Union internationale des télécommunications

UIT-R

Secteur des Radiocommunications de l'UIT

Recommandation UIT-R BS.2127-0

(06/2019)

Système de restitution ADM pour les systèmes sonores évolués

Série BS

Service de radiodiffusion sonore

Union internationale des télécommunications

Avant-propos

Le rôle du Secteur des radiocommunications est d'assurer l'utilisation rationnelle, équitable, efficace et économique du spectre radioélectrique par tous les services de radiocommunication, y compris les services par satellite, et de procéder à des études pour toutes les gammes de fréquences, à partir desquelles les Recommandations seront élaborées et adoptées.

Les fonctions réglementaires et politiques du Secteur des radiocommunications sont remplies par les Conférences mondiales et régionales des radiocommunications et par les Assemblées des radiocommunications assistées par les Commissions d'études.

Politique en matière de droits de propriété intellectuelle (IPR)

La politique de l'UIT-R en matière de droits de propriété intellectuelle est décrite dans la «Politique commune de l'UIT-T, l'UIT-R, l'ISO et la CEI en matière de brevets», dont il est question dans la Résolution UIT-R 1. Les formulaires que les titulaires de brevets doivent utiliser pour soumettre les déclarations de brevet et d'octroi de licence sont accessibles à l'adresse http://www.itu.int/ITU-R/go/patents/fr, où l'on trouvera également les Lignes directrices pour la mise en oeuvre de la politique commune en matière de brevets de l'UIT-T, l'UIT-R, l'ISO et la CEI et la base de données en matière de brevets de l'UIT-R.

Séries des Recommandations UIT-R

(Egalement disponible en ligne: http://www.itu.int/publ/R-REC/fr)

Séries	Titre
BO	Diffusion par satellite
BR	Enregistrement pour la production, l'archivage et la diffusion; films pour la télévision
BS	**Service de radiodiffusion sonore**
BT	Service de radiodiffusion télévisuelle
F	Service fixe
M	Services mobile, de radiorepérage et d'amateur y compris les services par satellite associés
P	Propagation des ondes radioélectriques
RA	Radio astronomie
RS	Systèmes de télédétection
S	Service fixe par satellite
SA	Applications spatiales et météorologie
SF	Partage des fréquences et coordination entre les systèmes du service fixe par satellite et du service fixe
SM	Gestion du spectre
SNG	Reportage d'actualités par satellite
TF	Emissions de fréquences étalon et de signaux horaires
V	Vocabulaire et sujets associés

***Note**: Cette Recommandation UIT-R a été approuvée en anglais aux termes de la procédure détaillée dans la Résolution UIT-R 1.*

Publication électronique
Genève, 2021

RECOMMANDATION UIT-R BS.2127-0[*, **]

Système de restitution ADM pour les systèmes sonores évolués

(2019)

Domaine d'application

La présente Recommandation contient la spécification du système de restitution de référence à utiliser, y compris pour l'échange de programmes, avec les systèmes sonores évolués spécifiés dans la Recommandation UIT-R BS.2051-2, et les métadonnées relatives aux signaux audio spécifiées dans la Recommandation UIT-R BS.2076-1 (Modèle de définition audio (ADM)). Le système de restitution convertit un ensemble de signaux audio avec des métadonnées associées en une configuration différente de signaux audio et de métadonnées, sur la base des métadonnées de contenus fournies, et des métadonnées de l'environnement local.

NOTE – Les lignes directrices relatives à l'utilisation du système de restitution sont en cours d'élaboration.

Mots clés

ADM, modèle de définition audio, métadonnées, système de restitution, AdvSS, système sonore évolué, signaux basés sur un canal, signaux basés sur un objet, signaux basés sur une scène, signaux multicanaux

L'Assemblée des radiocommunications de l'UIT,

considérant

a) que la Recommandation UIT-R BS.1909-0 (Qualité de fonctionnement requise d'un système sonore stéréophonique multicanal évolué destiné à être utilisé avec ou sans image associée) spécifie les exigences relatives aux systèmes sonores évolués utilisés avec ou sans image associée;

b) que la Recommandation UIT-R BS.2051-2 (Système sonore évolué pour la production de programmes) spécifie un système sonore évolué dont la configuration de reproduction surpasse celles spécifiées dans la Recommandation UIT-R BS.775-3, ou un système dont la configuration de reproduction peut prendre en charge des signaux d'entrée basés sur des canaux, sur des objets ou sur des scènes, ou leur association avec des métadonnées;

c) que la Recommandation UIT-R BS.2076-1 (Modèle de définition audio) spécifie la structure d'un modèle de métadonnées qui permet de décrire correctement le format et le contenu des fichiers audio;

d) que la Recommandation UIT-R BS.2094-1 (Définitions communes pour le modèle de définition audio) contient un ensemble de définitions communes pour le modèle de définition audio;

e) que la Recommandation UIT-R BS.2125-0 (Représentation série pour le modèle de définition audio) spécifie un format de métadonnées fondé sur le modèle de définition audio, segmenté en une série temporelle de trames;

---

[*] La présente Recommendation doit être portée à l'attention de l'Organisation internationale de normalisation (ISO), de la Commission électrotechnique internationale (CEI), de la Society of Motion Picture and Television Engineers (SMPTE) et de l'Institut européen des normes de télécommunications (ETSI).

[**] La Commission d'études 6 des radiocommunications a apporté des modifications de forme à la présente Recommandation en 2021 conformément à la Résolution UIT-R 1.

f) que la reproduction de systèmes sonores évolués nécessite de restituer les métadonnées associées aux signaux sonores afin de présenter le contenu dans l'une des configurations de haut-parleurs décrites dans la Recommandation UIT-R BS.2051-2;

g) que les utilisateurs de systèmes sonores évolués devraient être libres de choisir une méthode de restitution;

h) qu'il est souhaitable de disposer d'une spécification ouverte correspondant à une méthode unique de restitution de référence qui puisse être utilisée pour les programmes des systèmes sonores évolués;

i) que le système unique de restitution de référence devrait permettre aux producteurs et aux radiodiffuseurs de contenu de surveiller et de contrôler la qualité pendant la production du contenu, de vérifier l'utilisation des métadonnées, et de garantir l'interopérabilité avec les autres éléments de la chaîne de production,

recommande

1 que les méthodes de restitution décrites à l'Annexe 1 soient utilisées comme référence pour interpréter les métadonnées ADM spécifiées dans la Recommandation UIT-R BS.2076-1 et les signaux audio associés;

2 que la note 1 ci-dessous soit considérée comme partie intégrante de la Recommandation.

NOTE 1 – Le respect de cette Recommandation se fait à titre volontaire. Toutefois, celle-ci peut contenir des dispositions obligatoires (par exemple pour garantir l'interopérabilité ou l'applicabilité) et on considère que la Recommandation est respectée lorsque toutes ces dispositions obligatoires sont observées. Le futur d'obligation et les autres moyens d'expression de l'obligation comme le verbe "devoir" ainsi que leurs formes négatives servent à énoncer des prescriptions. L'utilisation de ces formes ne signifie pas qu'il est obligatoire de respecter totalement ou en partie la présente Recommandation.

Annexe 1

Spécifications relatives au système de restitution ADM pour les systèmes sonores évolués

TABLE DES MATIÈRES

1 Introduction

La présente Recommandation décrit un système de restitution audio fournissant une interprétation complète des métadonnées du modèle de définition audio (ADM) spécifiées dans la Recommandation UIT-R BS.2076-1. L'utilisation de métadonnées ADM est recommandée pour décrire les formats audio destinés à la production de programmes pour les systèmes sonores évolués (AdvSS), également appelés systèmes audio de prochaine génération (NGA). Ce système de restitution peut convertir les signaux audio dans toutes les configurations de haut-parleur spécifiées dans la Recommandation UIT-R BS.2051-2.

Cette spécification s'accompagne d'une mise en œuvre de référence à code source ouvert écrite en Python permettant le traitement du modèle ADM basé sur un fichier, disponible à l'adresse: https://www.itu.int/dms_pub/itu-r/oth/0a/07/R0A0700003E0001ZIPE.zip

Ce document de spécification fournit une description du code de référence.

1.1 Abréviations/glossaire

ADM	Modèle de définition audio *(Audio definition model)*
BMF	Format d'échange de métadonnées de radiodiffusion *(Broadcast metadata exchange format)*
BW64	Onde de radiodiffusion au format 64 *(Broadcast wave 64 format)*
BWF	Format d'onde de radiodiffusion *(Broadcast wave format)*
HOA	Signaux multicanaux d'ordre supérieur *(Higher-order ambisonics)*
NGA	Système audio de prochaine génération *(Next generation audio)*
PSP	Contrôleur de panoramique à source ponctuelle *(Point source panner)*
VBAP	Panoramique d'amplitude basé sur des vecteurs *(Vector base amplitude panning)*
XML	Langage de balisage extensible *(Extensible markup language)*

2 Conventions

2.1 Notations

Dans la présente Recommandation, les conventions suivantes seront utilisées:

– Le texte en italique désigne les éléments, sous-éléments, paramètres ou attributs du modèle ADM décrits dans la Recommandation UIT-R BS.2076-1: *audioObject*

– Le texte à chasse fixe désigne le code source (variables, fonctions, classes) de la mise en œuvre de référence: `core.point_source.PointSourcePanner`. Il est à noter que, dans un souci de visibilité, le préfixe `iar.` est omis.

– Les matrices sont indiquées par des lettres majuscules en gras: **X**

– Les vecteurs sont indiqués par des lettres minuscules en gras: **x**

– Les indices apparaissant sous la forme x_n indiquent le énième élément d'un vecteur **x**

– Les sections de texte à chasse fixe apparaissant en couleur servent à décrire la structure des données:

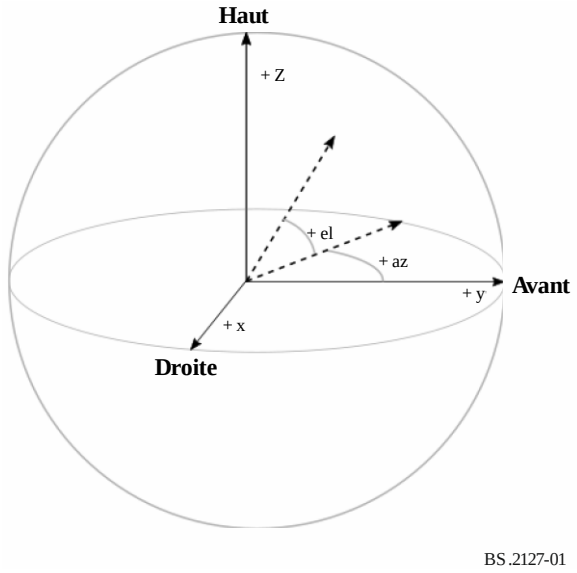
```
struct PolarPosition: Position {
  float azimuth, elevation, distance = 1;
};
```

2.2 Système de coordonnées

Les coordonnées cartésiennes et polaires sont utilisées tout au long du présent document.

