence.value` рассчитываются три коэффициента усиления:

$$g_c = \frac{1-x}{x+1};$$
$$g_l = g_r = \frac{x}{x+1}.$$

Это удовлетворяет следующим требованиям:

– $\forall x,\ g_l + g_r + g_c = 1;$

– $x = 0 \Longrightarrow g_l = g_r = 0 \wedge g_c = 1;$

– $x = \frac{1}{2} \Longrightarrow g_l = g_r = g_c = \frac{1}{3};$

– $x = 1 \Longrightarrow g_l = g_r = 0{,}5 \wedge g_c = 0.$

7.3.7.2 Расчет позиций

Полученные позиции зависят от флага `cartesian` в формате блока. Если установлены флаги `azimuthRange` и `cartesian` или если флаг `positionRange` установлен, а `cartesian` – нет, выдается предупредительное сообщение.

7.3.7.2.1 Поведение при `cartesian==true`

Когда `position` принимает значение $\mathbf{p}$, а `objectDivergence.positionRange` – значение x, центральная позиция просто смещается влево и вправо на x вдоль оси x и усекается до $[-1,1]$:

$$\begin{aligned}\mathbf{p}_c &= \mathtt{clip}(\mathbf{p}, -1,1);\\ \mathbf{p}_l &= \mathtt{clip}(\mathbf{p} - \{x, 0,0\}, -1,1);\\ \mathbf{p}_r &= \mathtt{clip}(\mathbf{p} + \{x, 0,0\}, -1,1).\end{aligned}$$

Функция `clip` определяется в пункте 7.3.2.

7.3.7.2.2 Поведение при `cartesian==false`

Для данного значения a атрибута `objectDivergence.azimuthRange` позиции рассчитываются таким образом, чтобы с точки зрения слушателя левый и правый источники были смещены на a градусов влево и вправо от центра и все три источника находились на прямой линии.

Это достигается путем определения трех позиций, расположенных вокруг оси $+y$ на расстоянии $d = \| \mathbf{p}_c \|_2$, где $\mathbf{p}_c$ – исходная позиция источника:

$$P'_l = \mathtt{cart}(a, 0, d);$$

$$P'_r = \mathtt{cart}(-a, 0, d);$$

$$P'_c = \mathtt{cart}(0,0, d).$$

Затем они поворачиваются вокруг исходного направления на источник с помощью матрицы вращения $\mathbf{M}$, которая определяется так, что $\mathbf{p}_c{}'$ отображается на исходную позицию источника $\mathbf{p}_c$:

$$[\mathbf{p}_l, \mathbf{p}_r, \mathbf{p}_c]^T = \mathbf{M} \cdot [\mathbf{p}'_l, \mathbf{p}'_r, \mathbf{p}'_c]^T.$$

7.3.8 Полярный пространственный панораматор

Обработка параметров полярного пространственного панораматора ADM осуществляется в классе `core.objectbased.gain_calc.PolarExtentHandler`; для создания вектора усиления при заданных параметрах положения и расширения в нем используются модули, описанные ниже.

Интерфейс к этому классу:

```
class PolarExtentHandler {
  PolarExtentHandler(PointSourcePanner psp);

  vector<float> handle(
    CartesianPosition position,
    float width,
    float height,
    float depth);
};
```

Структура класса `PolarExtentHandler` показана на рисунке 8.

РИСУНОК 8

Структура обработчика расширения

Внутренне этот объект содержит ссылку на `PolarExtentPanner`, как описано в пункте 7.3.8.2, которую он использует для вычисления векторов усиления.

Параметры `width` (ширина), `height` (высота) и `position` (положение) должны дублироваться и изменяться для обработки параметра `depth` (глубина) и компонента расстояния `position`; эти параметры передаются через полярный пространственный панораматор для генерирования вектора усиления для каждого громкоговорителя и, наконец, эти векторы усиления смешиваются. Эта процедура описана в пункте 7.3.8.2.

В режимах пространственного рендеринга в полярных координатах для генерирования коэффициентов усиления громкоговорителей используется расширяющий панораматор, описанный ниже.

7.3.8.1 Расширяющий панораматор

Форма распространенных источников определяется в рендерере с помощью весовой функции, которая при заданном трехмерном направлении позволяет рассчитать вес данного направления. Этот вес можно рассматривать как силу звука, которую данный объект должен воспроизводить в данном направлении. Например, для расположенного перед слушателем источника, который в ширину больше, чем в высоту, можно использовать весовую функцию, подобную той, которая представлена на рисунке 10.

Получив для каждого громкоговорителя коэффициенты усиления, отражающие эту весовую функцию, и применяя эти коэффициенты усиления к монофоническому объекту, а также применяя к результирующим каналам декорреляционную фильтрацию, можно создать впечатление распространенного или протяженного источника звука с заданными параметрами расширения.

Для вычисления вектора усиления для заданной весовой функции используется класс `SpreadingPanner`.

Набор из 1652 виртуальных позиций, используемый в расширяющем панораматоре, определяется следующим образом.

Для каждого угла места θ в диапазоне от $-90°$ до $90°$, включающем пять шагов, рассчитывается количество точек n, которые должны быть равномерно распределены по окружности при этом угле места, чтобы достичь приблизительно однородной плотности на поверхности единичной сферы:

$$n' = \frac{360}{5}\cos\theta$$

$$n = \max(\text{round}(n'),\ 1)$$

Далее, для каждого значения i в диапазоне от 0 до n – 1, включительно, рассчитывается азимут φ:

$$\varphi = 360\,\frac{i}{n}$$

Результатом являются декартовы координаты точки (φ, θ, 1).

Объекты этого типа содержат набор позиций виртуальных источников и вектор усиления громкоговорителя для каждой из этих позиций.

При включении системы точечный панораматор вычисляет вектор усиления для каждой позиции.

Для вычисления вектора усиления для данной весовой функции эта весовая функция применяется к позициям виртуальных источников. Результирующий вектор усиления для каждого виртуального источника умножается на предварительно рассчитанные векторы усиления громкоговорителей, так что получается один вектор усиления для каждого громкоговорителя. Затем он нормализуется по мощности для получения окончательного вектора усиления.

Это реализуется в модуле `core.objectbased.extent.SpreadingPanner`.

7.3.8.2 Пространственный рендеринг в полярных координатах

Процедура, используемая для расчета коэффициентов усиления громкоговорителей для параметров `position`, `width`, `height` и `depth` в полярных координатах, является следующей.

– Параметр `depth` интерпретируется как два распространенных источника с одинаковым направлением, но разными расстояниями. Расстояния до каждого:

$$d_1 = \max\left\{0, \|\,\texttt{position}\,\|_2 + \frac{\texttt{depth}}{2}\right\};$$

$$d_2 = \max\left\{0, \|\,\texttt{position}\,\|_2 - \frac{\texttt{depth}}{2}\right\}.$$

– Для каждого расстояния с помощью полярного пространственного панораматора вычисляются векторы усиления $\mathbf{g'}_1$ и $\mathbf{g'}_2$ с использованием параметра `position`, а также параметров `width` и `height`, измененных с помощью описанной ниже функции модификации расширения в полярных координатах.

– Векторы усиления смешиваются для получения выходного вектора усиления $\mathbf{g}$, где $\mathbf{g}_i$ – вектор усиления для i-го громкоговорителя:

$$\mathbf{g}_i = \sqrt{\frac{\mathbf{g'}_{1,i}{}^2 + \mathbf{g'}_{2,i}{}^2}{2}}.$$

7.3.8.2.1 Функция модификации расширения в полярных координатах

Функция модификации расширения используется для изменения параметров ширины и высоты с учетом расстояния.

Она имеет следующие свойства:

– при $\texttt{distance} = 0$ расширение всегда составляет 360°;

– при $\texttt{distance} = 1$ используется исходное значение расширения;

– при $\texttt{distance} > 1$ расширение уменьшается с увеличением расстояния;

– при $0 < \texttt{distance} < 1$ расширение изменяется вокруг значения $\texttt{distance} = 0$ по более крутой кривой меньшими шагами.

Функция модификации расширения в зависимости от параметров `extent` и `distance` определяется следующим образом.

– Расширение в градусах линейно преобразуется в расширение вдоль оси x с минимальным размером `size`:

$$\texttt{min\_size} = 0{,}2;$$

$$\texttt{size} = \texttt{min\_size} + \frac{(1-\texttt{min\_size})\times\texttt{extent}}{360°}.$$

– Если образуется прямоугольный треугольник, его прилежащая сторона и противолежащая сторона – это расстояние. Затем образованный угол используется для определения нового значения расширения; оно рассчитывается для расстояния 1 и значения `distance`:

$$e_1 = 4 \times \frac{180}{\pi} \times \text{atan2}(\texttt{size}, 1);$$

$$e_d = 4 \times \frac{180}{\pi} \times \text{atan2}(\texttt{size}, \texttt{distance}).$$

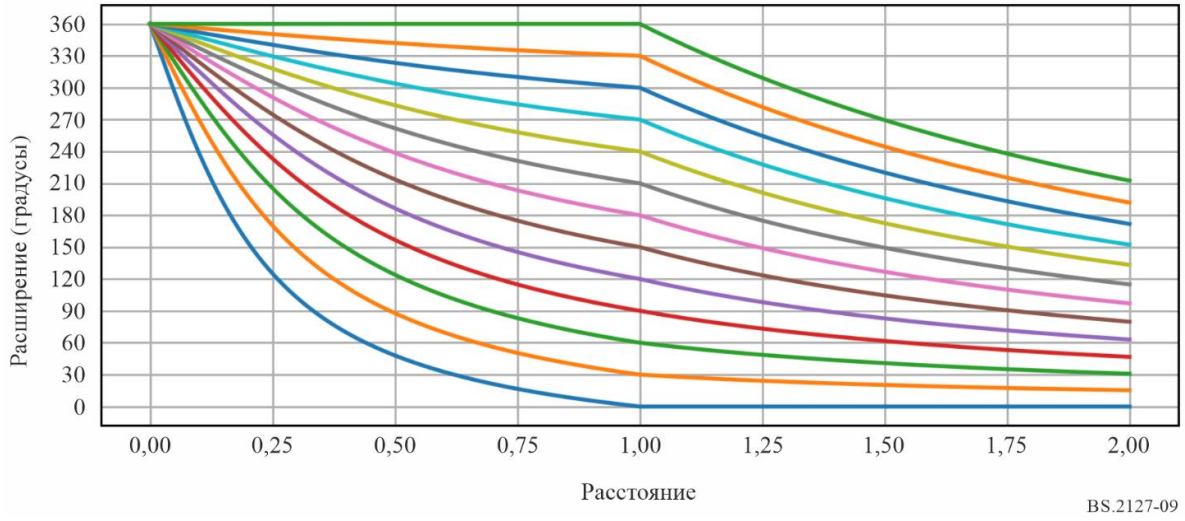
– Для обратного преобразования e_d в исходное значение расширения, когда $e_d = e_1$, применяется кусочно-линейная интерполяция:

$$\texttt{extent\_mod} = \begin{cases} \texttt{extent} \times \frac{e_d}{e_1} & e_d < e_1; \\ \texttt{extent} + (360° - \texttt{extent}) \times \frac{e_d - e_1}{360° - e_1} & e_d \geq e_1. \end{cases}$$

Это реализуется в модуле `core.objectbased.gain_calc.PolarExtentHandler.extent_mod`. Форма функции модификации расширения показана на рисунке 9.

РИСУНОК 9

Функция модификации расширения для распространенных источников в полярных координатах



ПРИМЕЧАНИЕ. – Каждая линия показывает, как изменяется выходное значение расширения в зависимости от расстояния для данного входного значения. При `distance` = 1 расширение не изменяется, поэтому, например, самая нижняя линия показывает, как результирующее расширение изменяется с расстоянием при входном значении расширения 0.

7.3.8.2.2 Полярный пространственный панораматор

Для обработки всего диапазона позиций и значений расширения, разрешенных в ADM, перед применением полярной весовой функции необходимо модифицировать размер. Используются следующие шаги.

– Модифицированные ширина и высота вычисляются как $max\{width, 5°\}$ и $max\{height, 5°\}$; они используются для получения вектора усиления с расширением g_s с помощью расширяющего панораматора, описанного в пункте 7.3.8.1, и полярной весовой функции, описанной ниже.

– Положение передается в точечный панораматор для получения вектора усиления точечного источника g_p.

Два вектора смешиваются для получения такого вектора g, чтобы для нулевых значений ширины и высоты использовались исключительно коэффициенты усиления точечного источника, но если ширина или высота превышают 5°, то используются исключительно пространственные коэффициенты усиления:

$$g_i = \sqrt{p g_{s,i}^2 + (1-p) g_{p,i}^2},$$

где:

$$p = clip\left(\frac{max(width, height)}{5}, 0{,}1\right).$$

Это необходимо для поддержки малых значений расширения. Здесь ненулевая часть пространственной функции должна быть достаточно большой, чтобы охватить несколько точек выборки для получения плавных коэффициентов усиления, а это предполагает такой минимальный разброс, который может оказаться больше желаемой величины.

Это реализуется в модуле `core.objectbased.extent.PolarExtentPanner.calc_pv_spread`.

7.3.8.2.3 Полярная весовая функция

Весовая функция для пространственного рендеринга в полярных координатах параметризуется трехмерным декартовым вектором `position`, а также углами `width` и `height` в градусах. Поскольку компонент расстояния для позиции не используется, это можно рассматривать как направление.

Весовая функция выглядит следующим образом.

– Вычисляется матрица поворота, которая отображает положение $\{0,1,0\}$ (непосредственно перед слушателем) на положение источника. Эта матрица принимает форму поворота вокруг точки $\{1,0,0\}$ с последующим поворотом вокруг точки $\{0,0,1\}$. Это реализуется в модуле `core.objectbased.extent.calc_basis`.

– Если высота больше ширины, то для упрощения вычисления система координат перевертывается, поскольку весовая функция для источника шириной w и высотой h будет такой же, как весовая функция для источника шириной h и высотой w, повернутого на 90° вокруг исходного положения. Для этого переменные `width` и `height` меняются местами, как и строки x и z матрицы поворота. См., например, рисунки 10 и 11, на которых одна и та же форма повернута на 90° (без учета деформации, вызванной используемым методом проецирования).

– Теперь приближенная весовая функция равна 1 внутри максимально скругленного прямоугольника `width` × `height` (стадион) в пространстве азимут – угол места с несколькими модификациями.

- Скругления представляют собой дуги окружности в декартовом пространстве, так как вес рассчитывается на основе угла между двумя векторами с началом в их центре. Когда `width` = `height`, весовая функция представляет собой окружность.
- При `width` $> 180°$ ширина увеличивается так, что когда она достигает $360°$ скругленные части полностью перекрываются, образуя "полосу", в которой весовая функция имеет одинаковое значение для всех позиций на одной и той же высоте. См. рисунки 12 и 13.
- По краям весовой функции добавляется затухание; вес падает с 1 до 0, когда угловое расстояние от величины расширения достигает 10°.