FIGURE 1

Système de coordonnées



Les coordonnées polaires sont spécifiées comme suit conformément à la Recommandation UIT-R BS.2076-1:

– L'azimut, noté φ, désigne l'angle dans le plan horizontal, qui vaut 0 degré vers l'avant et qui prend des valeurs positives lorsqu'on se déplace dans le sens contraire des aiguilles d'une montre.

– L'élévation, notée θ, désigne l'angle situé au-dessus du plan horizontal, qui vaut 0 degré vers l'avant et qui prend des valeurs positives lorsqu'on se déplace vers le haut.

Les coordonnées cartésiennes sont spécifiées comme suit conformément à la Recommandation UIT-R BS.2076-1:

– L'axe positif des Y est dirigé vers l'avant.

– L'axe positif des X est dirigé vers la droite.

– L'axe positif des Z est dirigé vers le haut.

Le décodeur HOA mentionné au paragraphe 9 utilise le système de coordonnées et la notation HOA comme spécifié dans la Recommandation UIT-R BS.2076-1, où:

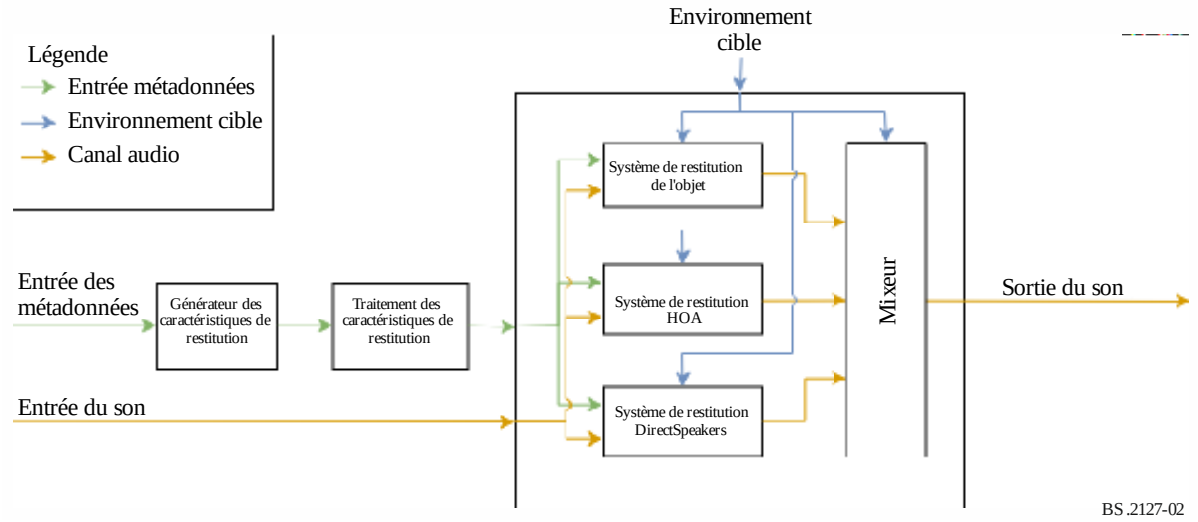
– L'élévation, notée θ, désigne l'angle en radians à partir de l'axe positif des Z.

– L'azimut, noté ϕ, désigne l'angle dans le plan horizontal en radians, qui vaut 0 vers l'avant et qui prend des valeurs positives lorsqu'on se déplace dans le sens contraire des aiguilles d'une montre.

3 Structure

FIGURE 2

Vue d'ensemble de l'architecture générale



L'architecture générale se compose de plusieurs composantes essentielles et étapes de traitement décrites dans les chapitres suivants du présent document.

– La conversion des données ADM en un ensemble de caractéristiques restituables est décrite au paragraphe 5.2.

– Le traitement facultatif permettant de spécifier le niveau d'importance et d'appliquer l'émulation des données de conversion aux caractéristiques de restitution est décrit au paragraphe 5.3.

– Le système de restitution est lui-même divisé en sous-composantes selon le type de caractéristique concerné (*typeDefinition*):

  • Le système de restitution des contenus basés sur des objets est décrit au paragraphe 7.
  • La restitution des signaux de haut-parleur direct est décrite au paragraphe 8.
  • La restitution de signaux HOA est décrite au paragraphe 9.
  • Les parties communes à toutes les composantes sont décrites au paragraphe 6.

Le traitement de type *Matrix* n'apparaît pas dans le diagramme car ce dernier est géré pendant la création des caractéristiques de restitution et dans le cadre des systèmes de restitution des autres types.

3.1 Comportement de l'environnement cible

Lors de l'initialisation, l'utilisateur peut sélectionner une configuration de haut-parleurs parmi celles spécifiées dans la Recommandation UIT-R BS.2051-2.

La position nominale de chaque haut-parleur (`polar_nominal_position`) est spécifiée dans la Recommandation UIT-R BS.2051-2. `M+SC` et `M-SC` ont des azimuts nominaux de 15° et −15°.

La position réelle de chaque haut-parleur (`polar_position`) peut être spécifiée par l'utilisateur. Si celle-ci n'est pas indiquée, la position nominale est utilisée. Les positions réelles indiquées sont vérifiées par rapport aux plages spécifiées dans la Recommandation UIT-R BS.2051-2; si elles ne sont pas conformes, un message d'erreur est généré. De plus, l'azimut absolu des deux haut-parleurs `M+SC` et `M-SC` doit être situé entre 5° et 25° ou entre 35° et 60°.

4 Interface XML du modèle ADM

L'ADM est un modèle de métadonnées générique qui peut être représenté naturellement comme un document XML. Les sous-sections ci-dessous décrivent comment le modèle ADM est mis en correspondance avec les structures de données internes. Ces dernières sont utilisées dans le cadre de la présente Recommandation et sont conformes aux structures de données utilisées dans la mise en œuvre de référence.

Il est à noter que, bien que le format XML soit généralement et communément utilisé pour représenter les métadonnées ADM, le système de restitution n'est pas limité à cette représentation.

La mise en correspondance du modèle ADM avec les structures de données internes obéit à un ensemble de règles simples décrites ci-dessous. Comme toutes les règles, celles-ci comportent des exceptions qui sont décrites dans les sous-sections suivantes.

– Tous les éléments du modèle ADM principal seront représentés sous la forme d'une sous-classe créée à partir de l'élément `ADMElement` ayant la signature suivante:

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

– Chaque classe d'élément du modèle ADM sera étendue à l'ensemble des attributs et sous-éléments du modèle ADM, lesquels sont mis en correspondance avec les attributs de classe.

– Si un sous-élément contient plus d'une valeur, il constitue une classe à lui seul. Par exemple, le sous-élément *jumpPosition* est une classe ayant pour signature:

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

– Pendant l'analyse syntaxique XML, les références aux autres éléments du modèle ADM sont stockées sous la forme d'identifiants simples utilisant le sous-élément comme nom d'attribut (p. ex., `AudioObject.audioPackFormatIDRef`). Pour simplifier l'accès ultérieurement, ces références sont ensuite résolues au cours d'une étape suivante, durant laquelle les éléments résolus sont ajoutés directement à chaque structure de données (`AudioObject.audioPackFormats`).

Conformément à ces règles, la signature complète de l'élément `AudioContent` est représentée comme suit:

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

Les éléments et les classes dédiées du modèle ADM principal sont exécutés dans `fileio.adm.elements.main_elements`. La résolution de la référence est exécutée dans chaque classe (dans le modèle `ADM` et chaque élément du modèle ADM principal) conformément à la méthode `lazy_lookup_references`.

L'analyse syntaxique et l'écriture du modèle ADM sont exécutées dans `fileio.adm.xml`.

4.1 AudioBlockFormat

L'élément *audioBlockFormat* se distingue des autres éléments du modèle ADM car ses sous-éléments et attributs diffèrent en fonction de l'élément *typeDefiniton*. Pour refléter cette différence, l'élément `AudioBlockFormat` est divisé en plusieurs classes, une par élément *typeDefinition* pris en charge: `AudioBlockFormatObjects`, `AudioBlockFormatDirectSpeakers` et `AudioBlockFormatHoa`.

Ces classes sont exécutées dans `fileio.adm.elements.block_formats`.

4.2 Sous-éléments relatifs à la position

Les positions sont représentées par plusieurs sous-éléments *position* dans le modèle ADM. Pour simplifier la gestion interne, les valeurs de ces sous-éléments sont combinées en un seul attribut dans la représentation de l'élément `AudioBlockFormat`.

Pour l'attribut *typeDefinition==Objects*, la position sera de type `ObjectPolarPosition` ou `ObjectCartesianPosition`, selon le système de coordonnées utilisé. Pour l'attribut *typeDefinition==DirectSpeakers*, la position sera de type `DirectSpeakerPolarPosition` ou `DirectSpeakerCartesianPosition`.

4.3 TypeDefinition

Les attributs *typeDefinition* et *typeLabel* décrivent une seule propriété. Par conséquent, une seule entité sera utilisée en interne pour les représenter.

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

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

5 Caractéristiques de restitution

Un élément `RenderingItem` est la représentation d'une caractéristique du modèle ADM devant être restituée, et contenant toutes les informations nécessaires à cet effet. Une caractéristique pourra donc représenter un seul élément *audioChannelFormat* ou un groupe d'éléments *audioChannelFormats*. Dans la mesure où les prescriptions appliquées à chaque attribut *typeDefinition* sont différentes, il convient de disposer de structures de métadonnées différentes afin de s'adapter aux besoins propres à chacun.

La section suivante décrit plus en détail les structures de métadonnées utilisées.

5.1 Structures de métadonnées

Les éléments `RenderingItems` reposent sur les classes de base suivantes:

- Élément `TypeMetadata` contenant tous les paramètres (potentiellement variables dans le temps) nécessaires pour restituer la caractéristique;
- Élément `MetadataSource` contenant une série d'objets `TypeMetadata`; et

– Élément `RenderingItem` associant un élément `MetadataSource` et une source d'échantillons audio ainsi que des informations supplémentaires n'étant pas nécessairement requises par le système de restitution.

Dans la mesure où les prescriptions appliquées à chaque attribut *typeDefinition* sont différentes, il convient de sous-classer les éléments `TypeMetadata` et `RenderingItem` selon chaque attribut *typeDefinition* afin de s'adapter aux besoins propres à chacun. L'élément `MetadataSource` est indépendant de l'attribut *typeDefinition*. Les données communes sont regroupées dans l'élément `ExtraData`:

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

Les données relatives au niveau d'importance seront stockées dans une structure `ImportanceData`:

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

Les références aux échantillons de signaux audio seront encapsulées dans des structures `TrackSpec` afin de permettre la spécification de pistes silencieuses et le traitement de type Matrix. L'élément `DirectTrackSpec` indique que les échantillons seront lus directement à partir de la piste d'entrée spécifiée. L'élément `SilentTrackSpec` indique que les échantillons seront tous à zéro.

```
struct TrackSpec {};

struct DirectTrackSpec: TrackSpec {
  int track_index;
};

struct SilentTrackSpec: TrackSpec {
};
```

Deux types d'éléments `TrackSpec` permettent de prendre en charge l'attribut *typeDefinition==DirectSpeakers*. L'élément `MatrixCoefficientTrackSpec` indique que les paramètres spécifiés dans le `coefficient` (à partir d'un élément de `coefficient` de type Matrix *audioBlockFormat*) sont appliqués aux échantillons de l'élément `input_track`, tandis que l'élément `MixTrackSpec` indique que les échantillons de plusieurs éléments `TrackSpec`s doivent être mixés ensemble.

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

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

Cette action est exécutée dans `core.utils.metadata_input`. Les sous-sections suivantes décrivent plus en détail les mises en œuvre spécifiques à chaque attribut *typeDefinition*.

5.1.1 DirectSpeakers

Pour l'attribut *typeDefinition==DirectSpeakers* l'élément `TypeMetadata` doit contenir l'élément *audioBlockFormat*, la liste des éléments *audioPackFormat*s conduisant à l'élément *audioChannelFormat* qui les renferme, ainsi que les données communes recueillies dans l'élément `ExtraData`.

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

Dans la mesure où chaque élément *audioChannelFormat* comportant un attribut *typeDefinition==DirectSpeakers* peut être traité indépendamment, l'élément `RenderingItem` ne contient qu'un seul élément `TrackSpec`.

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

5.1.2 Matrice

L'attribut *typeDefinition==Matrix* sera pris en charge par le mécanisme `TrackSpec` pour restituer les caractéristiques des autres types, de sorte qu'aucune classe explicite n'est requise pour les éléments `MatrixTypeMetadata` ou `MatrixRenderingItem`.

5.1.3 Objets

L'élément `ObjectTypeMetadata` doit contenir un élément *audioBlockFormat* ainsi que les données communes recueillies dans l'élément `ExtraData`.

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

Dans la mesure où chaque élément *audioChannelFormat* comportant un attribut *typeDefinition==DirectObjects* peut être traité indépendamment, l'élément `RenderingItem` ne contiendra qu'un seul élément `TrackSpec`.

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

5.1.4 HOA

Pour l'attribut *typeDefinition==HOA*, la situation est différente de celle concernant les attributs *typeDefinition==DirectSpeakers* et *typeDefinition==Objects*, car un paquet d'éléments *audioChannelFormats* doit être traité simultanément. En conséquence, l'élément `HOATypeMetadata` ne contient pas l'élément `audioBlockFormat` plus l'élément `ExtraData`; les informations nécessaires sont extraites des éléments *audioBlockFormats* et directement stockées dans l'élément `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;
  };
```

Pour la même raison, la situation est également différente en ce qui concerne l'élément `HOARenderingItem`. Dans ce cas, l'élément `HOARenderingItem` contient non seulement un élément `TrackSpec`, mais également un vecteur de l'élément `TrackSpec`s.

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

5.1.5 Binaural

L'attribut *typeDefinition==Binaural* n'étant pas pris en charge, il n'existe aucune classe pour l'élément `BinauralTypeMetadata` ou `BinauralRenderingItem`.

5.2 Détermination des caractéristiques de restitution

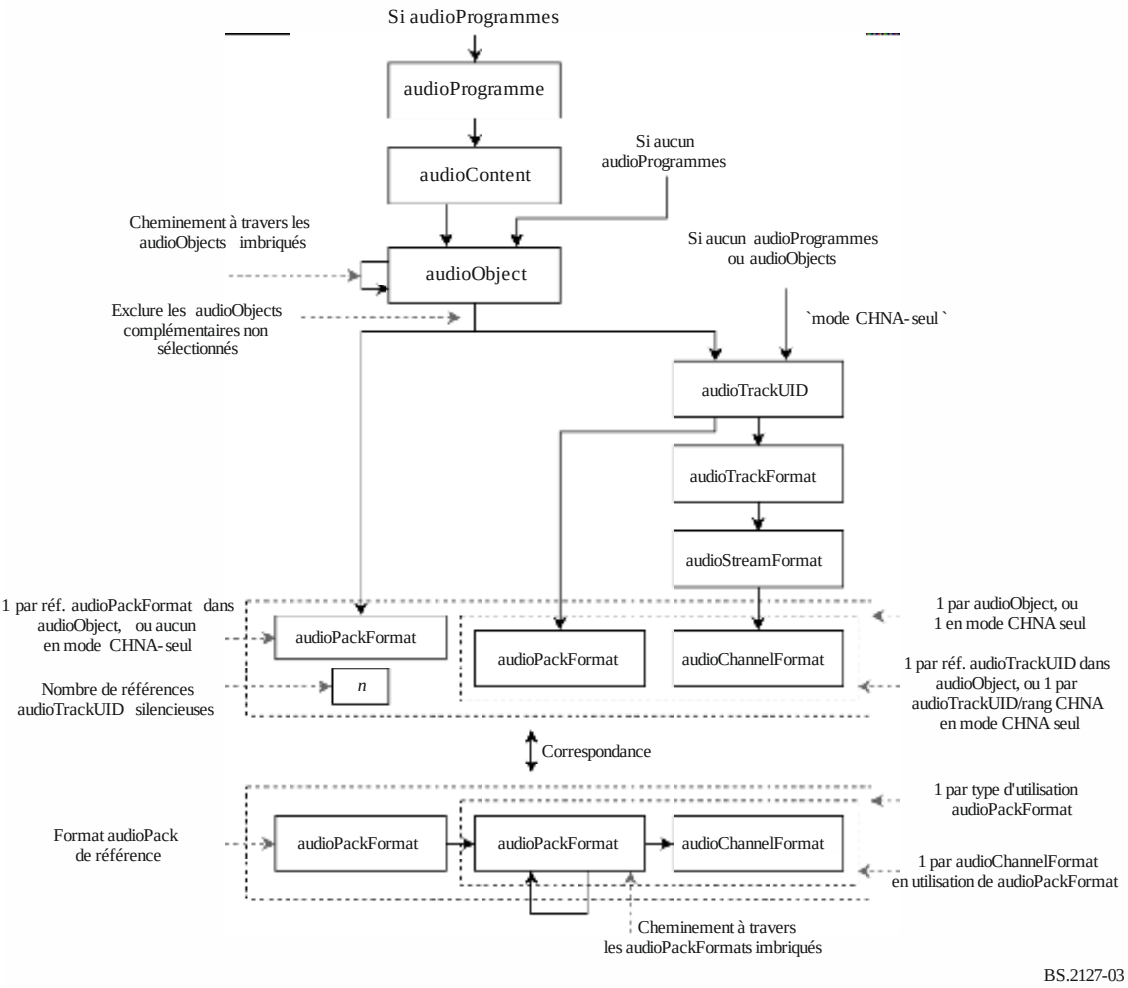
Pour déterminer les éléments `RenderingItems`, il convient d'analyser la structure du modèle ADM. La Figure 3 illustre la marche à suivre.

L'état du processus de sélection de la caractéristique est acheminé entre les différentes composantes dans un même objet appelé "état de sélection de la caractéristique", lequel, une fois complété, représente toutes les composantes qui forment un seul et même élément `RenderingItem`. Chaque composante accepte un seul état de sélection de la caractéristique, dont il renvoie des copies (de zéro à plusieurs) comportant un plus grand nombre d'entrées renseignées. Ces étapes sont regroupées dans l'élément `select_rendering_items`, une boucle imbriquée autour des états lorsqu'ils sont modifiés à tour de rôle par chaque composante.

Cette action est exécutée dans `core.select_items`.

FIGURE 3

Trajet à suivre dans la structure du modèle ADM pour déterminer les éléments `RenderingItems`



5.2.1 Point de départ

La sélection d'une caractéristique de restitution peut se faire à partir de plusieurs points de la structure du modèle ADM selon les éléments contenus dans le fichier.

S'il existe plusieurs éléments *audioProgramme*, un seul est sélectionné; s'il existe plusieurs éléments *audioObject*, tous seront sélectionnés; enfin, tous les éléments *audioTrackUID*s (rangs CHNA) sont sélectionnés (il s'agit du "mode CHNA seul").

5.2.2 Sélection de l'élément audioProgramme

Un seul élément *audioProgramme* est sélectionné. Le programme à utiliser peut être sélectionné par l'utilisateur. Si aucun élément *audioProgramme* n'est sélectionné, l'élément sélectionné sera celui ayant l'identifiant le plus bas numériquement.

5.2.3 Sélection de l'élément audioContent

Tous les éléments *audioContent*s référencés par l'élément *audioProgramme* sélectionné sont sélectionnés.

5.2.4 Sélection de l'élément audioObject

L'élément `audioObjects` sera réglé sur tous les trajets possibles de la hiérarchie en commençant par l'élément *audioContent* sélectionné (en suivant les liens de l'élément *audioObject*).

5.2.5 Gestion de l'élément *audioObject* complémentaire

Les références de type *audioComplementaryObject* seront interprétées comme des groupes de définition des éléments *audioObjects*, dont un seul sera reproduit.

Un groupe est décrit par les références de type *audioComplementaryObject* à partir de l'élément *audioObject* par défaut vers tous les éléments *audioObject*s du groupe autres que ceux par défaut. L'utilisateur peut fournir un ensemble d'éléments *audioObject*s à sélectionner afin d'ignorer les éléments par défaut. À partir de cette action, un ensemble d'éléments *audioObject*s à ignorer est déterminé, et les états sont rejetés lorsque l'un des éléments *audioObject*s apparaissant sur le trajet de l'élément *audioObject* concerné figure dans cet ensemble.

5.2.5.1 Sélection des éléments *audioObjects* complémentaires à ignorer

Dans un premier temps, l'ensemble d'éléments *audioObject*s sélectionné par l'utilisateur sera complété par les éléments par défaut de chaque groupe: pour chaque élément racine *audioObject* (un élément *audioObject* comportant des références de type *audioComplementaryObject*), si aucun des éléments *audioObject*s du groupe défini par l'élément racine *audioObject* de ce groupe n'apparaît dans l'ensemble, l'élément racine *audioObject* (par défaut) sera ajouté.

L'ensemble des éléments *audioObject*s à ignorer correspond ensuite à l'ensemble des éléments audioObjects complémentaires (c'est-à-dire les éléments *audioObject*s comportant une référence de type *audioComplementaryObject* et les éléments audioObjects indiqués par une référence de type *audioComplementaryObject*) moins l'ensemble d'éléments *audioObject*s sélectionné par l'utilisateur.

Si des éléments *audioObject*s n'appartenant à aucun groupe complémentaire sont sélectionnés, ou si plusieurs éléments *audioObject*s sont sélectionnés dans un seul groupe d'éléments *audioObject* (en raison d'une erreur de l'utilisateur ou d'un chevauchement des groupes), une erreur est générée.

5.2.6 Mise en correspondance de l'élément *audioPackFormat*

L'étape suivante consiste à mettre en correspondance les informations d'un élément *audioObject* (c'est-à-dire la liste des éléments *audioPackFormat*s et *audioTrackUID*s et le nombre de pistes silencieuses, ou simplement la liste de tous les éléments *audioTrackUID*s en mode CHNA seul) avec les structures des éléments *audioPackFormat* et *audioChannelFormat*.

Dans la mesure où plusieurs éléments doivent correspondre de part et d'autre, et non entrer en conflit, pour constituer une solution valide, cette action est spécifiée comme un problème de mise en correspondance/recherche plutôt que sous la forme de trajets spécifiques à résoudre dans les structures de référence.

La correspondance est considérée comme valide uniquement si une seule solution est trouvée. Si aucune solution n'est trouvée, les métadonnées sont contradictoires et une erreur sera générée. Si plusieurs solutions sont trouvées, les métadonnées sont ambiguës et une erreur sera générée. Pour ces deux types d'erreurs, des diagnostics sont exécutés afin d'indiquer à l'utilisateur les causes d'erreur potentielles.

5.2.6.1 Paquets à mettre en correspondance

La spécification des éléments *audioPackFormat*s à mettre en correspondance est fournie dans une liste énumérant les structures `AllocationPack`:

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

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

Chaque structure doit spécifier la racine de l'élément *audioPackFormat* (`root_pack`, l'élément *audioPackFormat* de niveau supérieur référençant tous les canaux à attribuer), ainsi qu'une liste des canaux à mettre en correspondance dans ce paquet. Chaque canal combine une référence de type *audioChannelFormat* et une liste d'éléments *audioPackFormat*s potentiels auxquels pourrait être associé ce canal.

Pour chaque élément *audioPackFormat* `pack` comportant un attribut *typeDefinition != Matrix*, un objet `AllocationPack` est créé comme suit:

– La valeur de l'élément `root_pack` est `pack`.

– L'élément `channels` contient une entrée pour chaque élément *audioChannelFormat* accessible depuis l'élément `pack` (en suivant de manière récursive les liens *audioPackFormat*), lorsque l'élément `pack_formats` contient tous les éléments *audioPackFormat*s sur le trajet entre les éléments `pack` et *audioChannelFormat* (`pack` inclus).

Bien qu'il s'agisse d'une légère simplification de la structure des éléments *audioPackFormat* et *audioChannelFormat*, l'avantage de cette représentation est sa capacité à reproduire les structures de référencement des éléments *audioPackFormat* et *audioChannelFormat* utilisées avec le contenu de l'élément *Matrix* décrit ci-après.

5.2.6.1.1 Gestion de l'élément Matrix

Les éléments *audioPackFormats* de type Matrix peuvent être référencés de multiples façons selon l'effet souhaité. Ces structures de référence sont reflétées dans les éléments `AllocationPack`s produits pour chaque élément audioPackFormat `pack`comportant un attribut *typeDefinition==Matrix*:

– Si l'élément `pack` est une matrice directe ou une matrice de décodage, la matrice doit être appliquée dès lors qu'un élément *audioObject* référence à la fois un élément `pack` et un ensemble d'identifiants *audioTrackUID*s qui référencent à leur tour l'élément `pack` et les canaux d'entrée, ou qui codent le format *audioPackFormat* de l'élément `pack`:

  • La valeur de l'élément `root_pack` est pack.

  • L'élément`channels` contient une valeur par canal *audioChannelFormat* dans l'élément *audioPackFormat* du paquet en entrée (l'élément *encodePackFormat* ou *inputPackFormat*, selon le type), où la valeur de l'élément `channel_format` est `channel` et celle de l'élément `pack_formats` est `[pack]`.

– Si l'élément `pack` est une matrice directe ou une matrice de décodage, la matrice doit être considérée comme ayant été préalablement appliquée aux échantillons du fichier dès lors qu'un élément *audioObject* référence à la fois un élément `pack` et un ensemble d'identifiants *audioTrackUID*s qui référencent à leur tour l'élément `pack` (ou les sous-paquets) et les canaux de l'élément `pack`:

  • La valeur de l'élément `root_pack` est `pack`.

  • L'élément `channels` contient une valeur par élément *audioChannelFormat* `channel` contenu dans l'élément `pack`, où la valeur de l'élément `channel_format` est `channel`, et l'élément `pack_formats` contient tous les éléments *audioPackFormat*s sur le trajet entre les éléments `pack` et `channel`.

– Si l'élément `pack` est une matrice de décodage, l'élément *encodePackFormat* correspondant suivi par l'élément `pack` peut être appliqué dès lors qu'un élément *audioObject* référence à la fois un élément `pack` et un ensemble d'identifiants *audioTrackUID*s qui référencent à leur tour l'élément *encodePackFormat* et les canaux de l'élément *inputPackFormat* du format *encodePackFormat*:

  • La valeur de l'élément `root_pack` est `pack`.

  • L'élément`channels` contient une valeur par élément audioChannelFormat `channel` contenu dans l'élément inputPackFormat de l'élément encodePackFormat de l'élément `pack`, où la valeur de l'élément `channel_format` est `channel`, et l'élément `pack_formats` contient tous les éléments audioPackFormats sur le trajet entre le format inputPackFormat et l'élément `channel`.

Le "type" de matrice de l'élément *audioPackFormat* est déterminé selon les règles suivantes:

– Lorsqu'il existe à la fois une référence de type *inputPackFormat* et de type *outputPackFormat*, il s'agit d'une matrice directe.

– Lorsqu'il existe une référence de type *inputPackFormat* mais aucune de type *outputPackFormat*, il s'agit d'une matrice de codage.

– Lorsqu'il existe une référence de type *outputPackFormat* mais aucune de type *inputPackFormat*, il s'agit d'une matrice de décodage.

– Lorsqu'il n'existe aucune référence de type *inputPackFormat* ni *outputPackFormat*, une erreur est générée.

5.2.6.2 Pistes et références de type *audioPackFormat* à mettre en correspondance

Les pistes à mettre en correspondance avec les éléments `AllocationPack`s seront indiquées par trois valeurs:

– Les éléments `tracks`, une liste d'éléments `AllocationTrack`s dont chacun représente un élément *audioTrackUID* (ou rang CHNA):

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

L'élément `channel_format` est obtenu à partir d'un élément *audioTrackUID* en suivant les références de type *audioTrackFormat*, *audioStreamFormat* et *audioChannelFormat*, tandis que l'élément `pack_format` est référencé directement par l'élément *audioTrackUID*.

– Les éléments `pack_refs`, une liste facultative des références de type *audioPackFormat* contenues dans un élément *audioObject*.

L'élément num_silent_tracks, qui correspond au nombre de pistes silencieuses à attribuer, représenté dans les références d'un élément *audioObject* dont la valeur est fixée à ATU_00000000.

Lorsque l'on détermine ces structures pour un élément *audioObject*:

- L'élément tracks contient une entrée pour chaque élément *audioTrackUID* (non silencieux) référencé à partir de l'élément *audioObject*.
- L'élément pack_refs est une liste de références de type *audioPackFormat* contenues dans l'élément *audioObject*.
- L'élément num_silent_tracks correspond au nombre d'éléments *audioTrackUID*s silencieux référencés (soit les références d'un élément *audioObject* dont la valeur est fixée à ATU_00000000).

tandis qu'en mode CHNA seul:

- L'élément tracks contient une entrée pour chaque élément *audioTrackUID* (ou rang CHNA) du fichier.
- La valeur de l'élément pack_refs est None.
- La valeur de l'élément num_silent_tracks est 0.

5.2.6.3 Mise en correspondance

Une solution de mise en correspondance est spécifiée sous forme de liste des objets de l'élément AllocatedPack:

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

Chaque solution associe chaque élément *audioChannelFormat* du pack avec un élément track, ou avec une piste silencieuse si l'élément AllocationTrack n'est pas spécifié.

Pour être valable, une solution doit présenter les caractéristiques suivantes:

1) Pour chaque élément AllocatedPack, chaque canal de l'élément AllocationPack intervient exactement une fois dans l'élément allocation.
2) Chaque piste de l'élément tracks apparaît exactement une fois à la sortie.
3) Le nombre de pistes silencieuses référencées en sortie est égal à num_silent_tracks.
4) Pour chaque élément AllocationChannel channel et AllocationTrack track associé, la valeur de l'élément track.channel_format est channel.channel_format, et celle de l'élément track.pack_format est channel.pack_formats.
5) Si la valeur de l'élément pack_refs n'est pas None, il existe une correspondance un à un entre l'élément pack_refs et les valeurs de l'élément pack.pack.root_pack pour chaque élément AllocatedPack pack.

Les solutions identiques, à l'exception de l'ordre des éléments AllocationPacks ou allocations contenus par ces derniers, sont considérées comme équivalentes.

Toute méthode permettant de déterminer toutes les solutions valables et uniques (non équivalentes) peuvent être utilisées. Dans la mise en œuvre de référence, la recherche de solutions s'effectue en envisageant les caractéristiques ci-dessus comme un problème de satisfaction des contraintes et en déterminant toutes les solutions possibles au moyen d'une recherche par retour en arrière.

5.2.6.3.1 Exemples

La mise en correspondance des formats de paquets est illustrée dans la série d'exemples ci-dessous.

Les structures utilisées dans ces exemples sont définies comme suit: `c1`, `c2`, etc. et `p1`, `p2`, etc. représentent les références aux éléments *audioChannelFormat*s et *audioPackFormat*s (mais il peut s'agir de tout objet, dans la mesure où l'élément `allocate_packs` utilise uniquement les informations contenues dans les structures de type `Allocation...` en comparant ces références en fonction de l'identité).

Un paquet au format mono et une piste y faisant référence:

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

Un paquet au format à deux canaux comprenant deux paires de pistes de référencement:

```
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)
```

La résolution d'une seule piste mono dans un élément *audioObject* donne une solution unique contenant un seul paquet attribué:

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

La résolution d'une seule piste mono en mode CHNA seul donne la même structure:

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

La résolution d'une seule piste silencieuse donne la même structure, hormis le fait que la référence à la piste est remplacée par la valeur `None`:

```
assert allocate_packs(
    packs=[ap1, ap2],
    tracks=[],
    pack_refs=[p1],
    num_silent_tracks=1,
) == [[AllocatedPack(pack=ap1, allocation=[(ac1, None)])]]
```

S'il y a plus de pistes que de canaux disponibles dans les références du paquet, le conflit généré entre les règles 2 et 5 empêchera toute solution:

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

S'il y a plus de pistes silencieuses que de canaux disponibles dans les références du paquet, le conflit généré entre les règles 2 et 5 empêchera toute solution:

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

En cas de non-concordance entre les références du paquet et les informations du canal/paquet contenues dans les pistes, le conflit généré entre les règles 1, 4 et 5 empêchera toute solution:

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

S'il y a plusieurs occurrences d'un paquet à canaux multiples dans un élément *audioObject*, l'attribution des pistes aux paquets est ambiguë, ce qui donne plusieurs solutions:

```
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 Post-traitement des solutions

Il est à noter que les résultats de la mise en correspondance sont spécifiés conformément aux structures d'entrée (`AllocationPack`, `AllocationChannel`, `AllocationTrack`), plutôt que selon les références sous-jacentes aux structures du modèle ADM. Cela permet d'effectuer une correspondance arbitraire entre les références de type *audioPackFormat* et *audioChannelFormat* (dans les éléments *audioObject* et *audioTrackUID*) et les informations transmises au système de restitution, dans la mesure où il n'existe pas de correspondance simple lorsque l'attribut *typeDefinition==Matrix* est utilisé.