Эта функция реализуется в модуле `core.objectbased.extent.PolarExtentPanner.get_weight_func`.

РИСУНОК 10

Полярная весовая функция для `width` **= 90° и** `height` **= 30°**

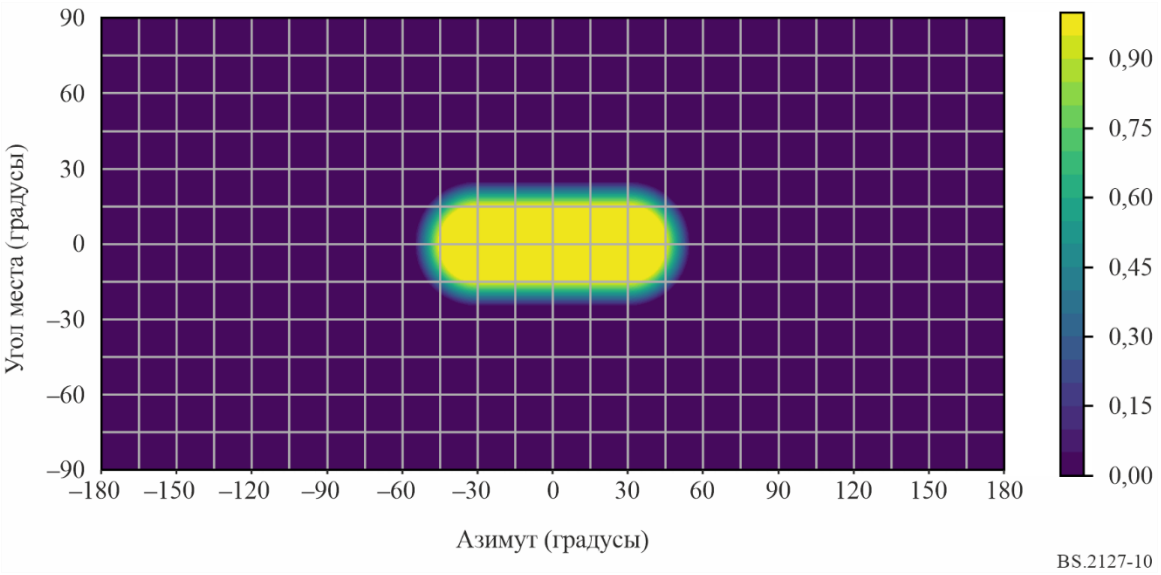


РИСУНОК 11

Полярная весовая функция для `width` **= 30° и** `height` **= 90°**

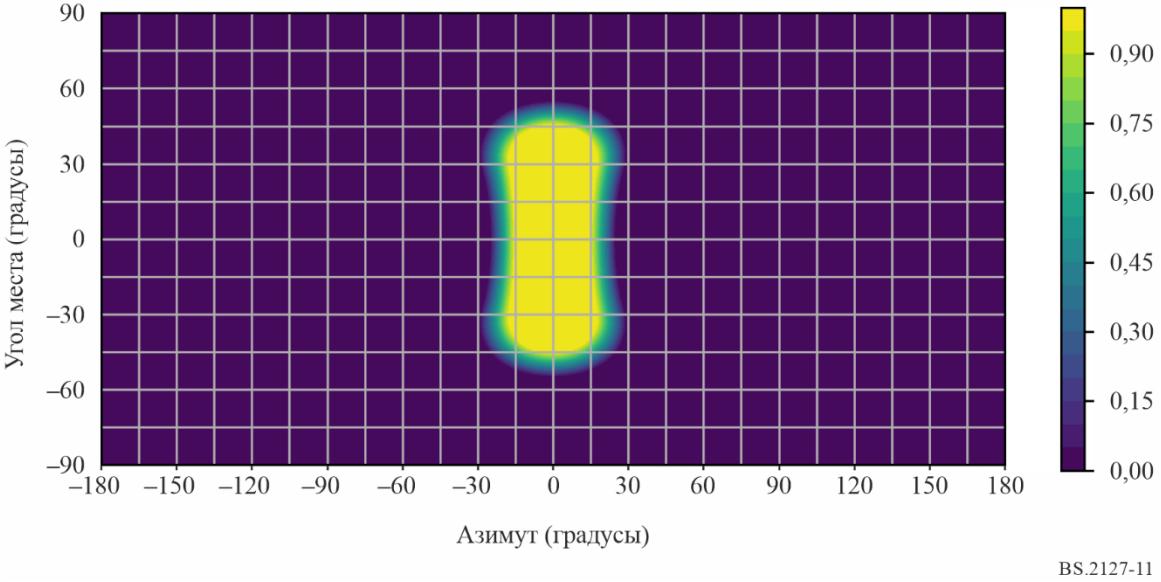


РИСУНОК 12

Полярная весовая функция для `width` = **300°** **и** `height` = **30°**

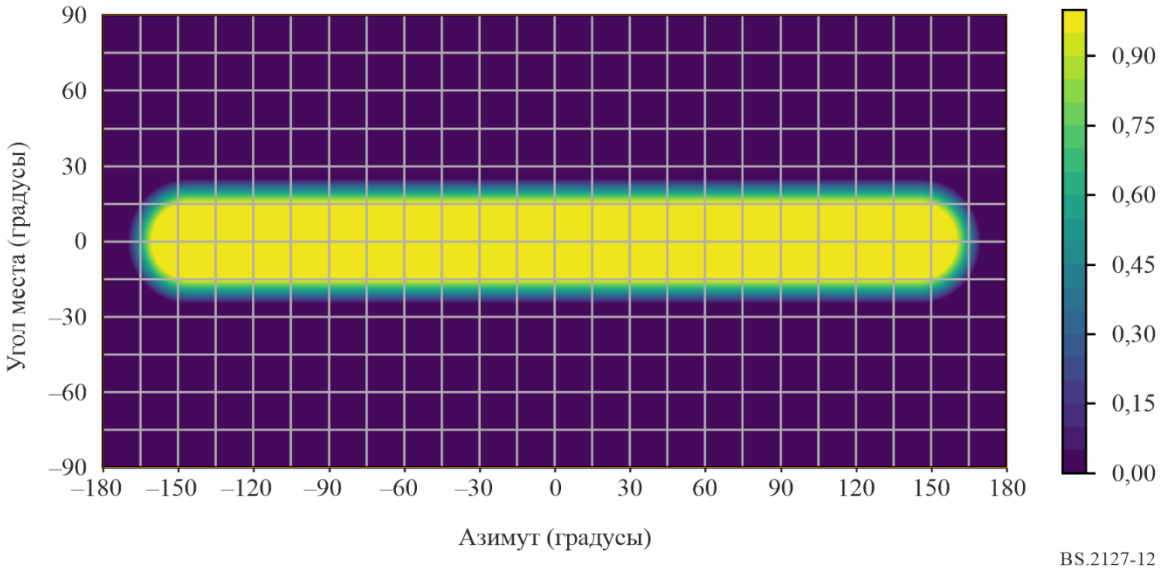
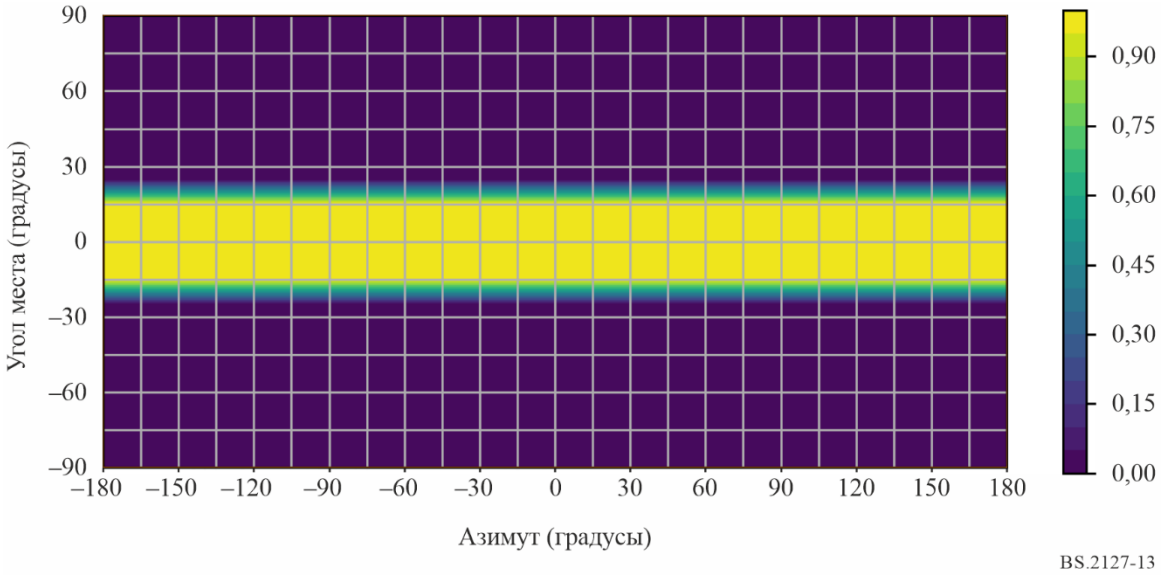


РИСУНОК 13

Полярная весовая функция для `width` = **360°** **и** `height` = **30°**



7.3.9 Позиции громкоговорителей в декартовой системе координат

Для того чтобы использовать точечный декартов панораматор, описанный в пункте 7.3.10, необходимо найти позицию каждого громкоговорителя в схеме расположения.

Интерфейс к этому компоненту является следующим:

```
vector<CartesianPosition> positions_for_layout(Layout layout).
```

Прежде всего, таблица позиций, соответствующих `layout.name`, приведена в пункте 11.2.

Для каждого канала `channel` в `layout.channels`, параметры `x`, `y` и `z` выходной позиции `CartesianPosition` определяются следующим образом.

– Если `channel.name` равно `M+SC` или `M−SC`, то:

$\{x, y, z\}$ = `point_polar_to_cart(channel.polar_position.azimuth, 0,1)`.

Следует отметить, что это предполагает абсолютную точность преобразования `point_polar_to_cart`. На практике позиции следует изменить таким образом, чтобы:

- $z = 0$;
- координаты y обоих экранных громкоговорителей были идентичными;
- координаты x обоих экранных громкоговорителей были точно симметричны относительно 0.

– В противном случае значения берутся из строки таблицы с именем `channel.name`.

Это реализуется в модуле `core.allocentric`.

7.3.10 Точечный панораматор в декартовой системе координат

Алгоритм декартового точечного панорамирования состоит из трехмерного расширения концепции панораматора "с двойным балансом", которая широко используется в производстве 5,1- и 7,1-канального объемного звука.

Входные данные панораматора состоят из положения объекта $[p_{ox}, p_{oy}, p_{oz}]$ и положений N выходных громкоговорителей, все в декартовых координатах. Обозначим положение j-го громкоговорителя $[p_{sx}(j),\ p_{sy}(j),\ p_{sz}(j)]$.

Что касается схемы расположения громкоговорителей, то для точечного панораматора необходимо соблюдение следующих условий, чтобы фантомное изображение объекта можно было точно разместить в любом месте помещения.

– Громкоговорители должны быть сгруппированы в одну или несколько дискретных плоскостей в измерении z.

– Громкоговорители в каждой плоскости должны быть сгруппированы в один или несколько дискретных рядов в измерении y.

– В любом ряду, где $-1 < y < 1$ (то есть в любом ряду, не пересекающем переднюю или заднюю стену помещения), громкоговорители должны находиться в положениях $x = 1$ и $x = -1$.

– Каждая позиция громкоговорителя должна находиться на поверхности куба помещения, то есть на полу, на потолке или на стенах.

– Позиции, соответствующие этим условиям, определяются с помощью процедуры, приведенной в пункте 7.3.9.

Приблизительные коэффициенты усиления громкоговорителей для данной позиции источника определяются следующим образом.

– Находят уровни громкоговорителей выше и ниже источника и вычисляют коэффициент усиления z для обоих этих уровней на основе положений уровней z и источника.

– В каждом из найденных уровней находят ряд громкоговорителей перед источником и за ним и рассчитывают коэффициент усиления y для каждого из этих рядов в зависимости от положения y рядов и источника.

– В каждом из найденных рядов находят пару громкоговорителей слева и справа от источника и вычисляют усиление по оси x для каждого из этих громкоговорителей в зависимости от положения x громкоговорителей и источника.

Будет выбрано до восьми громкоговорителей; для каждого из них коэффициент усиления равен $x \times y \times z$; другие громкоговорители имеют нулевое усиление.

Точная спецификация алгоритма приведена ниже; для каждого громкоговорителя j вычисляется коэффициент усиления $g^{point}(j_x, j_y, j_z)$. Следует отметить, что каждую ось можно отделить; также полезно заметить, что $g^{point}(x, y, z) = g^{point_x}(x) \times g^{point_y}(y) \times g^{point_z}(z)$ и что в качестве промежуточных значений в алгоритме фигурируют три независимых коэффициента усиления.

```
epsilon = 0.001 //малая положительная константа

//simplification: используйте объектно ориентированную систему координат, так чтобы
объект
//всегда находился в системе координат
for (j = 1 to N)
{
  p_sx(j) -= p_ox
  p_sy(j) -= p_oy
  p_sz(j) -= p_oz
}

for (j = 1 to N)
{
  //коэффициент усиления по оси Z
  z_this = p_sz(j)
  //найти громкоговорители в другой плоскости по другую сторону от объекта
  if (z_this >= 0) {
    z_other = max({p_sz : p_sz < z_this})
  } else {
    z_other = min({p_sz : p_sz > z_this})
  }
  if (isempty(z_other)) {
    gz = 1.0
  } else if (sign(z_other) == sign(z_this)) {
    gz = 0.0
  } else {
    gz = cos(z_this / (z_other - z_this) * pi /2)
  }

  //коэффициент усиления по оси Y
  //среди громкоговорителей в этой плоскости...
  p_sx_plane = p_sx({i:abs(p_sz(i) - z_this) < epsilon})
  p_sy_plane = p_sy({i:abs(p_sz(i) - z_this) < epsilon})
  y_this = p_sy(j)
  //...найти громкоговорители в ближайшем ряду по другую сторону от объекта
  if (y_this >= 0) {
    y_other = max({p_sy_plane : p_sy_plane < y_this})
  } else {
    y_other = min({p_sy_plane : p_sy_plane > y_this})
  }
  if isempty(y_other) {
    gy = 1.0
  } else if (sign(y_other) == sign(y_this)) {
    gy = 0.0
  } else {
    gy = cos(y_this / (y_other - y_this) * pi /2)
  }

  //коэффициент усиления по оси X
  //среди громкоговорителей в этой плоскости и в этом ряду...
  p_sx_row = p_sx_plane({i:abs(p_sy_plane(i) - y_this) < epsilon})
  x_this = p_sx(j)
  //найти громкоговорители в ближайшем столбце
  if (x_this >= 0) {
    x_other = max({p_sx_row : p_sx_row < x_this})
  } else {
    x_other = min({p_sx_row : p_sx_row > x_this})
  }
  if (isempty(x_other)) {
    gx = 1.0
  } else if (sign(x_other) == sign(x_this)) {
    gx = 0.0
```

```
    } else {
      gx = cos(x_this / (x_other - x_this) * pi /2)
    }
    g_point(j) = gx * gy * gz
  }
```

Следует отметить, что ненулевое усиление будут иметь не более восьми громкоговорителей и что сумма квадратов коэффициентов усиления громкоговорителя всегда равна 1, поэтому операция панорамирования является энергосберегающей.