Pour un élément `AllocatedPack pack` non matriciel, la correspondance est simple et directe; la valeur de l'élément `output_pack` est `pack.pack.root_pack`, et il existe une correspondance un à un entre les attributions dans l'élément `pack.allocation` et l'attribution du canal réel:

l'élément `AllocationChannel channel` est mis en correspondance avec l'élément `channel.channel_format`, l'élément `AllocationTrack track` est mis en correspondance avec un élément `DirectTrackSpec` pour l'index des pistes de l'élément *audioTrackUID* (ou rang CHNA) associé à l'élément `track`, et un élément `AllocationTrack` manquant est mis en correspondance avec un élément `SilentTrackSpec`.

Pour un élément `AllocatedPack pack` de type matriciel, une correspondance plus complexe est requise:

l'élément `pack.root_pack` étant toujours un paquet direct ou un paquet de décodage, (voir paragraphe 5.2.6.1.1), la valeur de l'élément `output_pack` est `pack.root_pack.outputPackFormat`.

Le canal de sortie vers l'attribution de pistes contient une entrée par élément *audioChannelFormat* `matrix_channel` dans l'élément `root_pack`. Il existe une correspondance un à un entre ces canaux et les éléments *audioChannelFormats* de l'élément `output_pack` établi par les références de type *outputChannelFormat*.

La valeur de l'élément *audioChannelFormat* est matrix_channel.block_formats[0]. outputChannelFormat.

L'élément `TrackSpec` est créé en suivant de manière récursive les références de l'élément *inputChannelFormat* de l'élément `matrix_channel` jusqu'aux éléments *audioChannelFormat*s référencés dans l'élément `pack.allocation`, en imbriquant les éléments `MatrixCoefficientTrackSpec`s et `MixTrackSpec`s, afin d'appliquer le traitement spécifié dans les éléments de type *coefficient* et de mixer ensemble plusieurs canaux d'entrée:

– Si l'élément `matrix_channel` est référencé dans un élément `pack.allocation`, renvoi d'un élément `DirectTrackSpec` ou `SilentTrackSpec` correspondant à l'élément `AllocationTrack` associé (voir ci-dessus).

– Sinon, renvoi d'un élément `MixTrackSpec` contenant un élément `MatrixCoefficientTrackSpec` pour chaque élément de type *coefficient* `c` contenu dans un élément `matrix_channel.block_formats[0].matrix` appliquant le traitement indiqué dans l'élément `c` à la piste spécifiée pour l'élément `c.inputChannelFormat`, déterminé de manière récursive.

Dans la mise en oeuvre de référence, cette action est exécutée dans deux sous-classes de l'élément `AllocationPack`, lesquelles permettent d'interroger l'élément *audioPackFormat* et d'attribuer le canal à utiliser par le système de restitution. De la même façon, l'association des éléments `AllocationTrack`s et de leurs identifiants *audioTrackUID*s correspondants est exécutée en utilisant une sous-classe de l'élément `AllocationTrack`.

5.2.7 Caractéristiques de restitution en sortie

Lorsque la racine de l'élément *audioPackFormat* a été déterminée et qu'un élément `TrackSpec` a été attribué à chacun des canaux, toutes les informations trouvées sont converties en un ou plusieurs éléments `RenderingItem`s.

Le processus utilisé à cet effet dépend du type de racine de l'élément *audioPackFormat*.

5.2.7.1 Composantes partagées

Certaines données des caractéristiques de restitution sont communes à plusieurs types, et sont donc établies de la même façon.

5.2.7.1.1 Importance

Un objet `ImportanceData` doit être créé à partir de l'état de sélection de la caractéristique à l'aide des valeurs suivantes:

– L'élément `audio_object` correspond à l'importance minimale spécifiée dans tous les éléments *audioObject*s du trajet.

– L'élément `audio_pack_format` correspond à l'importance minimale spécifiée dans tout élément *audioPackFormat* sur le trajet entre la racine de l'élément *audioPackFormat* et l'élément *audioChannelFormat*.

Dans les deux cas, la valeur `None` (importance non spécifiée) correspond à l'importance la plus élevée.

5.2.7.1.2 Données supplémentaires

Un objet `ExtraData` doit être créé à partir de l'état de sélection de la caractéristique à l'aide des valeurs suivantes:

– L'élément `object_start` correspond à l'heure de début (*start*) du dernier élément *audioObject* sur le trajet (valeur `None` en mode CHNA seul).

– L'élément `object_duration` correspond à la durée (*duration*) du dernier élément *audioObject* sur le trajet (valeur `None` en mode CHNA- seul).

– L'élément `reference_screen` correspond à la valeur *audioProgrammeReferenceScreen* de l'élément *audioProgramme* sélectionné (valeur `None` si aucun élément n'a été sélectionné).

– L'élément `channel_frequency` correspond à la fréquence (*frequency*) de l'élément *audioChannelFormat* sélectionné (ou à la valeur `None` si aucun élément n'a été sélectionné, comme lors de la création d'une caractéristique de restitution de signaux HOA).

5.2.7.2 Caractéristiques de restitution en sortie pour l'élément typeDefinition==Objects or DirectSpeakers

Le processus permettant de déterminer les caractéristiques de restitution des éléments *Objects* et *DirectSpeakers* est similaire; seuls diffèrent les types concernés et la sélection des paramètres.

Chaque paire constituée d'un élément *audioChannelFormat* et d'un élément `track_spec` produit une caractéristique de restitution.

Un élément `MetadataSource` est créé, lequel produit un élément `RenderingItem` (du type approprié) par élément *audioBlockFormat* dans l'élément *audioChannelFormat* sélectionné, où le champ de l'élément `extra_data` est déterminé comme indiqué ci-dessus, et le champ des éléments `audioPackFormats` contient tous les éléments *audioPackFormat*s sur le trajet entre la racine de l'élément *audioPackFormat* et l'élément *audioChannelFormat*. Cet élément est enveloppé dans un objet `RenderingItem` (là encore, du type approprié) avec les éléments `track_spec` et `importance` déterminés comme indiqué ci-dessus.

5.2.7.3 Caractéristiques de restitution en sortie pour l'élément *typeDefinition==HOA*

Chaque paquet d'attribution produit un élément `HOARenderingItem` qui contient toutes les informations nécessaires à la restitution d'un groupe de canaux constituant un flux HOA. Les informations sont diffusées dans les différents éléments *audioChannelFormats* et *audioPackFormats* (lorsqu'ils sont imbriqués), lesquels doivent être homogènes.

Les éléments *audioChannelFormat*s de type HOA doivent contenir un seul élément *audioBlockFormat*, sinon une erreur est générée.

Un seul objet NHOATypeMetadata est créé avec des paramètres établis conformément au Tableau 1.

TABLEAU 1

Propriétés des paramètres de l'élément HOATypeMetadata

Paramètre de l'élément HOATypeMetadata	Paramètre de l'élément *audioBlockFormat*	Paramètre de l'élément *audioPackFormat*	nombre de valeurs
rtime	*rtime*		une seule
duration	*duration*		une seule
orders	*order*		autant que de canaux
degrees	*order*		autant que de canaux
normalization	*normalization*	*normalization*	une seule
nfcRefDist	*nfcRefDist*	*nfcRefDist*	une seule
screenRef	*screenRef*	*screenRef*	une seule

Tous les paramètres doivent d'abord être déterminés pour chaque élément *audioChannelFormat* contenu dans la racine de l'élément *audioPackFormat*. Les paramètres qui s'appliquent à la fois aux éléments *audioBlockFormat* et *audioPackFormat* peuvent être réglés uniquement sur l'élément *audioBlockFormat* contenu dans l'élément *audioChannelFormat*, ou sur tout élément *audioPackFormat* situé sur le trajet entre la racine de l'élément *audioPackFormat* et l'élément *audioChannelFormat*. Si un paramètre apparaît plusieurs fois pour un même élément *audioChannelFormat*, celui-ci devra avoir la même valeur à chaque fois, sinon une erreur sera générée. Si aucune valeur n'est trouvée pour un paramètre et un élément *audioChannelFormat* donnés, la valeur appliquée par défaut sera celle spécifiée dans la Recommandation UIT-R BS.2076-1.

Lorsqu'un paramètre *nfcRefDist* a été trouvé pour un certain élément *audioChannelFormat*, une valeur de 0 se traduira par une valeur None, ce qui signifie que la compensation en champ proche (NFC) ne sera pas appliquée. Cette opération est effectuée à ce stade (plutôt que pendant l'analyse syntaxique XML) afin que le paramètre *nfcRefDist==0.0* soit considéré comme entrant en conflit avec le paramètre *nfcRefDist==1.0*, par exemple.

S'agissant des paramètres à une seule valeur (c'est-à-dire tous à l'exception des ordres [orders] et des degrés [degrees]), les paramètres déterminés pour tous les éléments *audioChannelFormats* devront être identiques, sinon une erreur sera générée.

La valeur extra_data est déterminée comme indiqué ci-dessus pour l'ensemble de l'élément *audioPackFormat*.

Chaque caractéristique d'attribution de canal (comme décrit ci-dessus) produira un élément HOARenderingItem à une entrée dans les éléments track_specs et importances, ainsi qu'un élément MetadataSource contenant uniquement l'objet HOATypeMetadata susmentionné.

5.3 Traitement des caractéristiques de restitution

Certaines fonctionnalités du système de restitution sont exécutées en modifiant la liste des caractéristiques de restitution sélectionnées. Le paragraphe 5.3.1 décrit comment retirer des contenus en fonction du niveau d'importance spécifié, et le paragraphe 5.3.3 explique comment émuler les effets de la conversion des métadonnées descendantes.

5.3.1 Émulation du niveau d'importance

Les paramètres relatifs au niveau d'*importance* définis dans la Recommandation UIT-R BS.2076-1 permettent à un système de restitution d'ignorer des caractéristiques en dessous d'un certain niveau d'importance pour des raisons encore inconnues propres à l'application.

Le modèle ADM spécifie trois paramètres différents à utiliser pour indiquer le niveau d'*importance*:

– attribut d'*importance* d'un élément audioObject
– attribut d'*importance* d'un élément audioPackFormat
– attribut d'*importance* d'un élément audioBlockFormat (typeDefinition==Object)

La principale différence entre ces attributs d'*importance* tient au fait que l'importance de l'élément *audioBlockFormat* est variable selon le temps, tandis que l'importance des éléments *audioObject* et *audioPackFormat* est statique.

Un seuil différent peut être utilisé pour chaque attribut d'*importance*. La détermination des valeurs de seuil souhaitées étant considérée comme très spécifique aux cas d'application et d'utilisation, elle n'entre pas dans le champ d'application d'une spécification portant sur les systèmes de restitution de production. En revanche, le système de restitution permet d'émuler les effets de l'application d'un seuil d'importance donné au modèle ADM. De cette façon, les producteurs de contenu peuvent étudier les effets liés à l'application des valeurs d'*importance* sur le système de restitution. Par conséquent, l'émulation du niveau d'importance ne fait pas partie du processus de restitution proprement dit, mais elle est appliquée sous la forme d'une étape de post-traitement des éléments `RenderingItems`.

5.3.1.1 Valeurs d'importance des éléments `RenderingItems`

Chaque caractéristique de restitution peut avoir son propre ensemble de valeurs d'*importance* effectives, car les éléments *audioObjects* et *audioPackFormats* peuvent être imbriqués. Ainsi, pour chaque élément `RenderingItem`, tous les éléments de référencement *audioObjects* et *audioPackFormats* intervenant dans la détermination de l'élément `RenderingItem` sont pris en compte.