Это реализуется в модуле `core.point_source.AllocentricPanner`.

7.3.11 Декартов пространственный панораматор

Пространственный панораматор предназначен для вычисления коэффициентов усиления для каждого громкоговорителя в схеме расположения выходных громкоговорителей с учетом положения и расширения объектов. Цель расширения – сделать так, чтобы объект казался крупнее, так что при максимальном расширении объект заполняет собой все помещение, а при нулевом представляется точечным объектом.

Для этого пространственный панораматор рассматривает сетку из множества виртуальных источников в помещении. Каждый виртуальный источник возбуждает громкоговорители точно так же, как любой объект, представляемый с помощью точечного панораматора. При заданных значениях положения и расширения объекта пространственный панораматор определяет, какой и сколько из этих виртуальных источников вносят свой вклад.

Для того чтобы рассчитать коэффициенты усиления для разнесенного объекта, необходимы следующие шаги. Более подробное объяснение каждого из шагов приводится в одном из следующих подразделов.

1 Предварительно смасштабировать параметры расширения.

2 Рассчитать точечные коэффициенты усиления для всех виртуальных источников.

3 Объединить все коэффициенты усиления от виртуальных источников в помещении, чтобы получить внутренние пространственные коэффициенты усиления.

4 Объединить все коэффициенты усиления от виртуальных источников по краям помещения, чтобы получить краевые пространственные коэффициенты усиления.

5 Объединить внутренние и краевые пространственные коэффициенты усиления, чтобы получить окончательные пространственные коэффициенты усиления.

6 Объединить окончательные пространственные коэффициенты усиления с точечными коэффициентами усиления для объекта.

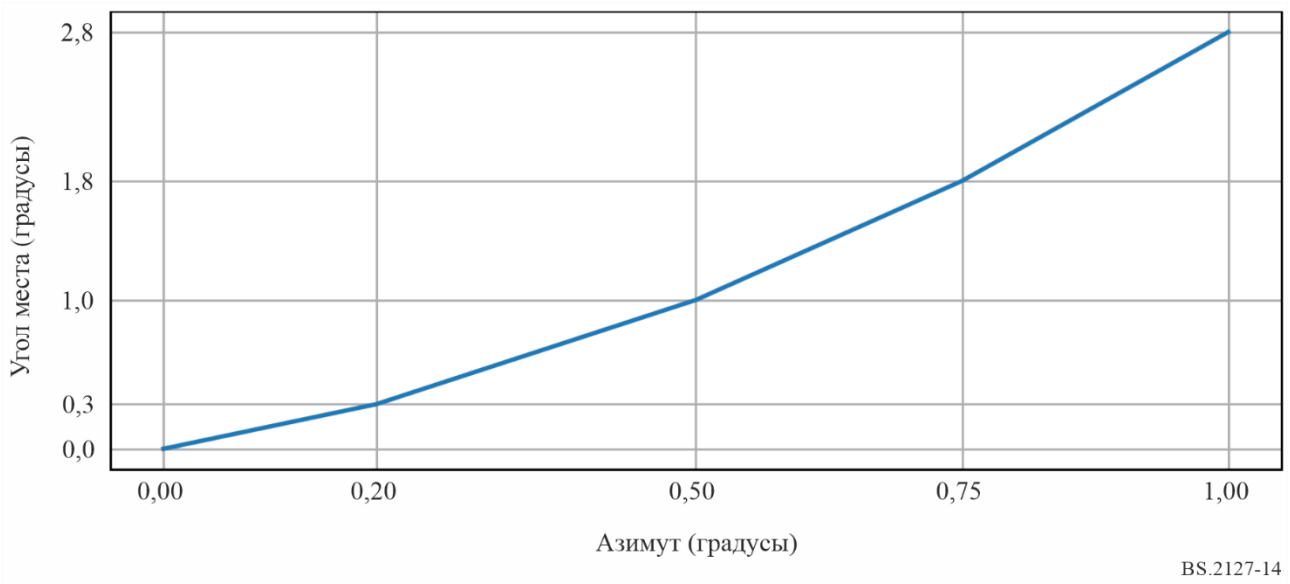
Декартов пространственный панораматор реализуется в модуле `core.objectbased.allo_extent.get_gains`.

7.3.11.1 Предварительное масштабирование параметров расширения

Перед вычислением любых коэффициентов усиления значения параметров расширения масштабируются, чтобы весовая функция источника работала более интуитивно понятно. Пользователь воспринимает значения $s \in [0{,}1]$, которые преобразуются в фактическое расширение, используемое алгоритмом в диапазоне $[0;\ 2{,}8]$. Преобразование выполняется кусочно-линейной функцией, определенной парами значений $(0{,}0), (0{,}2;\ 0{,}3), (0{,}5; 1{,}0), (0{,}75; 1{,}8), (1;\ 2{,}8)$, которые показаны на рисунке 14. Максимальное значение 2,8 гарантирует, что когда установлено максимальное расширение (1,0), объект действительно охватывает все помещение. В дальнейшем переменные $\hat{s}_x, \hat{s}_y, \hat{s}_z$ ссылаются на входные значения расширения после преобразования.

РИСУНОК 14

Кусочно-линейное преобразование между параметрами расширения ADM и внутренними значениями расширения алгоритма



Для сохранения желаемого поведения при экстремальных значениях расширения применяются минимальные значения $\hat{s}_x, \hat{s}_y, \hat{s}_z$ следующим образом:

$$s_x = \max\left(\hat{s}_x, \frac{2}{N_x - 1}\right),\ s_y = \max\left(\hat{s}_y, \frac{2}{N_y - 1}\right),\ s_z = \max\left(\hat{s}_z, \frac{2}{N_z - 1}\right).$$

Эти ограниченные значения s_x, s_y, s_z используются во всем алгоритме.

7.3.11.2 Расчет коэффициентов усиления виртуальных источников

Сетка виртуальных источников определяется как статическая прямоугольная равномерная сетка из $N_x \times N_y \times N_z$ узлов. Эта сетка охватывает диапазон позиций $[-1,1]$ в каждом измерении. Плотность устанавливается таким образом, чтобы в типовой схеме расположения между громкоговорителями находилось несколько источников. Эмпирическое тестирование показало, что подходящая сетка виртуальных источников создается при $N_x = N_y = N_z = 40$[1]. Для обозначения возможных координат виртуальных источников используются символы (x_s, y_s, z_s). Каждый виртуальный источник создает набор коэффициентов усиления $g_j^{point}(x_s, y_s, z_s)$ для каждого громкоговорителя $j = 1, \ldots, N_j$ схемы расположения в соответствии с алгоритмом декартового точечного панораматора, описанным в пункте 7.3.10. Следует отметить, что если какие-либо громкоговорители исключены из схемы расположения алгоритмом исключения зоны (см. пункт 7.3.5), то при расчете коэффициентов усиления используется сокращенная схема расположения громкоговорителей.

7.3.11.3 Объединение коэффициентов усиления виртуальных источников в помещении

Положение и расширение объекта $(x_o, y_o, z_o, s_x, s_y, s_z)$ используются для вычисления набора весовых коэффициентов, которые определяют вклад каждого виртуального источника в конечный коэффициент усиления[2]. Весовые коэффициенты каждого виртуального источника обозначаются

---

[1] Для схем расположения без громкоговорителей нижнего уровня диапазон виртуальных источников в измерении Z органичен диапазоном $[0,1]$, а рекомендуемое значение N_z равно 20.

[2] Для схем расположения без громкоговорителей нижнего уровня в алгоритме расширения в качестве положения объектов в измерении Z используется $z_o = \max(p_{oz}, 0)$. В противном случае $z_o = p_{oz}$. Для всех схем расположения в алгоритме расширения используется та же позиция X, Y, что и в точечном панораматоре (то есть $y_o = p_{oy}, x_o = p_{ox}$).

$w(x_s, y_s, z_s, x_o, y_o, z_o, s_x, s_y, s_z)$ и используются для масштабирования точечных коэффициентов усиления для каждого виртуального источника. После взвешивания коэффициенты усиления всех виртуальных источников суммируются для получения внутренних пространственных коэффициентов усиления:

$$g_j^{inside}(x_o, y_o, z_o, s_x, s_y, s_z) = \sum_{x_s,y_s,z_s} w\,(x_s, y_s, z_s, x_o, y_o, z_o, s_x, s_y, s_z) \times g_j^{point}(x_s, y_s, z_s).$$

Однако пространственный алгоритм комбинирует коэффициенты усиления виртуальных источников таким образом, что они изменяются в зависимости от расширения объекта. В общем виде это можно описать следующей формулой:

$$g_j^{inside}(x_o, y_o, z_o, s_x, s_y, s_z) = \left[\sum_{x_s,y_s,z_s} \left[w(x_s, y_x, z_s, x_o, y_o, z_o, s_x, s_y, s_y) \times g_j^{point}(x_s, y_s, z_s)\right]^p\right]^{\frac{1}{p}}.$$

Зависящий от расширения показатель степени p управляет плавностью усиления между громкоговорителями. Это обеспечивает однородный рост объекта при малых значениях s и правильное распределение энергии по всем направлениям при больших значениях s. Для того чтобы вычислить p, сначала выполняется сортировка $\{s_x, s_y, s_z\}$ в порядке убывания с получением упорядоченной триады $\{s_1, s_2, s_3\}$. Затем эту триаду можно объединить, чтобы получить эффективное расширение:

$$s_{eff} = \frac{6}{9}s_1 + \frac{2}{9}s_2 + \frac{1}{9}s_3.$$

Для схем с одной плоскостью расположения громкоговорителей, таких как 0+5+0, или в случае если применение исключения зоны приводит к сокращению схем расположения до одной плоскости, сначала выполняется сортировка $\{s_x, s_y\}$ в порядке убывания с получением упорядоченной пары $\{s_1, s_2\}$ и определяется:

$$s_{eff} = \frac{3}{4}s_1 + \frac{1}{4}s_2.$$

Для схем расположения 0+2+0 (стерео) или в случае если исключение зоны сводит набор громкоговорителей к одному ряду $s_{eff} = s_x$.

Затем эффективное расширение используется для вычисления кусочно-определенного показателя степени:

$$p = \begin{cases} 6 & \text{при } s_{eff} \le 0{,}5; \\ 6 - 4 \times \frac{s_{eff} - 0{,}5}{s_{\max} - 0{,}5} & \text{в других случаях,} \end{cases}$$

где $s_{\max} = 2{,}8$, так что при максимальном значении s $p = 2$.

Весовая функция также может обрабатывать каждую ось отдельно, и если используются отдельные весовые функции, то вычисление расширения в целом упрощается:

$$w(x_s, y_s, z_s, x_o, y_o, z_o, s_x, s_y, s_z) = w_x(x_s, x_o, s_x) w_y(y_s, y_o, s_y) w_z(z_s, z_o, s_z).$$

Выбранные функции выглядят как нечто среднее между кругами и квадратами (или в 3D-варианте – между сферами и кубами):

$$\begin{aligned} w_x(p, o, s) = w_y(p, o, s) &= 10^{-\min\left(\left[\frac{3}{2}\left(\frac{p-o}{2s}\right)\right]^4;\, 6{,}5\right)}; \\ w_z(p, o, s) &= 10^{-\min\left(\left[\frac{3}{2}\left(\frac{p-o}{s}\right)\right]^4;\, 6{,}5\right)} \times \cos\left(s\frac{3\pi}{7}\right). \end{aligned}$$

Это означает, что g_j^{inside} можно упростить до:

$$g_j^{inside}(x_o, y_o, z_o, s_x, s_y, s_z) = f_j^x(x_o, s_x) f_j^y(y_o, s_y) f_j^z(z_o, s_z),$$

где:

$$\begin{aligned} f_j^x(x_o, s_x) &= \sum_{x_s} \left[g_j^{point_x}(x_s) w_x(x_s, x_o, s_x) \right]^p ; \\ f_j^y(y_o, s_y) &= \sum_{y_s} \left[g_j^{point_y}(y_s) w_y(y_s, y_o, s_y) \right]^p ; \\ f_j^z(z_o, s_z) &= \sum_{z_s} \left[g_j^{point_z}(z_s) w_z(z_s, z_o, s_z) \right]^p . \end{aligned}$$

Следует отметить, что для схем, ограниченных одной плоскостью расположения громкоговорителей, $f_j^z(z_o, s_z) = 1$, а для одного ряда громкоговорителей $f_j^z(z_o, s_z) = f_j^y(y_o, s_y) = 1$.

Кроме того, очень малые значения $f_j(c, s)$ $(10^{-6{,}5})$ округляются до нуля во избежание потери значимости в отдельных реализациях.

К g_j^{inside} применяется нормализация:

$$\tilde{g}_j^{inside} = \begin{cases} \dfrac{g_j^{inside}}{\sqrt{\sum_n \left[g_n^{inside} \right]^2}} & \sqrt{\sum_n \left[g_n^{inside} \right]^2} > tol; \\ 0 & \text{в других случаях,} \end{cases}$$

где $tol = 10^{-5}$.

7.3.11.4 Объединение краевых коэффициентов усиления

Еще одно изменение заключается в том, что по эстетическим соображениям важно иметь режим, в котором противоположные громкоговорители не возбуждаются. Это достигается путем использования виртуальных источников, расположенных только по краям. Для обработки определенных схем расположения громкоговорителей как особых случаев:

- $dim = 1$ для схем расположения с единственным рядом громкоговорителей после применения исключения зоны (например, `0+2+0`);
- $dim = 2$ для схем с единственной плоскостью расположения громкоговорителей после применения исключения зоны (например, `0+5+0`);
- $dim = 4$ для схем с более чем двумя различающимися по высоте плоскостями расположения громкоговорителей после применения исключения зоны (например, `3+7+0` и `9+10+3`); и
- $dim = 3$ в других случаях.

Тогда краевые коэффициенты усиления:

$$\begin{aligned} g_j^{bound}(x_o, y_o, z_o, s_x, s_y, s_z) &= b_j^{floor}(z_o, s_z) f_j^x(x_o, s_x) f_j^y(y_o, s_y) \\ &+ b_j^{ceil}(z_o, s_z) f_j^x(x_o, s_x) f_j^y(y_o, s_y) \\ &+ b_j^{left}(x_o, s_x) f_j^y(y_o, s_y) f_j^z(z_o, s_z) \\ &+ b_j^{right}(x_o, s_x) f_j^y(y_o, s_y) f_j^z(z_o, s_z) \\ &+ b_j^{front}(y_o, s_y) f_j^x(x_o, s_x) f_j^z(z_o, s_z) \\ &+ b_j^{back}(y_o, s_y) f_j^x(x_o, s_x) f_j^z(z_o, s_z), \end{aligned}$$

где:

$$b_j^{floor}(z_o, s_z) = \begin{cases} \left[g_j^{point}(z_s = -1{,}0)w(-1{,}0; z_o; s_z)\right]^p & dim = 4; \\ 0 & \text{в других случаях;} \end{cases}$$

$$b_j^{ceil}(z_o, s_z) = \begin{cases} \left[g_j^{point}(z_s = 1{,}0)w(1{,}0; z_o; s_z)\right]^p & dim \geq 3; \\ 0 & \text{в других случаях;} \end{cases}$$

$$b_j^{left}(x_o, s_x) = \left[g_j^{point}(x_s = -1{,}0)w(-1{,}0; x_o; s_x)\right]^p;$$

$$b_j^{right}(x_o, s_x) = \left[g_j^{point}(x_s = 1{,}0)w(1{,}0; x_o; s_x)\right]^p;$$

$$b_j^{front}(y_o, s_y) = \begin{cases} \left[g_j^{point}(y_s = 1{,}0)w(1{,}0; y_o; s_y)\right]^p & dim > 1; \\ 0 & \text{в других случаях;} \end{cases}$$

$$b_j^{back}(y_o, s_y) = \begin{cases} \left[g_j^{point}(y_s = -1{,}0)w(-1{,}0; y_o; s_y)\right]^p & dim > 1; \\ 0 & \text{в других случаях.} \end{cases}$$

7.3.11.5 Объединение внутренних и краевых коэффициентов усиления

Теперь краевые коэффициенты усиления необходимо объединить с внутренними, поэтому для всех виртуальных источников внутри помещения вводится коэффициент затухания, при этом величина затухания равна доле объекта вне помещения.

Это дает:

$$g_j^{extent} = \left[\tilde{g}_j^{bound} + \left(\mu \times \tilde{g}_j^{inside}\right)\right]^{\frac{1}{p}},$$

где:

$$d_{bound} = \begin{cases} \min(x_o + 1{,}1 - x_o) & dim = 1; \\ \min(x_o + 1{,}1 - x_o, y_o + 1{,}1 - y_o) & dim = 2; \\ \min(x_o + 1{,}1 - x_o, y_o + 1{,}1 - y_o, z_o + 1, z_o - 1) & \text{в других случаях;} \end{cases}$$

$$\mu = \begin{cases} h(x_o, s_x)^3 & dim = 1; \\ h(x_o, s_x)h\left(y_o, s_y\right)^{\frac{3}{2}} & dim = 2; \\ h(x_o, s_x)h\left(y_o, s_y\right)h(z_o, s_z) & \text{в других случаях;} \end{cases}$$

и $h(c, s)$ – функция затухания для одного измерения;

$$h(c, s) = \begin{cases} \left[\frac{\max(2s;\, 0{,}4)^3}{0{,}16 \times 2s}\right]^{\frac{1}{3}} & d_{bound} \geq s \wedge d_{bound} \geq 0{,}4; \\ \left[\frac{d_{bound}}{2}\left(\frac{d_{bound}}{0{,}4}\right)^2\right]^{\frac{1}{3}} & \text{в других случаях.} \end{cases}$$

Когда часть разнесенного объекта начинает перемещаться за пределы помещения, все виртуальные источники внутри объекта начинают затухать, кроме тех, которые находятся по краям. Когда объект достигает края, влияние на пространственные коэффициенты усиления оказывают только краевые коэффициенты усиления. d_{bound} – это минимальное расстояние до края.

К g_j^{extent} применяется нормализация:

$$\widetilde{g}_j^{extent} = \begin{cases} \frac{g_j^{extent}}{\sqrt{\sum_n [g_n^{extent}]^2}} & \sqrt{\sum_n [g_n^{extent}]^2} > tol; \\ 0 & \text{в других случаях.} \end{cases}$$

7.3.11.6 Объединение пространственных и точечных коэффициентов усиления

Затем значения пространственных коэффициентов усиления объединяются с точечными коэффициентами усиления и применяется перекрестное затухание между ними как функция расширения:

$$g_j^{total} = \left(\alpha \times g_j^{point}(x_o, y_o, z_o)\right) + \left(\beta \times \tilde{g}_j^{extent}\right),$$

где:

$$\alpha = \begin{cases} \cos\left(\frac{s_{eff}}{s_{fade}} \times \frac{\pi}{2}\right) & s_{eff} < s_{fade}; \\ 0 & \text{в других случаях;} \end{cases}$$

$$\beta = \begin{cases} \sin\left(\frac{s_{eff}}{s_{fade}} \times \frac{\pi}{2}\right) & s_{eff} < s_{fade}; \\ 1 & \text{в других случаях;} \end{cases}$$

и $s_{fade} = 0{,}2$.

Это гарантирует плавное панорамирование и плавное нарастание объекта, обеспечивая хороший переход на всем пути между наименьшим и наибольшим возможными значениями расширения.

Наконец, к коэффициентам усиления в последний раз применяется нормализация:

$$G_j^S = \begin{cases} \frac{g_j^{total}}{\sqrt{\sum_n \left[g_n^{total}\right]^2}} & \sqrt{\sum_n \left[g_n^{total}\right]^2} > tol; \\ 0 & \text{в других случаях.} \end{cases}$$

7.3.12 Исключение зоны в полярных координатах

Применение исключения зоны осуществляется путем понижающего микширования вектора усиления для громкоговорителей, созданного ранее в калькуляторе усиления, во избежание вывода на громкоговорители, расположенные в исключенной зоне. Этот процесс можно разделить на две части – определение громкоговорителей, находящихся в исключенной зоне, как описано в пункте 7.3.12.1, и вычисление понижающего микширования для ухода от исключенных громкоговорителей, как описано в пункте 7.3.12.2.

Как при выборе исключенных громкоговорителей, так и при расчете матрицы понижающего микширования учитывается только номинальное положение громкоговорителей, поэтому небольшие изменения в позициях громкоговорителей не влияют на поведение функции исключения зоны.

7.3.12.1 Выбор исключенных громкоговорителей

Выбор громкоговорителей осуществляется путем обработки списка объектов `ExclusionZone` с получением для каждого громкоговорителя двоичного флага, который принимает значение true (истинно), если громкоговоритель находится в любой из зон исключения и поэтому должен быть исключен.

Для объектов `CartesianZone`, чтобы определить, находится ли громкоговоритель в такой зоне, используется следующее выражение, где $\{x, y, z\}$ – номинальная позиция громкоговорителя, преобразованная из области с радиусом 1 в полярных координатах:

$$\begin{aligned} &\mathtt{minX} - \epsilon < \quad x < \mathtt{maxX} + \epsilon; \\ \wedge\, &\mathtt{minY} - \epsilon < \quad y < \mathtt{maxY} + \epsilon; \\ \wedge\, &\mathtt{minZ} - \epsilon < \quad z < \mathtt{maxZ} + \epsilon, \end{aligned}$$

где $\epsilon = 10^{-6}$ – страховочный запас, учитывающий ошибки округления при преобразовании между полярными и декартовыми координатами.

Для объектов PolarZone, чтобы определить, находится ли громкоговоритель в такой зоне, используется следующее выражение, где φ и θ – номинальные азимут и угол места громкоговорителя:

$$\begin{array}{ll} & \mathrm{minElevation} - \epsilon < \theta < \mathrm{maxElevation} + \epsilon \\ \wedge & \left(\begin{array}{ll} & |\theta| > 90 - \epsilon \\ \vee & \mathrm{IAR}(\varphi, \mathrm{minAzimuth}, \mathrm{maxAzimuth}, \epsilon) \end{array} \right), \end{array}$$

IAR – это функция inside_angle_range; см. пункт 6.2.

Угол места громкоговорителя всегда должен находиться в допустимом диапазоне, а азимут – только если абсолютное значение угла места меньше 90°.

Это реализуется в модуле
core.objectbased.gain_calc.ZoneExclusionHandler.get_excluded.

7.3.12.2 Понижающее микширование для исключенных громкоговорителей

Как только определяются громкоговорители, расположенные в пределах зоны исключения, создается матрица понижающего микширования для отвода усиления от этих громкоговорителей.

Объект панораматора исключения зоны связывает с каждым громкоговорителем схемы расположения список групп выходных громкоговорителей. Матрица понижающего микширования такова, что усиление от исключенного громкоговорителя направляется на все неисключенные громкоговорители первой группы, в которой имеются неисключенные громкоговорители. Эта функциональная возможность подробно описывается в следующих двух подразделах.

В качестве примера в таблице 3 представлены группы громкоговорителей в формате 4+5+0. Первая строка показывает, что если громкоговоритель M+030 исключен, то предназначенный для него выходной сигнал будет направлен на громкоговоритель M+000, если он не исключен, в случае чего сигнал будет направлен на громкоговоритель M−030 и. т. д. до громкоговорителя U−110.

Более сложным примером, в котором определенную роль играет группирование, является позиция M+000. Когда она исключена, этот канал распределяется между неисключенными громкоговорителями в позиции $\{\mathrm{M}+030, \mathrm{M}-030\}$, если оба эти громкоговорителя не исключены, в случае чего канал будет направлен к неисключенным громкоговорителям в позиции $\{\mathrm{M}+110, \mathrm{M}-110\}$, и т. д.

ТАБЛИЦА 3

Пример связи громкоговорителей для схемы расположения 4+5+0

Вход	Выходные группы
M + 030	{M + 030}, {M + 000}, {M − 030}, {M + 110}, {M − 110}, {U + 030}, {U − 030}, {U + 110}, {U − 110}
M − 030	{M − 030}, {M + 000}, {M + 030}, {M − 110}, {M + 110}, {U − 030}, {U + 030}, {U − 110}, {U + 110}
M + 000	{M + 000}, {M + 030 , M − 030}, {M + 110 , M − 110}, {U + 030 , U − 030}, {U + 110 , U − 110}
M + 110	{M + 110}, {M − 110}, {M + 030}, {M + 000}, {M − 030}, {U + 110}, {U − 110}, {U + 030}, {U − 030}
M − 110	{M − 110}, {M + 110}, {M − 030}, {M + 000}, {M + 030}, {U − 110}, {U + 110}, {U − 030}, {U + 030}
U + 030	{U + 030}, {U − 030}, {U + 110}, {U − 110}, {M + 030}, {M + 000}, {M − 030}, {M + 110}, {M − 110}
U − 030	{U − 030}, {U + 030}, {U − 110}, {U + 110}, {M − 030}, {M + 000}, {M + 030}, {M − 110}, {M + 110}
U + 110	{U + 110}, {U − 110}, {U + 030}, {U − 030}, {M + 110}, {M − 110}, {M + 030}, {M + 000}, {M − 030}
U − 110	{U − 110}, {U + 110}, {U − 030}, {U + 030}, {M − 110}, {M + 110}, {M − 030}, {M + 000}, {M + 030}

Эта функциональная возможность реализуется в модулях
core.objectbased.zone.ZoneExclusionDownmix и core.objectbased.gain_calc.
ZoneExclusionHandler.

7.3.12.2.1 Определение групп громкоговорителей

Во время инициализации для каждого громкоговорителя определяются группы вывода.

Для каждого входного громкоговорителя каждому выходному громкоговорителю назначается кортеж чисел с плавающей запятой, называемый *ключом*. Группы вывода состоят из выходных громкоговорителей, отсортированных по ключу и собранных в группы с аналогичными ключами. Таким образом порядок и группировка определяются главным образом функцией ключа.

Ключ входного и выходного громкоговорителей состоит из четырех ключей:

– целочисленного приоритета уровня, который равен нулю, если оба громкоговорителя находятся на одном и том же уровне, и увеличивается при разделении входного и выходного уровней, при этом предпочтение отдается громкоговорителю, находящемуся на более высоком уровне. Приоритеты уровней берутся из таблицы 4;

– целочисленного приоритета передний/задний (front/back priority), который ниже, если как входной, так и выходной громкоговорители находятся спереди, сбоку или позади слушателя. С учетом компонента y номинального положения входного и выходного громкоговорителей в полярных координатах после их преобразования в декартовы координаты y_i и y_o этот расчет выполняется следующим образом:

$$|\mathrm{sgn}y_i - \mathrm{sgn}y_o|;$$

– векторного расстояния между номинальными позициями двух громкоговорителей, чтобы отдавать предпочтение меньшим перемещениям;

– абсолютной разности номинальных координат y между двумя громкоговорителями, чтобы разделить группы, не симметричные относительно плоскостей yz или xz.

ТАБЛИЦА 4

Приоритеты уровней между двумя громкоговорителями

Входной уровень	Нижний	Средний	Верхний	Самый верхний
Нижний	0	1	2	3
Средний	3	0	1	2
Верхний	3	2	0	1
Самый верхний	3	2	1	0

7.3.12.2.2 Применение исключения зон

Матрица понижающего микширования для набора исключенных громкоговорителей E рассчитывается следующим образом.

– Для N громкоговорителей расчет начинается с матрицы $N \times N$ понижающего микширования $\mathbf{D}$, причем каждый элемент инициализируется в 0.

– Для каждого входного громкоговорителя i рассматривается каждая группа индексов C громкоговорителей-кандидатов в строке i таблицы группы.

  • Если все громкоговорители в группе находятся в наборе игнорируемых громкоговорителей, то есть $C \subseteq E$, переходят к следующей группе.

  • В противном случае для каждого j в $C \setminus E$ (набор неисключенных громкоговорителей в группе) устанавливается:

$$D_{i,j} = \frac{1}{|C \setminus E|}$$

и осуществляется переход к следующему громкоговорителю.

Если все громкоговорители исключены, $\mathbf{D}$ устанавливается в единичную матрицу.

Затем матрица $\mathbf{D}$ применяется к входному вектору усиления $\mathbf{G}$, что приводит к вектору $\mathbf{G}'$:

$$\mathbf{G}'_j = \sqrt{\sum_i \mathbf{G}_i^2 \, \mathbf{D}_{i,j}}.$$

7.4 Фильтры декорреляции

При рендеринге объектов, у которых параметр *diffuse* больше 0, используется распределенный тракт рендерера объектов с одним фильтром декорреляции на каждый выходной громкоговоритель.

Используются всечастотные КИХ-фильтры со случайной фазой длиной $N = 512$ выборок. Фильтр для данного выхода генерируется следующим образом.

– Генерируется псевдослучайный вектор $\mathbf{r}$ со значениями в диапазоне $[0,1)$ длиной $\frac{N}{2} - 1$ с использованием генератора псевдослучайных числен MT19937, начальным заполнением которого является индекс имен каналов в отсортированном списке всех имен каналов для данной схемы расположения.