Les règles suivantes sont appliquées:

– Si un élément *audioObject* possède une valeur d'*importance* inférieure au seuil fixé, tous les éléments *audioObjects* référencés seront également ignorés. Pour cela, la valeur d'*importance* la plus basse de tous les éléments *audioObjects* conduisant à un élément `RenderingItem` sera utilisée pour spécifier le niveau d'*importance* de l'élément *audioObject* relatif à cet élément `RenderingItem`.
– Si un élément *audioPackFormat* possède une valeur d'*importance* inférieure au seuil fixé, tous les éléments *audioPackFormats* référencés seront également ignorés. Pour cela, la valeur d'*importance* la plus basse de tous les éléments *audioPackFormats* conduisant à un élément `RenderingItem` sera utilisée pour spécifier le niveau d'*importance* de l'élément *audioPackFormat* relatif à cet élément `RenderingItem`.
– Un élément *audioObject* sans valeur d'*importance* ne sera pas pris en compte pour déterminer le niveau d'*importance* d'un élément `RenderingItem`.

– Un élément *audioPackFormat* sans valeur d'*importance* ne sera pas pris en compte pour déterminer le niveau d'*importance* d'un élément `RenderingItem`.

Cette action est exécutée dans `fileio.utils.RenderingItemHandler`.

5.3.1.2 Gestion de l'importance statique

Pour un élément `RenderingItem` associé à un élément `ImportanceData`, la caractéristique sera supprimée de la liste des caractéristiques à restituer si la valeur d'importance statique (*audioObject*, *audioPackFormat*) est inférieure au seuil correspondant défini par l'utilisateur:

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

Cette action est exécutée dans `core.importance.filter_audioObject_by_importance` et `core.importance.filter_audioPackFormat_by_importance`.

5.3.1.3 Gestion de l'importance variable dans le temps

La gestion du niveau d'importance appliqué à l'élément *audioBlockFormat* (*typeDefinition==Object*) ne peut être effectuée en filtrant les éléments `RenderingItems`, car cette caractéristique peut être inférieure au seuil uniquement pour une durée limitée. Afin d'émuler le rejet des caractéristiques dans ce cas particulier, l'élément `RenderingItem` sera effectivement mis en silence pendant la durée de l'élément *audioBlockFormat*. Dans ce contexte, "mettre en silence un élément *audioBlockFormat*" revient à supposer que l'élément `bf.gain` est égal à zéro pour un élément *audioBlockFormat* `bf`.

Cette action est exécutée dans `core.importance.MetadataSourceImportanceFilter`.

5.3.2 Émulation de conversion

L'émulation de la conversion des métadonnées peut éventuellement être appliquée aux caractéristiques de restitution. La fonctionnalité d'émulation de la conversion peut être désactivée ou réglée pour convertir les métadonnées au format polaire ou cartésien.

Si l'émulation de la conversion des données est désactivée, la fonction appropriée est sélectionnée parmi celles décrites au paragraphe 10, puis appliquée à l'ensemble des éléments *audioBlockFormats* (*typeDefinition==Objects*) dans les caractéristiques de restitution sélectionnées.

6 Composantes partagées du système de restitution

Cette section décrit les composantes qui sont communes aux sous-systèmes de restitution des différents attributs typeDefinitions.

6.1 Contrôleur de panoramique à source ponctuelle de type polaire

Le contrôleur de panoramique à source ponctuelle constitue l'élément central du système de restitution; à partir des informations de configuration des haut-parleurs et d'une direction en trois dimensions, il produit un gain de haut-parleur, lequel, lorsqu'il est appliqué à un signal mono en forme d'onde/numérique et reproduit vers les haut-parleurs, doit donner à l'auditeur l'impression que le son émane de la source souhaitée.

Le contrôleur de panoramique à source ponctuelle est utilisé dans l'ensemble du système de restitution (où il permet de restituer les sources ponctuelles spécifiées par les métadonnées de l'objet), dans le système de restitution de la portée (où il assure le retour pour le système de restitution de l'élément *DirectSpeakers*), et dans le cadre du processus de conception du décodeur HOA.

Ce système de restitution est doté d'un contrôleur de panoramique à source ponctuelle basé sur la formulation VBAP [2]. Plusieurs améliorations ont été apportées pour permettre une utilisation plus adaptée aux environnements de radiodiffusion:

– Outre les triplets de haut-parleurs comme dans une configuration VBAP, le contrôleur de panoramique à source ponctuelle prend également en charge les haut-parleurs disposés en quadrilatères groupés en forme d'atome. Cette solution permet de résoudre les mêmes problèmes que l'utilisation de haut-parleurs virtuels dans d'autres systèmes, mais elle facilite en outre le fonctionnement global du panoramique.

– La triangulation des haut-parleurs est effectuée dans la position nominale des haut-parleurs puis soumise à une distorsion afin de correspondre au positionnement réel des haut-parleurs. Ainsi, le comportement du panoramique est toujours en adéquation avec les adaptations d'une configuration donnée.

– L'utilisation de haut-parleurs virtuels et du sous-mixage permet de modifier la restitution dans certaines situations afin de corriger les effets perceptuels observés et de produire les comportements souhaités dans des configurations peu denses.

– Pour éviter une conception trop complexe en cas de configuration extrêmement réduite des haut-parleurs, le système 0+2+0 est géré comme un cas particulier.

6.1.1 Architecture

Le contrôleur de panoramique à source ponctuelle contient une liste d'objets au niveau de l'interface `RegionHandler`; chaque objet région sera chargé de produire des gains de haut-parleur sur une portée spatiale donnée.

Afin de produire des gains pour une certaine direction, le contrôleur de panoramique à source ponctuelle interrogera successivement chaque région, laquelle renverra un vecteur de gain si elle peut gérer cette direction, ou un résultat nul dans le cas contraire. Le vecteur de gain utilisé est celui de la première région capable de gérer la direction.

Pour être valable, un contrôleur de panoramique à source ponctuelle doit remplir les deux conditions suivantes:

– Au moins une région est capable de gérer une direction donnée.

– Toutes les régions capables de gérer une direction donnée produisent des gains similaires (dans certaines limites de tolérance).

– Les gains produits dans une région sont homogènes par rapport à la direction souhaitée.

Toutes ces propriétés garantissent la bonne définition des gains produits par le contrôleur de panoramique à source ponctuelle pour toutes les directions, ainsi que leur homogénéité par rapport à la direction souhaitée (dans certaines limites de tolérance).

Les sections suivantes décrivent les types d'interface `RegionHandler` disponibles et les processus de configuration utilisés pour générer la liste des régions d'une configuration donnée.

Ce comportement est exécuté dans `core.point_source.PointSourcePanner`.

De plus, une classe `PointSourcePannerDownmix` est exécutée avec la même interface. Lorsque l'interface est interrogée à l'aide d'une position, un autre élément `PointSourcePanner` est invité à obtenir un vecteur de gain auquel sont appliquées une matrice de sous-mixage et une normalisation de puissance. Cette méthode est utilisée au paragraphe 6.1.3.1 pour remettre en correspondance les haut-parleurs virtuels.

6.1.2 Types de région

La plupart des régions produisent des gains pour un sous-ensemble de canaux de sortie; la mise en correspondance de ce sous-ensemble de canaux avec le vecteur intégral des canaux est exécutée dans `core.point_source.RegionHandler.handle_remap`.

6.1.2.1 Triplet

Cette configuration se présente sous la forme d'une région sphérique triangulaire composée de trois haut-parleurs permettant la mise en oeuvre d'un VBAP basique.

La région est initialisée avec les positions en trois dimensions des trois haut-parleurs:

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

Les trois gains de sortie **g** d'une direction donnée D sont tels que:

– $\mathbf{g} \cdot \mathbf{P} = s\mathbf{d}$ pour tout $s > 0$, dans certaines limites de tolérance.

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

– $\| \mathbf{g} \|_2 = 1$

Ce type d'interface `RegionHandler` est exécuté dans `core.point_source.Triplet`.

6.1.2.2 VirtualNgon

Cette configuration se présente sous la forme d'une région composée de n haut-parleurs réels qui se divise en triangles auxquels s'ajoute un haut-parleur virtuel. Chaque triangle est constitué de deux haut-parleurs réels adjacents et d'un haut-parleur virtuel dont le signal est réduit par mixage vers les haut-parleurs réels conformément aux coefficients de sous-mixage fournis.

Par exemple, l'utilisation de quatre haut-parleurs réels en position $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4\}$ et d'un haut-parleur virtuel en position $\mathbf{p}_v$ donnera les triangles suivants:

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

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

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

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

Lorsque ce type d'interface `RegionHandler` est interrogé à l'aide d'une position, chaque triangle doit être testé à tour de rôle jusqu'à ce que l'un d'eux renvoie des gains valides, de la même façon que le contrôleur de panoramique à source ponctuelle de niveau supérieur. Cette action produit un vecteur de n gains pour les haut-parleurs réels $\mathbf{g} = \{g_1, \ldots, g_n\}$ et un gain pour le haut-parleur virtuel g_v, dont le signal est réduit par mixage vers les haut-parleurs réels conformément aux coefficients de sous-mixage $\mathbf{w}_{\mathrm{dmx}}$ fournis:

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

Enfin, la normalisation de la puissance permet d'obtenir des gains finaux:

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

Ce type d'interface `RegionHandler` est exécuté dans `core.point_source.VirtualNgon`.

6.1.2.3 QuadRegion

Cette configuration se présente sous la forme d'une région sphérique quadrilatérale composée de quatre haut-parleurs.

Les gains sont calculés pour chaque haut-parleur en divisant la position en deux composantes, x et y. La valeur x peut être considérée comme la position horizontale dans le quadrilatère, 0 se situant sur le bord gauche de l'affichage et 1 sur le bord droit, et la valeur y comme la position verticale, 0 se situant sur le bord inférieur et 1 sur le bord supérieur.

Les valeurs x et y sont mises en correspondance avec un gain pour chaque haut-parleur à l'aide des équations (1) et (2). Les valeurs x et y (et donc les gains des haut-parleurs correspondants) qui permettent d'obtenir un vecteur de vitesse donné peuvent être déterminées en résolvant les équations (1) à (3).

La solution à ce problème présente un niveau de complexité proche d'une configuration VBAP et produit un gain identique à cette dernière sur les bords du quadrilatère. Ainsi, elle peut être utilisée avec d'autres types d'interface `RegionHandler` dans un seul contrôleur de panoramique à source ponctuelle, conformément aux règles énoncées au paragraphe 6.1.1.

Les gains obtenus sont différenciables à l'infini par rapport à la position dans la région, et produisent des résultats comparables au panoramique par paire dans les situations courantes.

Ce type d'interface `RegionHandler` est exécuté dans `core.point_source.QuadRegion`.