– Вектор фазы $\mathbf{p}$ длиной $\frac{N}{2} + 1$ определяется следующим образом:

$$\mathbf{p}_n = \begin{cases} 2\pi\mathbf{r}_{n-1} & 1 \le n \le \frac{N}{2} - 1; \\ 0 & \text{в других случаях.} \end{cases}$$

– Соответствующий вектор частоты $\mathbf{x}$ определяется как $\mathbf{x}_n = \exp(i\mathbf{p}_n)$.

– Для получения фильтра во временной области берется обратное вещественное преобразование Фурье (функция `irfft`) из неотрицательно-частотных компонентов $\mathbf{x}$.

Это реализуется в модуле `core.objectbased.decorrelate.design_decorrelators`.

Задержка, вносимая этими фильтрами, совпадает с задержкой выборки $\frac{(N-1)}{2}$ в прямом тракте.

8 Рендеринг элементов при typeDefinition==DirectSpeakers

Для рендеринга элементов *audioChannelFormat* при *typeDefintion==DirectSpeakers* эти элементы направляются в соответствующий громкоговоритель. Если это невозможно, то в качестве запасного варианта используется точечный панораматор (PSP).

Основной алгоритм является следующим:

1 Для входных данных, определенных с использованием общих определений *audioPackFormat,* описывающих схемы расположения, указанные в Рекомендации МСЭ-R BS.2051-2, применяются правила преобразования в соответствии с пунктом 8.1.

2 Следует определить, относятся ли метаданные к каналу LFE (см. пункт 8.2). Если да, то рассматриваются только выходы LFE, а если нет – только выходы не-LFE.

3 Если какая-либо из меток *speakerLabels* соответствует громкоговорителю (см. пункт 8.3), канал направляется к первому соответствующему громкоговорителю. Если ни одна из этих меток не соответствует громкоговорителю, переходят к следующему шагу.

4 Если указан атрибут `screenEdgeLock`, номинальная позиция смещается к краю экрана по горизонтали и/или вертикали. Минимальные и максимальные границы остаются без изменений (см. пункт 8.4).

5 Если номинальная позиция какого-либо громкоговорителя находится в указанных границах (см. пункт 8.5), канал направляется к ближайшему к указанной номинальной позиции громкоговорителю. Используемые позиции громкоговорителей определяются типом позиции, как указано в пункте 8.5. Если в пределах границ нет громкоговорителей или ближайший к номинальной позиции громкоговоритель не является уникальным, переходят к следующему шагу.

6 Если метаданные относятся к LFE, канал направляется к громкоговорителю `LFE1` (если таковой имеется) или отбрасывается. Если метаданные относятся к каналу, не являющемуся LFE, используется PSP, соответствующий типу системы координат, применяемой для определения его позиции, для рендеринга канала в его номинальной позиции.

В следующих подразделах отдельные шаги описываются более подробно.

Это реализуется в модуле `core.direct_speakers.panner.DirectSpeakersPanner`.

8.1 Правила преобразования

– Если последний элемент *audioPackFormat*, указанный в атрибуте `type_metadata.audioPackFormats`, не является форматом пакета общих определений (то есть он был указан во входных метаданных, а не считан из файла общих определений), то правила преобразования не применяются.

– Следует найти идентификатор последнего элемента *audioPackFormat* в `type_metadata.audioPackFormats` в таблице 15, чтобы определить `input_layout`. Если он не указан, правила преобразования не применяются.

– Следует попытаться применить каждое правило, указанное в таблице 16, по очереди. Если применимо любое правило, то для воспроизведения этого канала используются `gains` (коэффициенты усиления) первого из подходящих правил. Если не подходит ни одно из правил, переходят к следующему шагу. Правило подходит, если выполнены все следующие условия:

  • `rule.speakerLabel` соответствует первой (и единственной) метке *speakerLabel* после применения процедуры нормализации, описанной в пункте 8.3;

  • `input_layout` (как определено выше) указан в списке `rule.input_layouts`, если он приведен;

  • имя схемы расположения выходных громкоговорителей `layout.name` указано в списке `rule.output_layouts`, если он приведен;

  • все имена каналов, перечисленные в списке `rule.gains`, присутствуют в `layout.channel_names`.

8.2 Определение LFE

Канал считается каналом LFE, если элемент частоты в audioChannelFormat имеет значение, указанное в п. 6.3, либо если имеется метка *speakerLabel*, указывающая на канал LFE (LFE1 или LFE2 после применения описанного ниже процесса сопоставления).

ПРИМЕЧАНИЕ. – IAR не использует смещение уровня канала LFE на +10 дБ (см. Рекомендацию МСЭ-R BS.775) относительно уровня основного канала для калибровки воспроизведения, поскольку это делается в устройстве воспроизведения.

8.3 Сопоставление меток громкоговорителей

Сопоставление меток *speakerLabel* подходит только для меток, используемых в Рекомендации МСЭ-R BS.2051-2 (например, `M+030`), и для URN, используемых в файле общих определений ADM из Рекомендации МСЭ-R BS.2094-1 (например, `urn:itu:bs:2051:0:speaker:M+030`). Метки LFE1 и LFE2 определены в Рекомендации МСЭ-R BS.2051-2. При использовании в файле ADM следующих меток *speakerLabel* применяются некоторые замены:

– `LFE → LFE1`;

– `LFEL → LFE1`;

– `LFER → LFE2`.

8.4 Блокировка на краю экрана

В реализации *screenEdgeLock* для *typeDefintion==DirectSpeakers* используется такой же компонент `ScreenEdgeLockHandler`, какой применяется для *typeDefintion==Objects*; он подробно описан в пункте 7.3.4 и используется только для преобразования номинальной позиции; минимальные и максимальные границы остаются без изменений.

Это означает, что если указаны границы, то они интерпретируются как абсолютные, независимо от положения экрана; источник привязывается только к каналу в пределах первоначально указанных границ. Если границы не указаны, то активизируются характеристики точечного панораматора, вызывающиеся привязывание источника к краю экрана, независимо от того, есть ли там громкоговоритель или нет. Не рекомендуется использовать координаты границ вместе с элементом *screenEdgeLock*.

8.5 Соответствие границам

Указанная минимальная или максимальная граница расширяет допустимый диапазон от номинальной позиции. Если минимальная или максимальная граница не указана, устанавливается номинальная координата. Громкоговоритель соответствует границам, если все его координаты лежат в пределах указанных *границ*. Исключение составляют громкоговорители с полярными координатами на полюсах (например, `T+000`), которые соответствуют любому азимутальному диапазону, поскольку их азимут не определен.

Громкоговоритель в номинальной полярной позиции `speaker` соответствует границам, указанным в полярных координатах, если:

$$\begin{pmatrix} \text{IAR}(\texttt{speaker.azimuth}, \texttt{azimuth.min}, \texttt{azimuth.max}, \epsilon) \\ \vee \quad |\texttt{speaker.elevation}| \geq 90° - \epsilon \end{pmatrix};$$
$$\wedge \quad \texttt{elevation.min} - \epsilon \leq \texttt{speaker.elevation} \leq \texttt{elevation.max} + \epsilon;$$
$$\wedge \quad \texttt{distance.min} - \epsilon \leq \texttt{speaker.distance} \leq \texttt{distance.max} + \epsilon,$$

где IAR – функция `inside_angle_range` (см. пункт 6.2), а $\epsilon = 10^{-5}$ – страховочный запас для учета ошибок округления.

Громкоговоритель в декартовой позиции `speaker` , преобразованной в декартовы координаты с использованием пункта 7.3.9, соответствует границам, указанным в декартовых координатах, если:

$$\begin{aligned} &\texttt{X.min} - \epsilon \leq \texttt{speaker.X} \leq \texttt{X.max} + \epsilon; \\ \wedge\ &\texttt{Y.min} - \epsilon \leq \texttt{speaker.Y} \leq \texttt{Y.max} + \epsilon; \\ \wedge\ &\texttt{Z.min} - \epsilon \leq \texttt{speaker.Z} \leq \texttt{Z.max} + \epsilon \end{aligned}$$

имеют значение true (истинно).

9 Рендеринг элементов при typeDefinition==HOA

9.1 Поддерживаемые форматы HOA

9.1.1 Порядок и степень HOA

Сигналы HOA, определенные в Рекомендации МСЭ-R BS.2076-1, могут обрабатываться до 50-го порядка (подробнее см. ниже). В ADM каналы HOA сигнализируются индивидуально по их порядку и степени через соответствующие подэлементы типа HOA. Таким образом могут быть представлены полностью трехмерные сцены HOA (содержащие каждый порядок l и каждую степень m до заданного порядка L), двухмерные сцены HOA (содержащие каждый компонент HOA, такой, что $|m| = l$ до заданного порядка L), а также сцены HOA со смешанным порядком.

Однако в том случае, когда два сигнала HOA имеют одинаковый порядок *и* одинаковую степень, возникает исключение, и рендеринг сигналов не производится.

9.1.2 Нормализация

Тип нормализации сигнала HOA указывается с помощью подэлемента HOA *normalization*. Описываемый рендерер поддерживает все три возможных типа нормализации (N3D, SN3D и FuMa). В ADM нормализация HOA указывается для каждого сигнала HOA отдельно, поэтому теоретически можно определить сцены HOA, в которых для разных сигналов используется нормализация разных типов. Однако описываемый рендерер это не поддерживает: для всех каналов HOA в *audioBlockFormat* должна использоваться нормализация одного и того же типа. Наконец, следует отметить, что нормализация типа FuMa поддерживается только до третьего порядка.

9.2 Неподдерживаемые подэлементы

В настоящее время при рендеринге не интерпретируются следующие три подэлемента типа HOA:

– *nfcRefDist*, который указывает эталонное расстояние для громкоговорителей. Эффект компенсации ближнего поля (NFC), компенсирующий несоответствие между эталонным расстоянием для громкоговорителей и расстоянием, на котором громкоговорители расположены в схеме воспроизведения, в данном рендерере не реализован. Реализация этого эффекта при рендеринге HOA значительно повышает сложность вычислений рендерера при относительно незначительном влиянии на восприятие звукового контента слушателем;

– *screenRef*, который указывает на то, связан ли компонент HOA с экраном. Ожидаемое использование этого подэлемента в контексте HOA неоднозначно, поэтому он не учитывается при рендеринге;

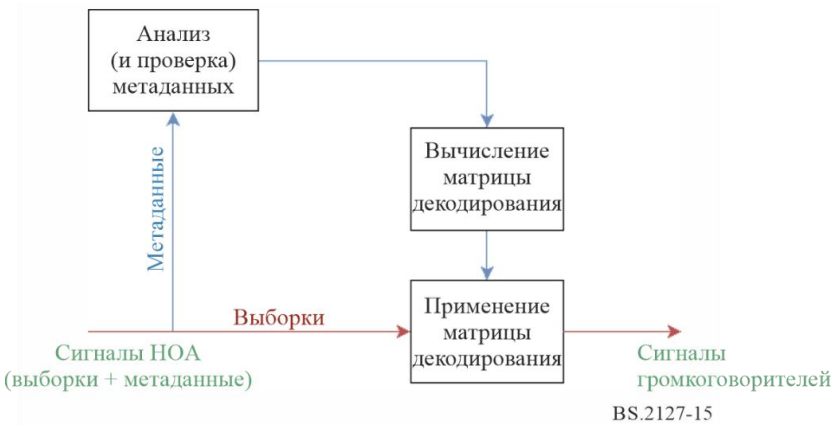
– *equation*, который предназначен для использования взамен подэлементов *order* и *degree*. Текущий стандарт ADM не содержит точных правил относительно формата, используемого для определения математических формул. Следовательно, этот подэлемент не может надежно поддерживаться.

Следует отметить, что, как и в случае подэлемента *normalization*, все каналы HOA в *audioBlockFormat* должны совместно использовать для рендеринга одни и те же значения *nfcRefDist* и *screenRef*.

9.3 Рендеринг сигналов HOA через громкоговорители

РИСУНОК 15

Блок-схема рендеринга HOA



Процесс рендеринга сигналов HOA через громкоговорители представлен на рисунке 15. Сначала метаданные ADM анализируются, чтобы определить формат объекта HOA и проверить, можно ли однозначно обработать сигналы. В частности, как указано выше, во всех каналах HOA *audioBlockFormat* должны использоваться одинаковые значения подэлементов *normalization*, *nfcRefDist* и *screenRef*. Затем вычисляется матрица декодирования для громкоговорителей и применяется к сигналам HOA. Это выражается следующим уравнением:

$$\mathbf{S}_{\mathrm{spk}} = \mathbf{D}\,\mathbf{S}_{\mathrm{HOA}},$$

где:

$\mathbf{S}_{\mathrm{spk}}$: матрица сигналов для громкоговорителей с размерностью $N_{\mathrm{spk}} \times N_{\mathrm{samp}}$;

$\mathbf{S}_{\mathrm{HOA}}$: матрица сигналов HOA с размерностью $N_{\mathrm{HOA}} \times N_{\mathrm{samp}}$;

$\mathbf{D}$: матрица действительных значений с размерностью $N_{\mathrm{spk}} \times N_{\mathrm{HOA}}$, называемая матрицей декодирования HOA.

N_{HOA}, N_{spk} и N_{samp} обозначают соответственно число сигналов HOA, сигналов громкоговорителей и выборок.

В данном подразделе описывается расчет матрицы декодирования в порядке следования каналов ACN, однако при этом используется распределение каналов, указанное в параметрах *order* и *degree* в *audioBlockFormat*.

Матрица декодирования применяется посредством использования структуры канала обработки блоков, описанной в пункте 6.4. В частности, для каждого входящего объекта `HOATypeMetadata` генерируется один блок обработки `FixedMatrix`, который применяет матрицу декодирования между моментами времени, определенными в пункте 6.5.

9.3.1 Расчет матрицы декодирования HOA

В рендерере реализуется метод декодирования HOA AllRAD [1]. Данный метод обеспечивает надежное декодирование HOA в нерегулярных схемах размещения громкоговорителей, таких как схемы расположения, описанные в Рекомендации МСЭ-R BS.2051-2. Расчет матрицы декодирования выполняется в модуле `core.scenebased.design.HOADecoderDesign`.

Концептуально метод декодирования AllRAD эквивалентен следующему:

1) декодирование сигналов HOA для сетки виртуальных громкоговорителей, равномерно распределенных по сфере; и
2) панорамирование сигналов виртуальных громкоговорителей через реальные громкоговорители.

Математически это можно выразить так:

$$\mathbf{D}' = \nu\, \mathbf{G}\, \mathbf{D}_{\mathrm{virt}};$$

$$\mathbf{D} = \mathbf{D}' \mathrm{diag}(\mathbf{n}^{-1}),$$

где $\mathbf{D}'$ – матрица декодирования HOA для нормализации *N3D*, $\mathbf{G}$ – матрица усиления для панорамирования, $\mathbf{D}_{\mathrm{virt}}$ – матрица декодирования для виртуальных громкоговорителей, а ν – коэффициент нормализации энергии. $\mathbf{D}$ – окончательная матрица декодирования после применения к $\mathbf{D}'$ вектора нормализации HOA $\mathbf{n}$ для достижения требуемой нормализации.