6.1.2.3.1 Formulation

À partir des coordonnées cartésiennes de quatre haut-parleurs, $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4]$ en se déplaçant dans le sens contraire des aiguilles d'une montre depuis le point de vue de l'auditeur, le vecteur de gain $\mathbf{g}$ est calculé comme pour la direction d'une source $\mathbf{d}$ de la manière suivante:

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

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

Où les valeurs x et y sont choisies pour que le vecteur de vitesse $\mathbf{g} \cdot \mathbf{P}$ suive la direction $\mathbf{d}$ souhaitée. L'amplitude du vecteur de vitesse r n'est pas pertinente car une normalisation de la puissance est appliquée aux gains:

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

pour tout $r > 0$.

6.1.2.3.2 Solution

Pour une valeur x, tous les vecteurs de vitesse $\mathbf{d}$ ayant une valeur égale à x sont situés sur un plan formé par l'origine du système de coordonnées et deux points à une certaine distance du haut et du bas du quadrilatère:

$$(1-x)\mathbf{p}_1 + x\mathbf{p}_2$$

$$(1-x)\mathbf{p}_4 + x\mathbf{p}_3$$

Par conséquent:

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

Cette équation peut être résolue pour déterminer la valeur x de la direction d'une source $\mathbf{d}$donnée.

Collecte des termes x:

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

Développement du produit vectoriel et collecte des termes:

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

Enfin, multiplication par **D**:

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

La solution pour x constitue donc la racine d'un polynôme qui peut être résolu à l'aide des méthodes classiques.

En remplaçant **P** par $\mathbf{P}'$ dans les équations ci-dessus, il est également possible de déterminer la valeur y:

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

Les gains **g** peuvent ensuite être calculés à l'aide des équations 1 et 2. Dans la mesure où l'échelle de **d** est ignorée dans l'équation (4), il est possible de trouver des solutions produisant un vecteur de vitesse diamétralement opposé à celui souhaité. Cela peut être vérifié en s'assurant que:

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

6.1.2.4 StereoPanDownmix

Les signaux de sortie d'une source ponctuelle pour la stéréophonie (0+2+0) sont établis par une méthode reposant sur un sous-mixage de 0+5+0 à 0+2+0. Cette méthode est mise en œuvre séparément.

La procédure est la suivante:

– La direction est déplacée au moyen d'un contrôleur de panoramique à source ponctuelle configuré sur 0+5+0 pour produire un vecteur de cinq gains, $\mathbf{g}'$, dans l'ordre M+030, M-030, M+000, M+110, M-110.

– Une matrice de conversion de format de 0+5+0 à 0+2+0 est appliquée pour produire des gains de stéréo $\mathbf{G}''$ dans l'ordre M+030, M-030:

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

– Normalisation de la puissance $\mathbf{g}''$ à une valeur déterminée par l'équilibre entre les haut-parleurs avant et arrière en $\mathbf{g}'$, de sorte que les sources entre M+030 et M-030 ne soient pas atténuées, mais que les sources entre M-110 et M+110 soient atténuées de 3 dB.

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

Ce type d'interface `RegionHandler` est exécuté dans `core.point_source.StereoPanDownmix`.

NOTE – La valeur **g** de (0+5+0) à (0+2+0) correspond intégralement aux coefficients de sous-mixage spécifiés dans la Recommandation UIT-R BS.775 comme suit:

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

6.1.3 Procédure de configuration

La procédure de configuration fait intervenir un contrôleur de panoramique à source ponctuelle contenant les types d'interface `RegionHandler` mentionnés ci-dessus pour une structure donnée. Elle consiste à prendre un objet `Layout` (défini au paragraphe 11.1.3) pour produire un élément `PointSourcePanner`.

Dans un premier temps, le comportement est sélectionné selon l'attribut `Layout::name`. Si l'attribut `Layout::name` est `0+2+0`, la configuration est gérée à l'aide de la fonction de configuration spéciale pour la stéréophonie, décrite au paragraphe 6.1.3.2. Tous les autres cas sont gérés par une fonction générique décrite au paragraphe 6.1.3.1.

La procédure de configuration est gérée dans `core.point_source.configure`.

6.1.3.1 Procédure applicable aux configurations génériques

La procédure utilisée pour paramétrer un élément `PointSourcePanner` dans le cadre de configurations de haut-parleurs génériques est la suivante:

1) Mettre à jour l'azimut des positions nominales des haut-parleurs marqués `M+SC` ou `M-SC` afin de garantir la bonne triangulation des haut-parleurs d'écran très espacés. Si l'azimut réel (`polar_position.azimuth`) est φ, la valeur de l'azimut nominal φ_n (`polar_nominal_position.azimuth`) est:

$$\varphi_n = \mathrm{sgn}(\varphi) \times \begin{cases} 45 & |\varphi| > 30 \\ 15 & \text{otherwise} \end{cases}$$

2) Déterminer l'ensemble de haut-parleurs virtuels remis en correspondance comme décrit ci-dessus. Ces haut-parleurs sont ajoutés à la configuration d'ensemble des haut-parleurs afin d'être traités comme des haut-parleurs réels.

3) Créer deux listes contenant les coordonnées cartésiennes normalisées des haut-parleurs, qui seront utilisées au cours des étapes suivantes; l'une recense les positions nominales des haut-parleurs (afin de trianguler la configuration des haut-parleurs), et l'autre contient les positions réelles des haut-parleurs (à utiliser pour créer les régions). Les positions nominales des haut-parleurs sont celles spécifiées dans la Recommandation UIT-R BS.2051-2, tandis que les positions réelles correspondent à celles qui sont effectivement utilisées par le système de reproduction actuel.

4) À chaque liste s'ajoutent un ou deux haut-parleurs virtuels qui deviendront le haut-parleur virtuel au centre d'un élément `VirtualNgon`:

   - La position $0{,}0, -1$ (en dessous de l'auditeur) doit toujours être ajoutée, car aucune des configurations de haut-parleurs définies dans la Recommandation UIT-R BS.2051-2 ne possède de haut-parleur dans cette position.

- La position 0,0,1 (au-dessus de l'auditeur) doit être ajoutée si la configuration ne possède aucun haut-parleur marqué `T+000` ou `UH+180`. Ce haut-parleur n'est pas utilisé en cas de valeur égale à `UH+180` car lorsqu'il est utilisé dans la configuration `3+7+0` définie dans la Recommandation UIT-R BS.2051-2, cette position peut coïncider avec celle du haut-parleur virtuel, ce qui crée une variation d'échelon dans la fonction de panoramique.

5) Prendre la coque convexe des positions nominales des haut-parleurs. Si cet algorithme est exécuté avec l'arithmétique en virgule flottante, des erreurs peuvent provoquer la fragmentation de certaines facettes de la coque convexe; les facettes sont alors fusionnées dans les limites de tolérance fixées afin que le résultat soit identique à celui qui aurait été obtenu si l'algorithme avait été exécuté avec l'arithmétique exacte.

6) Créer une classe `PointSourcePannerDownmix` avec les régions suivantes:

- Pour chaque facette de la coque convexe ne contenant pas de haut-parleur virtuel ajouté à l'étape 3:
  - S'il s'agit d'une facette à trois arêtes, créer un `Triplet` avec les positions réelles des haut-parleurs correspondant aux sommets de la facette.
  - S'il s'agit d'une facette à quatre arêtes, créer un élément `QuadRegion` avec les positions réelles des haut-parleurs correspondant aux sommets de la facette.
- Pour chaque haut-parleur virtuel ajouté à l'étape 3, créer un élément `VirtualNgon` avec les positions réelles des haut-parleurs adjacents (tous les haut-parleurs partageant une facette de coque convexe avec le haut-parleur virtuel) sur le bord, la position du haut-parleur virtuel au centre, et tous les coefficients de sous-mixage fixés à $\frac{1}{\sqrt{n}}$, où n correspond au nombre de haut-parleurs.

  Il convient de noter qu'aucune des configurations définies dans la Recommandation UIT-R BS.2051-2 ne comporte de facettes à plus de quatre arêtes.

Les coefficients de sous-mixage sont utilisés pour mettre en correspondance les haut-parleurs virtuels avec les haut-parleurs physiques, comme décrit ci-dessous.

Cette action est exécutée dans `core.point_source._configure_full`.

6.1.3.1.1 Détermination des haut-parleurs virtuels avec sous-mixage direct

Pour chaque haut-parleur de couche moyenne, un haut-parleur virtuel est ajouté aux couches supérieure et inférieure au niveau du même azimut que le haut-parleur réel lorsqu'il n'y a pas de haut-parleur réel dans la couche supérieure ou inférieure de cette zone. Le coefficient de sous-mixage de ces haut-parleurs virtuels devra permettre la mise en correspondance directe des données de sortie avec le haut-parleur de couche moyenne correspondant.

Comme les haut-parleurs réels, les haut-parleurs virtuels possèdent une position réelle et une position nominale; la position réelle est établie à partir des positions réelles des haut-parleurs réels, et la position nominale est établie à partir des positions nominales des haut-parleurs réels. L'ajout ou non d'un haut-parleur virtuel dépend des positions nominales des haut-parleurs réels. Ainsi, pour une configuration donnée, le même ensemble de haut-parleurs virtuels est toujours utilisé.

Pour déterminer l'ensemble de haut-parleurs virtuels d'une configuration donnée, on applique la procédure suivante:

– Pour chaque $i \in [1, N]$, où $N = \texttt{len(layouts.channels)}$ correspond au nombre de canaux, on définit:

$$\begin{aligned}
\varphi_{i,r} &= \texttt{layouts.channels[i].polar_position.azimuth} \\
\varphi_{i,n} &= \texttt{layouts.channels[i].polar_nominal_position.azimuth} \\
\theta_{i,r} &= \texttt{layouts.channels[i].polar_position.elevation} \\
\theta_{i,n} &= \texttt{layouts.channels[i].polar_nominal_position.elevation}
\end{aligned}$$

– On définit trois ensembles d'indices de canaux en identifiant les canaux des couches supérieure, moyenne et inférieure de la configuration:

$$\begin{aligned}
S_u &= \{i \mid 30° \le \theta_{i,n} \le 70°\} \\
S_m &= \{i \mid -10° \le \theta_{i,n} \le 10°\} \\
S_l &= \{i \mid -70° \le \theta_{i,n} \le -30°\}
\end{aligned}$$

– Les haut-parleurs virtuels possèdent les mêmes azimuts nominaux et réels que le haut-parleur réel correspondant. L'élévation réelle correspond à l'élévation moyenne des haut-parleurs réels de la couche, le cas échéant, ou à $-30°$ ou $30°$ pour les couches inférieure et supérieure. L'élévation nominale est toujours égale à $-30°$ ou $30°$ pour les couches inférieure ou supérieure.