9.3.1.1 Позиции виртуальных громкоговорителей

Для упрощения расчета матрицы декодирования угловые положения виртуальных громкоговорителей должны быть как можно более равномерно распределены по сфере. Кроме того, как показывает опыт, виртуальных громкоговорителей должно быть примерно в два раза больше, чем сигналов HOA.

В описываемом рендерере позиции виртуальных громкоговорителей составляют *сферический T-дизайн* из 5200 точек, что делает его хорошо подходящим для декодирования сигналов HOA до 50-го порядка.

9.3.1.2 Расчет матрицы декодирования виртуальных громкоговорителей

При расчете матрицы декодирования для виртуальных громкоговорителей сначала вычисляется матрица коэффициентов HOA для виртуальных громкоговорителей $\mathbf{Y}_{\mathrm{virt}}$. Эта матрица определяется следующим образом:

$$\mathbf{Y}_{\mathrm{virt}} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{N_{\mathrm{virt}}}];$$

$$\mathbf{y}_n = [Y_0^0(\theta_n, \varphi_n), Y_1^{-1}(\theta_n, \varphi_n), \ldots]^{\mathrm{T}},$$

где (θ_n, φ_n) – угол места и азимут *n*-го виртуального громкоговорителя (с использованием системы координат HOA и системы обозначений, определенной в Рекомендации МСЭ-R BS.2076-1), а Y_l^m – действительная сферическая гармоническая функция порядка l и степени m с *N3D*-нормализацией. Следует отметить, что значение каждого члена $Y_l^m(\theta, \varphi)$ зависит от подэлементов *order* и *degree* каждого канала HOA.

Затем вычисляется матрица декодирования HOA для виртуальных громкоговорителей как транспонирование $\mathbf{Y}_{\mathrm{virt}}$:

$$\mathbf{D}_{\mathrm{virt}} = N_{\mathrm{samp}}^{-1}\mathbf{Y}_{\mathrm{virt}}^{\mathrm{T}}.$$

Для выбора позиций виртуальных громкоговорителей и нормализации *N3D* это эквивалентно принятию псевдообратного значения $\mathbf{Y}_{\mathrm{virt}}$.

9.3.1.3 Расчет матрицы коэффициентов усиления панорамирования

Панорамирование VBAP обычно используется для расчета матрицы коэффициентов усиления панорамирования в методе декодирования HOA AllRAD. В описываемой реализации рендерера в качестве метода вычисления коэффициентов усиления панорамирования просто используется метод, предназначенный для объектов панорамирования точечных источников (`core.point_source`).

9.3.1.4 Нормализация энергии

Матрица декодирования HOA нормирована таким образом, что в том случае, когда сцена HOA состоит из одного точечного источника, общая мощность сигналов громкоговорителей равна мощности исходного сигнала, усредненной по всем возможным местоположениям источников на сфере.

Математически коэффициент нормализации ν рассчитывается следующим образом:

$$\nu = \frac{\sqrt{N_{\mathrm{virt}}}}{\|\mathbf{G}\,\mathbf{D}_{\mathrm{virt}}\,\mathbf{Y}_{\mathrm{virt}}\|_{\mathrm{F}}},$$

где $\|\cdot\|_{\mathrm{F}}$ – норма Фробениуса.

9.3.1.5 Нормализация HOA

Матрица декодирования делится на вектор $\mathbf{n}$, чтобы преобразовать сигнал для нормализации *N3D*, для чего предназначена матрица $\mathbf{D}'$. $\mathbf{n}$ определяется для заданного параметра *нормализации* `norm` следующим образом:

$$\mathbf{n}_n^m = \frac{{N_{\mathrm{norm}}}_n^{|m|}}{{N_{\mathrm{N3D}}}_n^{|m|}};$$

$$\mathbf{n} = [\mathbf{n}_0^0, \mathbf{n}_1^{-1}, \ldots].$$

10 Преобразование метаданных

В данном разделе описывается метод преобразования параметров из полярных в декартовы координаты и наоборот в элементах *audioBlockFormat* при *typeDefinition==Objects*. Преобразование метаданных по своей природе не может быть точным; его результаты не будут точно совпадать с исходными. Следовательно, результаты преобразования необходимо контролировать. Следует отметить, что преобразование расширения необратимо, поэтому следует избегать его преобразования туда и обратно между полярными и декартовыми координатами.

Интерфейс для функциональной возможности преобразования является следующим:

```
AudioBlockFormat to_cartesian(AudioBlockFormat input);
AudioBlockFormat to_polar(AudioBlockFormat input);
```

Когда вызывается функция `to_cartesian` с атрибутом `AudioBlockFormat input`, где установлен флаг `input.cartesian`, значение `input` возвращается "как есть". И наоборот, когда вызывается функция `to_polar` с атрибутом `AudioBlockFormat input`, где флаг `input.cartesian` не установлен, значение `input` также возвращается "как есть".

Иначе в обоих случаях `input.cartesian` инвертируется, и в значение `input` перед его возвращением вносятся следующие изменения:

– `input.position` преобразуется согласно пункту 10.1;

– `input.width`, `input.height` и `input.depth` преобразуются согласно пункту 10.2;

– `input.objectDivergence` преобразуется согласно пункту 10.3.

Преобразование реализуется в модуле `core.objectbased.conversion`.

10.1 Преобразование *position*

Позиции преобразуются таким образом, чтобы положение громкоговорителя в схеме расположения `4+5+0` из полярных координат преобразовывалось в декартовы координаты этого громкоговорителя, используемые в декартовом точечном панораматоре, как указано в таблице 8.

Следует отметить, что независимо от схемы расположения каналов рендерера используется одно и то же преобразование, основанное на конфигурации каналов 4+5+0. Это сделано для того, чтобы результаты преобразования всегда были согласованы, даже в тех случаях, когда во время преобразования используемая схема воспроизведения рендерера неизвестна. Схема расположения 4+5+0 выбрана в первую очередь для обеспечения правильного преобразования контента, созданного с использованием схемы 0+5+0.

В данном подразделе даны общие определения, используемые для преобразования в обоих направлениях; сами функции преобразования описаны в пунктах 10.1.1 и 10.1.2.

Функции `map_linear_to_az` и `map_az_to_linear` определяют обратимое преобразование позиций источников между азимутами (φ) и линейными координатами (x) для пары громкоговорителей с азимутами φ_l и $azimuth_r$ с учетом кривых панорамирования точечных панораматоров, используемых для полярных и декартовых координат.

Например, полярной позиции φ_o между 0° и −30° соответствует позиция x, определяемая как:

$$x = \texttt{map\_az\_to\_linear}(0, -30, \varphi_o).$$

Линейно-азимутальное преобразование определяется следующим образом:

$$\texttt{map\_linear\_to\_az}(\varphi_l, \varphi_r, x) = \varphi_{\texttt{mid}} + \varphi_{\texttt{rel}},$$

где:

$$\begin{aligned}
\varphi_{\texttt{mid}} &= \frac{\varphi_l + \varphi_r}{2};\\
\varphi_{\texttt{range}} &= \varphi_r - \varphi_{\texttt{mid}};\\
g'_l &= \cos\frac{x\pi}{2};\\
g'_r &= \sin\frac{x\pi}{2};\\
g_r &= \frac{g'_r}{g'_l + g'_r};\\
\varphi_{\texttt{rel}} &= \frac{180}{\pi}\arctan\left(2\left(g_r - \frac{1}{2}\right)\tan\left(\frac{\pi}{180}\varphi_{\texttt{range}}\right)\right).
\end{aligned}$$

Обратная функция определяется следующим образом:

$$\texttt{map\_az\_to\_linear}(\varphi_l, \varphi_r, \varphi) = \frac{2}{\pi}\text{atan2}(g_r, 1 - g_r),$$

где:

$$\begin{aligned}\varphi_{\mathtt{mid}} &= \frac{\varphi_l+\varphi_r}{2};\\ \varphi_{\mathtt{range}} &= \varphi_r-\varphi_{\mathtt{mid}};\\ \varphi_{\mathtt{rel}} &= \varphi-\varphi_{\mathtt{mid}};\\ g_r &= \frac{1}{2}+\frac{\tan\left(\frac{\pi}{180}\varphi_{\mathtt{rel}}\right)}{2\tan\left(\frac{\pi}{180}\varphi_{\mathtt{range}}\right)}.\end{aligned}$$

Это преобразование проводится между позициями громкоговорителей среднего уровня по следующим правилам, задающим левый и правый азимуты, а также левые и правые позиции x и y для данного входного азимута:

$$find_{sector}(\varphi)\quad\begin{cases}\{30{,}0,\{-1{,}1\},\{0{,}1\}\} & IAR(\varphi,0{,}30);\\ \{0,-30,\{0{,}1\},\{1{,}1\}\} & IAR(\varphi,-30{,}0);\\ \{-30,-110,\{1{,}1\},\{1,-1\}\} & IAR(\varphi,-110,-30);\\ \{-110{,}110,\{1,-1\},\{-1,-1\}\} & IAR(\varphi,110,-110);\\ \{110{,}30,\{-1,-1\},\{-1{,}1\}\} & IAR(\varphi,30{,}110);\end{cases}$$

$$find_{\square}(\varphi)\quad\begin{cases}\{30{,}0,\{-1{,}1\},\{0{,}1\}\} & IAR(\varphi,0{,}45);\\ \{0,-30,\{0{,}1\},\{1{,}1\}\} & IAR(\varphi,-45{,}0);\\ \{-30,-110,\{1{,}1\},\{1,-1\}\} & IAR(\varphi,-135,-45);\\ \{-110{,}110,\{1,-1\},\{-1,-1\}\} & IAR(\varphi,135,-135);\\ \{110{,}30,\{-1,-1\},\{-1{,}1\}\} & IAR(\varphi,45{,}135),\end{cases}$$

где IAR – функция `inside_angle_range`, описанная в пункте 6.2.

Следующие параметры являются общими для преобразования в обоих направлениях:

$$\begin{aligned}\theta_{\mathtt{top}} &= 30;\\ \theta'_{\mathtt{top}} &= 45;\\ \epsilon &= 1\times10^{-10}.\end{aligned}$$

10.1.1 Преобразование полярных координат в декартовы

Для преобразования позиции в полярных координатах с азимутом φ, углом места θ и расстоянием d в декартовы координаты используется функция:

$$\mathtt{point\_polar\_to\_cart}(\varphi,\theta,d)=x,y,z,$$

где если $|\theta|>\theta_{\mathtt{top}}$, то:

$$\begin{aligned}\theta' &= \theta'_{\mathtt{top}}+\left(90-\theta'_{\mathtt{top}}\right)\frac{|\theta|-\theta_{\mathtt{top}}}{90-\theta_{\mathtt{top}}};\\ z &= d\mathrm{sgn}(\theta);\\ r_{xy} &= d\tan\left(\frac{\pi}{180}(90-\theta')\right),\end{aligned}$$

в противном случае:

$$\begin{aligned}\theta' &= \theta'_{\mathtt{top}}\frac{\theta}{\theta_{\mathtt{top}}}\\ z &= d\tan\left(\frac{\pi}{180}\theta'\right);\\ r_{xy} &= d\end{aligned}$$

и наконец:

$$\begin{aligned}\{\varphi_l, \varphi_r, \{x_l, y_l\}, \{x_r, y_r\}\} &= \texttt{find\_sector}(\varphi);\\ \varphi' &= \texttt{relative\_angle}(\varphi_r, \varphi);\\ \varphi'_l &= \texttt{relative\_angle}(\varphi_r, \varphi_l);\\ p &= \texttt{map\_az\_to\_linear}(\varphi'_l, \varphi_r, \varphi');\\ x &= r_{xy}\big(x_l + p(x_r - x_l)\big);\\ y &= r_{xy}\big(y_l + p(y_r - y_l)\big).\end{aligned}$$

Функция `relative_angle` описана в пункте 6.7.

10.1.2 Преобразование декартовых координат в полярные

Для преобразования декартовых координат позиции x, y и z в полярные используется функция:

$$\texttt{point\_cart\_to\_polar}(x, y, z) = \varphi, \theta, d,$$

где если $|x| < \epsilon$ и $|y| < \epsilon$, то:

$$\{\varphi, \theta, d\} = \begin{cases}\{0{,}0{,}0\} & |z| < \epsilon;\\ \{0{,}90\,\mathrm{sgn}(z), |z|\} & \text{в других случаях,}\end{cases}$$

в противном случае продолжаем:

$$\begin{aligned}\varphi' &= -\tfrac{180}{\pi}\,\mathrm{atan2}(x, y);\\ \{\varphi_l, \varphi_r, \{x_l, y_l\}, \{x_r, y_r\}\} &= \texttt{find\_cart\_sector}(\varphi');\\ [g_l \quad g_r] &= [x \quad y] \cdot \begin{bmatrix} x_l & y_l \\ x_r & y_r \end{bmatrix}^{-1};\\ r_{xy} &= g_l + g_r;\\ \varphi'_l &= \texttt{relative\_angle}(\varphi_r, \varphi_l);\\ \varphi_{\texttt{rel}} &= \texttt{map\_linear\_to\_az}\left(\varphi'_l, \varphi_r, \tfrac{g_r}{r_{xy}}\right);\\ \varphi &= \texttt{relative\_angle}(-180, \varphi_{\texttt{rel}});\\ \theta' &= \tfrac{180}{\pi}\arctan\tfrac{z}{r_{xy}}.\end{aligned}$$

Если $|\theta'| > \theta'_{\texttt{top}}$, то:

$$\begin{aligned}|\theta| &= \theta_{\texttt{top}} + \big(90 - \theta_{\texttt{top}}\big)\tfrac{|\theta'| - \theta'_{\texttt{top}}}{90 - \theta'_{\texttt{top}}};\\ \theta &= |\theta|\,\mathrm{sgn}\theta';\\ d &= |z|,\end{aligned}$$

в противном случае:

$$\begin{aligned}\theta &= \theta'\tfrac{\theta_{\texttt{top}}}{\theta'_{\texttt{top}}};\\ d &= r_{xy}.\end{aligned}$$

Функция `local_coordinate_system` описана в пункте 6.8.

10.2 Преобразование расширения

Преобразование параметров расширения осуществляется в два этапа:

– функции $whd2xyz$ и $xyz2whd$ преобразуют параметры расширения из декартовых координат в полярные и наоборот в предположении, что позиция источника находится непосредственно перед слушателем с радиусом 1;

– функции point_polar_to_cart и point_cart_to_polar управляют преобразованием позиции и расширения. Позиции преобразуются с использованием методов, описанных в пункте 10.1. Для преобразования расширения используются функции $whd2xyz$ и $xyz2whd$, причем в декартовых координатах поворачивается для совмещения позиций.

Следует отметить, что преобразование расширения в общем случае необратимо.

10.2.1 Преобразование полярных координат в декартовы

Функция dist_polar_to_cart принимает позицию источника в полярных координатах в виде азимута, угла места и расстояния, а также ширину, высоту и глубину в полярных координатах и возвращает декартовы координаты x, y и z и декартовы размеры x, y и z:

$$\texttt{extent\_polar\_to\_cart}(\varphi, \theta, d, \texttt{width}, \texttt{height}, \texttt{depth}) = \{x, y, z, s_x, s_y, s_z\},$$

где:

$$\begin{aligned} \{x, y, z\} &= \texttt{point\_polar\_to\_cart}(\varphi, \theta, d) \\ \{s_{x,f}, s_{y,f}, s_{z,f}\} &= \texttt{whd2xyz}(\texttt{width}, \texttt{height}, \texttt{depth}) \\ [\mathbf{M}_x \quad \mathbf{M}_y \quad \mathbf{M}_z] &= \text{diag}([s_{x,f}, s_{y,f}, s_{z,f}]) \cdot \texttt{local\_coordinate\_system}(\varphi, \theta); \\ s_x &= \| \mathbf{M}_x \|_2 \\ s_y &= \| \mathbf{M}_y \|_2 \\ s_z &= \| \mathbf{M}_z \|_2 \end{aligned}$$

и

$$whd2xyz(width, height, depth) = \{s_{x,w}, \max(s_{y,w}, s_{y,h}, s_{y,d}), s_{z,h}\},$$

где:

$$\begin{aligned} s_{x,w} &= \begin{cases} \sin\frac{\pi}{180}\frac{\texttt{width}}{2} & \texttt{width} < 180; \\ 1 & \text{в других случаях;} \end{cases} \\ s_{y,w} &= \frac{1-\cos\frac{\pi\ \texttt{width}}{180\ \ 2}}{2}; \\ s_{z,h} &= \begin{cases} \sin\frac{\pi}{180}\frac{\texttt{height}}{2} & \texttt{height} < 180; \\ 1 & \text{в других случаях;} \end{cases} \\ s_{y,h} &= \frac{1-\cos\frac{\pi\ \texttt{height}}{180\ \ 2}}{2}; \\ s_{y,d} &= \texttt{depth}. \end{aligned}$$

10.2.2 Преобразование декартовых координат в полярные

Функция dist_cart_to_polar принимает исходную позицию в декартовых координатах в виде координат x, y и z, расширение в декартовых координатах в виде размеров x, y и z и возвращает позицию и расширение в полярных координатах в виде азимута, угла места, расстояния и ширины, высоты и глубины:

$$\texttt{extent\_cart\_to\_polar}(x, y, z, s_x, s_y, s_z) = \{\varphi, \theta, d, \texttt{width}, \texttt{height}, \texttt{depth}\},$$

где:

$$\begin{aligned} \{\varphi, \theta, d\} &= \texttt{point\_cart\_to\_polar}(x, y, z); \\ [\mathbf{M}_x \quad \mathbf{M}_y \quad \mathbf{M}_z] &= \text{diag}([s_x, s_y, s_z]) \cdot \texttt{local\_coordinate\_system}(\varphi, \theta)^T; \\ s_{x,f} &= \| \mathbf{M}_x \|_2; \\ s_{y,f} &= \| \mathbf{M}_y \|_2; \\ s_{z,f} &= \| \mathbf{M}_z \|_2; \\ \{\texttt{width}, \texttt{height}, \texttt{depth}\} &= \texttt{xyz2whd}(s_{x,f}, s_{y,f}, s_{z,f}) \end{aligned}$$

и

$$xyz2whd(s_x, s_y, s_z) = \{w, h, d\},$$

где:

$$\begin{aligned} w_{sx} &= 2\frac{180}{\pi}\arcsin s_x; \\ w_{sy} &= 2\frac{180}{\pi}\arccos(1 - 2s_y); \\ w &= w_{sx} + s_x\max(w_{sy} - w_{sx}, 0); \\ h_{sz} &= 2\frac{180}{\pi}\arcsin s_z; \\ h_{sy} &= 2\frac{180}{\pi}\arccos(1 - 2s_y); \\ h &= h_{sz} + s_z\max(h_{sy} - h_{sz}, 0); \\ \{s_{x,\mathrm{eq}}, s_{y,\mathrm{eq}}, s_{z,\mathrm{eq}}\} &= \mathrm{whd2xyz}(w, h, 0); \\ d &= \max(0, s_y - s_{y,\mathrm{eq}},). \end{aligned}$$

10.3 Преобразование objectDivergence

Значения `azimuthRange` и `positionRange` преобразуются в соответствии со следующим отношением:

$$positionRange = tan\frac{270 \times azimuthRange}{\pi}.$$

11 Структуры данных и таблицы

11.1 Структуры внутренних метаданных

11.1.1 Общие структуры

```
struct Position { };

struct PolarPosition : Position {
  float azimuth, elevation, distance = 1;
};

struct CartesianPosition : Position {
  float x, y, z;
};

struct Screen { };

struct PolarScreen : Screen {
  float aspectRatio;
  PolarPosition centrePosition;
  float widthAzimuth;
};

struct CartesianScreen : Screen {
  float aspectRatio;
  CartesianPosition centrePosition;
  float widthX;
};

struct Frequency {
  optional<float> lowPass;
  optional<float> highPass;
};

struct ExtraData {
  Fraction object_start;
  Fraction object_duration;
```

```
  Screen reference_screen;
  Frequency channel_frequency;
};
```

11.1.2 Входные метаданные

```
struct ChannelLock {
  optional<float> maxDistance;
};

struct ObjectDivergence {
  float value;
  optional<float> azimuthRange;
  optional<float> positionRange;
};

struct JumpPosition {
  bool flag;
  optional<float> interpolationLength;
};

struct ExclusionZone { };

struct CartesianZone : ExclusionZone {
  float minX;
  float minY;
  float minZ;
  float maxX;
  float maxY;
  float maxZ;
};

struct PolarZone : ExclusionZone {
  float minElevation;
  float maxElevation;
  float minAzimuth;
  float maxAzimuth;
};

struct ScreenEdgeLock {
    enum Horizontal { LEFT; RIGHT; };
    enum Vertical { BOTTOM; TOP; };

    optional<Horizontal> horizontal;
    optional<Vertical> vertical;
};

struct ObjectPosition { };

class PolarObjectPosition : ObjectPosition {
    float azimuth, elevation, distance;
    ScreenEdgeLock screenEdgeLock;
};

class CartesianObjectPosition | ObjectPosition {
    float X, Y, Z;
    ScreenEdgeLock screenEdgeLock;
};

struct AudioBlockFormatObjects {
  ObjectPosition position;
  bool cartesian;
  float width, height, depth;
  float diffuse;
  optional<ChannelLock> channelLock;
  optional<ObjectDivergence> objectDivergence;
  optional<JumpPosition> jumpPosition;
  bool screenRef;
```

```
  int importance;
  vector<ExclusionZone> zoneExclusion;
};

struct ObjectTypeMetadata {
  AudioBlockFormatObjects block_format;
  ExtraData extra_data;
};
```

11.1.3 Данные воспроизведения среды

```
struct Channel {
  string name;
  /// Реальное положение громкоговорителя
  PolarPosition polar_position;
  /// Номинальное положение громкоговорителя, как в Рек.bs.2051-2
  PolarPosition polar_nominal_position;
  bool is_lfe;
};

struct Layout {
  /// имя схемы расположения в формате МСЭ, например "9+10+3"
  string name;
  vector<Channel> channels;
  Screen screen;
};
```

11.2 Аллоцентрические позиции громкоговорителей

Эти данные доступны в машиночитаемой форме в `iar/core/data/allo_positions.yaml`, но включены сюда для справок.

ТАБЛИЦА 5

Аллоцентрические позиции громкоговорителей для схемы расположения 0+2+0

Канал	X	Y	Z
M+030	−1	1	0
M–030	1	1	0

ТАБЛИЦА 6

Аллоцентрические позиции громкоговорителей для схемы расположения 0+5+0

Канал	X	Y	Z
M+030	−1	1	0
M–030	1	1	0
M+000	0	1	0
M+110	−1	−1	0
M–110	1	−1	0
LFE1	–1	1	–1

ТАБЛИЦА 7

Аллоцентрические позиции громкоговорителей для схемы расположения 2+5+0

Канал	X	Y	Z
M+030	−1	1	0
M–030	1	1	0
M+000	0	1	0
M+110	−1	−1	0
M–110	1	−1	0
U+030	−1	1	1
U–030	1	1	1
LFE1	–1	1	–1

ТАБЛИЦА 8

Аллоцентрические позиции громкоговорителей для схемы расположения 4+5+0

Канал	X	Y	Z
M+030	−1	1	0
M–030	1	1	0
M+000	0	1	0
M+110	−1	−1	0
M–110	1	−1	0
U+030	−1	1	1
U–030	1	1	1
U+110	−1	−1	1
U–110	1	−1	1
LFE1	–1	1	–1

ТАБЛИЦА 9

Аллоцентрические позиции громкоговорителей для схемы расположения 4+5+1

Канал	X	Y	Z
M+030	−1	1	0
M–030	1	1	0
M+000	0	1	0
M+110	−1	−1	0
M–110	1	−1	0
U+030	−1	1	1
U–030	1	1	1
U+110	−1	−1	1
U–110	1	−1	1
B+000	0	1	−1
LFE1	–1	1	–1

ТАБЛИЦА 10

Аллоцентрические позиции громкоговорителей для схемы расположения 3+7+0

Канал	X	Y	Z
M+000	0	1	0
M+030	−1	1	0
M–030	1	1	0
U+045	−1	1	1
U–045	1	1	1
M+090	−1	0	0
M–090	1	0	0
M+135	−1	−1	0
M–135	1	−1	0
UH+180	0	−1	1
LFE1	–1	1	–1
LFE2	1	1	–1

ТАБЛИЦА 11

Аллоцентрические позиции громкоговорителей для схемы расположения 4+9+0

Канал	X	Y	Z
M+030	−1	1	0
M–030	1	1	0
M+000	0	1	0
M+090	−1	0	0
M–090	1	0	0
M+135	−1	−1	0
M–135	1	−1	0
U+045	−1	1	1
U–045	1	1	1
U+135	−1	−1	1
U–135	1	−1	1
LFE1	–1	1	–1

ТАБЛИЦА 12

Аллоцентрические позиции громкоговорителей для схемы расположения 9+10+3

Канал	X	Y	Z
M+060	−1	0,414214	0
M–060	1	0,414214	0
M+000	0	1	0
M+135	−1	−1	0
M–135	1	−1	0
M+030	−1	1	0
M–030	1	1	0
M+180	0	−1	0
M+090	−1	0	0
M–090	1	0	0
U+045	−1	1	1
U–045	1	1	1
U+000	0	1	1
T+000	0	0	1
U+135	−1	−1	1
U–135	1	−1	1
U+090	−1	0	1
U–090	1	0	1
U+180	0	−1	1
B+000	0	1	−1
B+045	−1	1	−1
B–045	1	1	−1
LFE1	–1	1	–1
LFE2	1	1	–1

ТАБЛИЦА 13

Аллоцентрические позиции громкоговорителей для схемы расположения 0+7+0

Канал	X	Y	Z
M+030	−1	1	0
M–030	1	1	0
M+000	0	1	0
M+090	−1	0	0
M–090	1	0	0
M+135	−1	−1	0
M–135	1	−1	0
LFE1	–1	1	–1

ТАБЛИЦА 14

Аллоцентрические позиции громкоговорителей для схемы расположения 4+7+0

Канал	X	Y	Z
M+030	−1	1	0
M–030	1	1	0
M+000	0	1	0
M+090	−1	0	0
M–090	1	0	0
M+135	−1	−1	0
M–135	1	−1	0
U+045	−1	1	1
U–045	1	1	1
U+135	−1	−1	1
U–135	1	−1	1
LFE1	–1	1	–1

11.3 Данные преобразования DirectSpeakers

Эти данные доступные в машиночитаемой форме `core.direct_speakers.panner.itu_packs` и `core.direct_speakers.panner.rules`, но включены сюда для справок.

ТАБЛИЦА 15

Преобразование общих определений *audioPackFormatID* в имена схем расположения (см. пункт 8.1)

audioPackFormatID	`input_layout`
AP_00010001	0+1+0
AP_00010002	0+2+0
AP_00010003	0+5+0
AP_00010004	2+5+0
AP_00010005	4+5+0
AP_00010007	3+7+0
AP_00010008	4+9+0
AP_00010009	9+10+3
AP_0001000c	0+5+0
AP_0001000f	0+7+0
AP_00010010	4+5+1
AP_00010017	4+7+0

ТАБЛИЦА 16

Правила преобразования DirectSpeakers (см. пункт 8.1)

Входное значение *speakerLabel*	**Выходные коэффициенты усиления воспроизведения**	`input_layouts`	`output_layouts`
M+000	M+000 = 1		
M+000	M+030 = M–030 = $\sqrt{\frac{1}{2}}$		
M+060	M+060 = 1		
M-060	M–060 = 1		
M+060	M+110 = $\sqrt{\frac{1}{3}}$, M+030 = $\sqrt{\frac{2}{3}}$		
M–060	M–110 = $\sqrt{\frac{1}{3}}$, M–030 = $\sqrt{\frac{2}{3}}$		
M+060	M+030 = M+090 = $\sqrt{\frac{1}{2}}$		
M–060	M–030 = M–090 = $\sqrt{\frac{1}{2}}$		
M+060	M+030 = 1		
M–060	M–030 = 1		
M+090	M+090 = 1		
M–090	M–090 = 1		
M+090	M+030 = $\sqrt{\frac{1}{3}}$, M+110 = $\sqrt{\frac{2}{3}}$	9+10+3	
M–090	M–030 = $\sqrt{\frac{1}{3}}$, M–110 = $\sqrt{\frac{2}{3}}$	9+10+3	
M+090	M+030 = M+110 = $\sqrt{\frac{1}{2}}$		
M–090	M–030 = M–110 = $\sqrt{\frac{1}{2}}$		
M+090	M+030 = $\sqrt{\frac{1}{2}}$		
M–090	M–030 = $\sqrt{\frac{1}{2}}$		
M+110	M+110 = 1		
M–110	M–110 = 1		
M+110	M+135 = 1		
M–110	M–135 = 1		
M+110	M+030 = $\sqrt{\frac{1}{2}}$		
M–110	M–030 = $\sqrt{\frac{1}{2}}$		
M+135	M+135 = 1		
M–135	M–135 = 1		
M+135	M+110 = 1		
M–135	M–110 = 1		

ТАБЛИЦА 16 (*продолжение*)

Входное значение *speakerLabel*	**Выходные коэффициенты усиления воспроизведения**	`input_layouts`	`output_layouts`
M+135	M+030 = $\sqrt{\frac{1}{2}}$		
M–135	M–030 = $\sqrt{\frac{1}{2}}$		
M+180	M+180 = 1		
M+180	M+135 = M–135 = $\sqrt{\frac{1}{2}}$		
M+180	M+110 = M–110 = $\sqrt{\frac{1}{2}}$		
M+180	M+030 = M–030 = $\sqrt{\frac{1}{4}}$		
U+000	U+000 = 1		
U+000	U+030 = U–030 = $\sqrt{\frac{1}{2}}$		
U+000	U+045 = U–045 = $\sqrt{\frac{1}{2}}$		
U+000	M+000 = 1		
U+000	M+030 = M–030 = $\sqrt{\frac{1}{2}}$		
U+030	U+030 = 1		
U–030	U–030 = 1		
U+030	U+045 = 1		
U–030	U–045 = 1		
U+030	M+030 = 1		
U–030	M–030 = 1		
U+045	U+045 = 1		
U–045	U–045 = 1		
U+045	U+030 = 1		
U–045	U–030 = 1		
U+045	M+030 = 1		
U–045	M–030 = 1		
U+090	U+090 = 1		
U–090	U–090 = 1		
U+090	UH+180 = $\sqrt{\frac{1}{3}}$, U+045 = $\sqrt{\frac{2}{3}}$	9+10+3	
U–090	UH+180 = $\sqrt{\frac{1}{3}}$, U–045 = $\sqrt{\frac{2}{3}}$	9+10+3	
U+090	U+030 = U+110 = $\sqrt{\frac{1}{2}}$		
U–090	U–030 = U–110 = $\sqrt{\frac{1}{2}}$		
U+090	U+045 = U+135 = $\sqrt{\frac{1}{2}}$		

ТАБЛИЦА 16 (*продолжение*)

Входное значение *speakerLabel*	**Выходные коэффициенты усиления воспроизведения**	`input_layouts`	`output_layouts`
U–090	U–045 = U–135 = $\sqrt{\frac{1}{2}}$		
U+090	M+090 = 1		
U–090	M–090 = 1		
U+090	U+030 = M+110 = $\sqrt{\frac{1}{2}}$		
U–090	U–030 = M–110 = $\sqrt{\frac{1}{2}}$		
U+090	M+030 = M+110 = $\sqrt{\frac{1}{2}}$		
U–090	M–030 = M–110 = $\sqrt{\frac{1}{2}}$		
U+090	M+030 = $\sqrt{\frac{1}{2}}$		
U–090	M–030 = $\sqrt{\frac{1}{2}}$		
U+110	U+110 = 1		
U–110	U–110 = 1		
U+110	U+135 = 1		
U–110	U–135 = 1		
U+110	U+045 = UH+180 = $\sqrt{\frac{1}{2}}$		
U–110	U–045 = UH+180 = $\sqrt{\frac{1}{2}}$		
U+110	M+110 = 1		
U–110	M–110 = 1		
U+110	M+135 = 1		
U–110	M–135 = 1		
U+110	M+030 = $\sqrt{\frac{1}{2}}$		
U–110	M–030 = $\sqrt{\frac{1}{2}}$		
U+135	U+135 = 1		
U–135	U–135 = 1		
U+135	U+110 = 1		
U–135	U–110 = 1		
U+135	U+045 = $\sqrt{\frac{1}{3}}$, UH+180 = $\sqrt{\frac{2}{3}}$	9+10+3	
U–135	U–045 = $\sqrt{\frac{1}{3}}$, UH+180 = $\sqrt{\frac{2}{3}}$	9+10+3	
U+135	U+045 = UH+180 = $\sqrt{\frac{1}{2}}$		

ТАБЛИЦА 16 (*продолжение*)

Входное значение *speakerLabel*	**Выходные коэффициенты усиления воспроизведения**	`input_layouts`	`output_layouts`
U–135	U–045 = UH+180 = $\sqrt{\frac{1}{2}}$		
U+135	M+135 = 1		
U–135	M–135 = 1		
U+135	M+110 = 1		
U–135	M–110 = 1		
U+135	M+030 = $\sqrt{\frac{1}{2}}$		
U–135	M–030 = $\sqrt{\frac{1}{2}}$		
U+180	U+180 = 1		
U+180	UH+180 = 1		
U+180	U+135 = U–135 = $\sqrt{\frac{1}{2}}$		
U+180	U+110 = U–110 = $\sqrt{\frac{1}{2}}$		
U+180	M+135 = M–135 = $\sqrt{\frac{1}{2}}$		
U+180	M+110 = M–110 = $\sqrt{\frac{1}{2}}$		
U+180	M+030 = M–030 = $\sqrt{\frac{1}{4}}$		
UH+180	UH+180 = 1		
UH+180	U+180 = 1		
UH+180	U+135 = U–135 = $\sqrt{\frac{1}{2}}$		
UH+180	U+110 = U–110 = $\sqrt{\frac{1}{2}}$		
UH+180	M+135 = M–135 = $\sqrt{\frac{1}{2}}$		
UH+180	M+110 = M–110 = $\sqrt{\frac{1}{2}}$		
UH+180	M+030 = M–030 = $\sqrt{\frac{1}{4}}$		
T+000	T+000 = 1		
T+000	U+045 = U–045 = U+135 = U–135 = $\sqrt{\frac{1}{4}}$		
T+000	U+030 = U–030 = U+110 = U–110 = $\sqrt{\frac{1}{4}}$		
T+000	U+045 = U–045 = UH+180 = $\sqrt{\frac{1}{3}}$		
T+000	U+045 = U–045 = M+135 = M–135 = $\sqrt{\frac{1}{4}}$		

ТАБЛИЦА 16 (*окончание*)

Входное значение *speakerLabel*	**Выходные коэффициенты усиления воспроизведения**	`input_layouts`	`output_layouts`
T+000	U+030 = U–030 = M+110 = M–110 = $\sqrt{\frac{1}{4}}$		
T+000	M+030 = M–030 = M+135 = M–135 = $\sqrt{\frac{1}{4}}$		
T+000	M+030 = M–030 = M+110 = M–110 = $\sqrt{\frac{1}{4}}$		
T+000	M+030 = M–030 = $\sqrt{\frac{1}{4}}$		
B+000	B+000 = 1		
B+000	M+000 = 1		
B+000	M+030 = M–030 = $\sqrt{\frac{1}{2}}$		
B+045	B+045 = 1		
B–045	B–045 = 1		
B+045	M+030 = 1		
B–045	M–030 = 1		
LFE1	LFE1 = 1	9+10+3, 3+7+0	9+10+3, 3+7+0
LFE2	LFE2 = 1	9+10+3, 3+7+0	9+10+3, 3+7+0
LFE1	LFE1 = $\sqrt{\frac{1}{2}}$	9+10+3, 3+7+0	
LFE2	LFE1 = $\sqrt{\frac{1}{2}}$	9+10+3, 3+7+0	
LFE1	LFE1 = 1		

Библиография

[1] F. Zotter and M. Frank (2012), *All-round ambisonic panning and decoding*, *Journal of the audio engineering society*, vol. 60, no. 10, pp. 807-820.

[2] V. Pulkki, (1997), *Virtual sound source positioning using vector base amplitude panning*, *Journal of the audio engineering society*, vol. 45, no. 6, pp. 456-466.

Прилагаемый документ 1 к Приложению 1 (информативный)

Руководство к соответствующим частям спецификации метаданных ADM

A1.1 Метаданные ADM для рендерера ADM МСЭ-R

Цель нижеследующей таблицы состоит в том, чтобы предоставить сводный перечень ключевых элементов рендерера с указанием их местоположения в спецификациях, приведенных в Приложении 1. Спецификации следует брать из указанных ссылок.

Метаданные ADM ***подэлемент (атрибут)[система координат]***	**Рекомендация МСЭ-R BS.2076-1**	**Приложение1 к настоящей Рекомендации**
typeDefinition == "DirectSpeakers"	Пункт 5.4.3.1 Таблица 11	Пункт 8
speakerLabel		Пункт 8.2
position (azimuth, elevation, distance, screenEdgeLock)		Пункт 8
typeDefinition == "Matrix"	Пункт 5.4.3.2	Пункт 5.2.6.1.1 Пункт 5.2.6.4
outputChannelIDRef	Таблица 12	Пункт 5.2.6.1.1
matrix → coefficient (gain, gainVar, phase, phaseVar, delay, delayVar)	Таблица 13	Пункт 5.6.4
input/outputPackFormatIDRef	Пункт 5.5.5.1	Пункт 5.2.6.1.1
encode/decodePackFormatIDRef		Пункт 5.2.6.1.1
typeDefinition == "Objects"	Пункт 5.4.3.3	Пункт 7
position (azimuth, elevation, distance, screenEdgeLock) [*polar*]	Таблица 14	Пункт 6.1 Пункт 7 Пункт 7.3.4
position (X, Y, Z, screenEdgeLock) [*cartesian*]	Таблица 15	Пункт 6.1 Пункт 7 Пункт 7.3.10
width, height, depth [*polar*]	Таблица 14	Пункт 7.3.8
width, height, depth [*cartesian*]	Таблица 15	Пункт 7.3.11
cartesian	Таблица 16	Пункт 7.3.1 Пункт 7.3.2
gain		Пункт 7.3.1
diffuse		Пункт 7.3.1 Пункт 7.4
channelLock (maxDistance)		Пункт 7.3.6
objectDivergence (azimuthRange, positionRange) [*polar*]		Пункт 7.3.7 Пункт 7.3.1
objectDivergence (azimuthRange, positionRange) [*cartesian*]		Пункт 7.3.7 Пункт 7.3.1
jumpPosition (interpolationLength)		Пункт 7.2
zoneExclusion → zone (minX, maxX, minY, maxY, minZ, maxZ, minElevation, maxElevation, minAzimuth, maxAzimuth)		Пункт 7.3.5 Пункт 7.3.12
screenRef		Пункт 7.3.3
importance		Пункт 5.3.1 Пункт 5.2.7.1.1
typeDefinition == "HOA"	Пункт 5.4.3.4	Пункт 9 Пункт 5.2.7.3
equation	Таблица 17	Пункт 9.2
order		Пункт 9.1.1 Пункт 9.3.1.2

Метаданные ADM ***подэлемент (атрибут)[система координат]***	**Рекомендация МСЭ-R BS.2076-1**	**Приложение1 к настоящей Рекомендации**
degree		Пункт 9.1.1 Пункт 9.3.1.2
normalization	Пункт 5.4.3.4	Пункт 9.1.2 Пункт 9.3.1.5
nfcRefDist	Таблица 17	Пункт 9.2
screenRef		Пункт 9.2
typeDefinition == "Binaural"	Пункт 5.4.3.5	–

Прилагаемый документ 2 к Приложению 1 (информативный)

Альтернативная конфигурация виртуальных громкоговорителей

A2.1 Спецификация альтернативной конфигурации виртуальных громкоговорителей

Конфигурация виртуальных громкоговорителей VBAP, альтернативная той, что представлена в пункте 6.1.3.1, описывает позиции виртуальных громкоговорителей, не расположенных в полюсах, и их понижающие коэффициенты. Обработка метаданных ADM остается такой же, как указано в основной части настоящей Рекомендации, – без дополнительных метаданных. Альтернативные позиции виртуальных громкоговорителей и их понижающие коэффициенты основаны на оптимизациях "на слух". Ниже приведено описание этой альтернативной конфигурации виртуальных громкоговорителей.

A2.1.1 Процесс настройки

Процесс настройки выполняется в соответствии с шагами, описанными в пункте 6.1.3.1, за исключением второго шага, который должен быть следующим:

2) Сначала путем поиска в таблицах, приведенных в пункте A2.1.2, определяются виртуальные громкоговорители. В каждом подразделе пункта A2.1.2 определены конфигурация виртуальных громкоговорителей и их понижающие коэффициенты для конкретной схемы расположения, определенной в Рекомендации МСЭ-R BS.2051-2.

Другие шаги процесса настройки – шаг (1) и шаги (3)–(6) – остаются такими же, как описано в пункте 6.1.3.1.

A2.1.2 Таблица виртуальных громкоговорителей и понижающих коэффициентов

В приведенных ниже таблицах виртуальные громкоговорители (указанные азимутом и углом места) находятся в первой строке, а физические – в первом столбце. Номинальные и реальные позиции виртуальных громкоговорителей совпадают. В таблице приведены понижающие коэффициенты от виртуальных громкоговорителей к физическим.

Система A 0+2+0

Для системы A 0+2+0 используется метод, основанный на понижающем микшировании от системы B 0+5+0 к системе A 0+2+0, как описано в пункте 6.1.2.4. Для получения каналов 0+5+0 используются виртуальные громкоговорители системы B 0+5+0, как описано ниже.

Система B 0+5+0

	−45, 45	45, 45	−135, 45	135, 45	−45, −45	45, −45	−135, −45	135, −45
M+030		1,0				1,0		
M–030	1,0				1,0			
M+000								
LFE1								
M+110			0,3162	0,9486			0,3162	0,9486
M–110			0,9486	0,3162			0,9486	0,3162

Система C 2+5+0

	−135, 30	135, 30	−45, −45	45, −45	−135, −45	135, −45
M+030				1,0		
M–030			1,0			
M+000						
LFE1						
M+110	0,3162	0,9486			0,3162	0,9486
M–110	0,9486	0,3162			0,9486	0,3162
U+030						
U–030						

Система D 4+5+0

	−45, −45	45, −45	−110, −45	110, −45
M+030		1,0		
M–030	1,0			
M+000				
LFE1				
M+110			0,3162	0,9486
M–110			0,9486	0,3162
U+030				
U–030				
U+110				
U–110				

Система E 4+5+1

В этой схеме расположения присутствуют как верхний, так и нижний громкоговорители. Если оболочка заполнена, виртуальные громкоговорители не требуются.

Система F 3+7+0

	−135, 30	135, 30	−45, –45	45, −45	−135, −45	135, −45
M+000						
M+030				1,0		
M–030			1,0			
U+045						
U–045						
M+090						
M–090						
M+135		0,7071				1,0
M–135	0,7071				1,0	
UH+180	0,7071	0,7071				
LFE1						
LFE2						

Система G 4+9+0

	−45, −45	45, −45	−135, −45	135, −45
M+030		1,0		
M–030	1,0			
M+000				
LFE1				
M+090				
M–090				
M+135				1,0
M–135			1,0	
U+045				
U–045				
U+135				
U–135				
M+SC				
M–SC				

Система H 9+10+3

Содержит громкоговорители как в верхней, так и в нижней полусфере; оболочка заполнена, так что виртуальные громкоговорители не требуются.

Система I 0+7+0

	−45, 45	45, 45	−135, 45	135, 45	−45, −45	45, −45	−135, −45	135, −45
M+030		1,0						
M–030	1,0				1,0	1,0		
M+000								
LFE1								
M+090								
M–090								
M+135				1,0				1,0
M–135			1,0				1,0	

Система J 4+7+0

	−45, −45	45, −45	−135, −45	135, −45
M+030		1,0		
M–030	1,0			
M+000				
LFE1				
M+090				
M–090				
M+135				1,0
M–135			1,0	
U+045				
U–045				
U+135				
U–135				