On définit deux valeurs d'élévation nominale:

$$\theta'_{u,n} = 30°$$

$$\theta'_{l,n} = -30°$$

On définit deux valeurs d'élévation réelle:

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

$$\theta'_{l,r} = \begin{cases} 30° & |S_u| = 0 \\ \frac{\sum_{j \in S_l} \varphi_{j,r}}{|S_l|} & otherwise \end{cases}$$

– On ajoute des haut-parleurs à une couche uniquement si l'azimut nominal absolu du haut-parleur de couche moyenne correspondant est supérieur ou égal à l'azimut nominal absolu maximal des haut-parleurs réels de la couche, plus 40°. Ces limites de l'azimut sont définies comme suit:

$$L_u = \begin{cases} 0 & |S_u| = 0 \\ \max\limits_{j \in S_u} |\varphi_{j,n}| + 40° & otherwise \end{cases}$$

$$L_l = \begin{cases} 0 & |S_l| = 0 \\ \max\limits_{j \in S_l} |\varphi_{j,n}| + 40° & otherwise \end{cases}$$

– Pour chaque j de S_m:

- Créer un haut-parleur supérieur virtuel si $\varphi_{j,n} \ge L_u$, en le définissant par un canal `Channel` struct `channel`, tel que:

$$\begin{aligned}
\texttt{channel.polar_position.azimuth} &= \varphi_{j,r} \\
\texttt{channel.polar_position.elevation} &= \theta'_{u,r} \\
\texttt{channel.polar_nominal_position.azimuth} &= \varphi_{j,n} \\
\texttt{channel.polar_nominal_position.elevation} &= \theta'_{u,n}
\end{aligned}$$

- Créer un haut-parleur inférieur virtuel si $\varphi_{j,n} \geq L_l$, en le définissant par un canal `Channel` struct `channel`, tel que:

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

  Ces deux haut-parleurs ont des coefficients de sous-mixage qui acheminent les gains vers le haut-parleur de la couche moyenne correspondant j.

Cette action est exécutée dans `core.point_source.extra_pos_vertical_nominal`.

6.1.3.2 Procédure pour `0+2+0`

Pour 0+2+0, on obtient un élément `PointSourcePanner` doté d'une seule région `StereoPanDownmix`.

Cette action est exécutée dans `core.point_source._configure_stereo`.

6.2 Méthode pour déterminer si un angle est compris dans un intervalle, avec une marge d'erreur

On utilise la fonction `inside_angle_range` pour comparer des angles à des plages angulaires données permettant de spécifier des plages qui incluent l'arrière du système de coordonnées. Cette procédure est employée pour la zone d'exclusion et les composants *DirectSpeakers*, dans les paragraphes 7.3.12.1 et 8.4.

La signature est la suivante:

```
bool inside_angle_range(float x, float start, float end, float tol=0,0);
```

Elle est vraie pour tout angle x appartenant à l'arc de cercle qui commence à `start` et tourne en sens antihoraire jusqu'à `end`, en étant étendu par `tol`. Tous les angles sont exprimés en degrés.

Dans le cas fréquent où:

$$-180 \leq \texttt{start} \leq \texttt{end} \leq 180$$

Cette fonction équivaut à:

$$\texttt{start} - \texttt{tol} \leq \mathrm{x}' \leq \texttt{end} + \texttt{tol}$$

Où $\mathrm{x}' = \mathrm{x} + 360 \times i$ pour tout i tel que $-180 < \mathrm{x}' \leq 180$.

Dans d'autres cas, le comportement du système est plus subtil. Par exemple, si $\texttt{start} = 90$ et $\texttt{end} = -90$, la partie arrière du système de coordonnées est définie par:

$$\mathrm{x}' \leq -90 \vee \mathrm{x}' \geq 90$$

Le Tableau 2 donne des exemples de plages d'angles, accompagnées des expressions équivalentes.

TABLEAU 2

Expressions équivalentes à `inside_angle_range(x, start, end, tol)`

`start`	`end`	`tol`	**Expression équivalente**
−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

Cette fonction est exécutée dans `core.geom.inside_angle_range`.

6.3 Méthode pour déterminer si un canal est prévu pour les effets basses fréquences (LFE) à partir de ses métadonnées de fréquence

Les métadonnées de fréquence, qui peuvent être présentes sous la forme de sous-éléments *frequency* des éléments *audioChannelFormats*, permettent de vérifier si un canal est un canal à effets basses fréquences.

Pour représenter les métadonnées de fréquence, on emploie la structure de données suivante:

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

La fonction dont la signature est

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

évalue la structure

$$\texttt{frequency.lowPass} \wedge \neg\texttt{frequency.highPass} \wedge (\texttt{frequency.lowPass} \le 200\,\text{Hz})$$

et donne le résultat `True` (vrai) si le canal est considéré comme un canal LFE et le résultat `False` (faux) dans les autres cas.

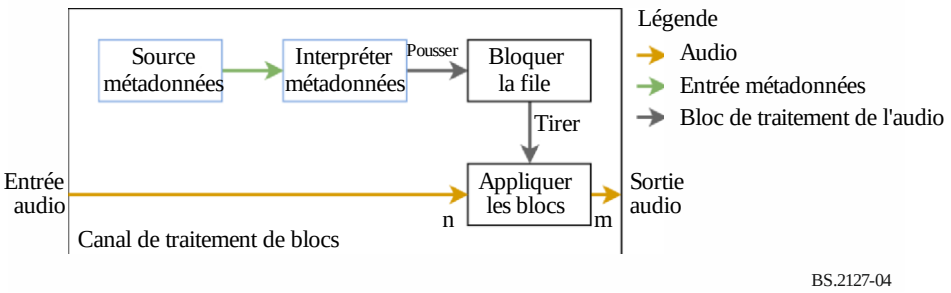
Cette action est exécutée dans `core.renderer_common.is_lfe`.

6.4 Canal de traitement de blocs

La restitution de métadonnées ADM temporisées nécessite une fonctionnalité identique pour toutes les valeurs de l'attribut *typeDefinition*: pour un sous-ensemble donné de canaux d'entrée, un traitement est appliqué entre deux limites de temps afin de produire des canaux de haut-parleurs à la sortie.

FIGURE 4

Structure utilisée pour traiter les canaux concernés. Les composantes représentées en bleu sont d'origine externe



La Figure 4 montre la structure employée pour ce faire. L'interface de cette composante est la suivante:

```
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);
};
```

Le système fournit la source de métadonnées `MetadataSource`. Il s'agit du mécanisme pour fournir les métadonnées au système de restitution. Son interface est la suivante:

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

L'utilisation répétée de la commande `get_next_block` entraîne la réception par le canal de traitement de blocs d'une séquence de blocs `TypeMetadata` (voir la description au paragraphe 5), qui correspondent aux blocs de métadonnées limités dans le temps nécessaires pour la restitution.

Ces blocs de métadonnées sont interprétés à l'aide de la fonction `interpret_metadata`, que le système de restitution propose pour chaque attribut *typeDefinition*. Ces fonctions demandent un élément `TypeMetadata` et renvoient une liste d'objets `ProcessingBlock`, qui contiennent le traitement sonore limité dans le temps nécessaire à la mise en œuvre de l'élément `TypeMetadata` soumis. Le paragraphe 7.2 détaille l'interprétation pour l'attribut *typeDefinition==Objects*. Pour les attributs *typeDefinition==HOA* et *typeDefinition==DirectSpeakers*, un seul élément `ProcessingBlock` est renvoyé.

Voici l'interface externe des objets `ProcessingBlock`:

```
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);
}
```

On considère que les échantillons soumis au traitement `process` sont un sous-ensemble des échantillons dans le fichier entrée/sortie, de telle sorte que `input_samples`[i] et `output_samples`[i] représentent l'ensemble des échantillons à l'entrée et à la sortie `in_out_samples_start` + i. Les attributs `first_sample` et `last_sample` définissent la plage de numéros d'échantillons de l'ensemble s qui seront affectés par le `process`:

$$\texttt{first\_sample} \leq s \leq \texttt{last\_sample}$$

Les moments `start_sample` et `end_sample` désignent les numéros relatifs de début et de fin de la plage d'échantillons, qui permettent de déterminer les attributs `first_sample` et `last_sample` et peuvent servir aux applications de la sous-classe `ProcessingBlock`.

Les objets du canal `BlockProcessingChannel` emmagasinent une file d'objets `ProcessingBlock`, alimentée par des demandes de blocs par l'élément `metadata_source` et leur traitement par la procédure `interpret_metadata`. La fonction `BlockProcessingChannel.process` applique les blocs de traitement présents dans cette file aux échantillons qui lui sont soumis, en utilisant les attributs `first_sample` et `last_sample` pour déterminer quand passer au bloc suivant.

Cette structure permet le découplage des composantes du système de restitution. Ainsi, la taille des échantillons sonores traités ne dépend pas de la taille des blocs de métadonnées, ce qui ne compromet pas le respect de l'échantillon lors du traitement des métadonnées et évite de complexifier les systèmes de restitution par la prise en compte de la chronologie concrète.

La décision de permettre au système de restitution de demander les blocs de métadonnées respecte l'interprétation des métadonnées temporelles par le système. Si, à l'inverse, les métadonnées étaient envoyées au système, la composante chargée de l'envoi devrait savoir quand le bloc suivant est nécessaire, alors que cela dépend des données temporelles qu'il contient.

Cette fonction est exécutée dans `core.renderer_common`.

6.4.1 Types de`ProcessingBlock` mis en œuvre

Voici trois types de blocs de traitement (ProcessingBlock) courants:

L'élément `FixedGains` applique n gains à un seul canal d'entrée et donne les résultats sous la forme de n canaux de sortie;

L'élément `FixedMatrix` applique une matrice de gains NxM à N canaux d'entrée pour former M canaux de sortie;

L'élément `InterpGains` applique n gains interpolés de manière linéaire à un seul canal d'entrée et donne les résultats sous la forme de n canaux de sortie. Deux vecteurs de gain, `gains_start` et `gains_end`, sont donnés. Ils représentent les gains qui doivent être appliqués aux moments `start_sample` et `end_sample`. Le gain $g(i,s)$ appliqué au canal i pour l'échantillon s est donné par:

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

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

6.5 Interprétation générale des métadonnées temporelles

La détermination des moments de début et de fin des blocs est commune aux systèmes de restitution des différents attributs *typeDefinitions*. Pour un bloc `block` d'objet `TypeMetadata`, la procédure suivante est employée:

– Les moments de début et de fin de l'objet qui contient le bloc sont déterminés par les éléments `block.extra_data.object_start` et `block.extra_data.object_duration`. Si la valeur de l'élément `object_start` est "None", l'hypothèse selon laquelle le début de l'objet correspond au moment 0 est alors formulée. Si la valeur de l'élément object_duration est "None", l'hypothèse selon laquelle la durée de l'objet est infinie est alors formulée.

– Les moments de début et de fin du bloc sont déterminés grâce aux attributs `rtime` et `duration`:

  - Si la valeur des attributs `rtime` et `duration` est différente de `None`, alors le début du bloc correspond au début de l'objet plus `rtime`, et la fin du bloc correspond au début du bloc plus `duration`.
  - Si la valeur des attributs `rtime` et `duration` est `None`, alors le bloc s'étend du début de l'objet à la fin de l'objet.
  - Les autres combinaisons des attributs `rtime` et `duration` sont considérées comme des **erreurs**. Si plusieurs objets *audioBlockFormat* sont inclus dans un élément *audioChannelFormat*, les attributs *rtime* et *duration* doivent être fournis. Si l'objet *audioObject* est entièrement couvert par un seul bloc, aucun attribut *rtime* ou *duration* ne doit être donné. Dans le cas contraire, le comportement n'est pas défini.

La cohérence des indications temporelles doit être vérifiée. Les blocs finissant après la fin de l'objet et les chevauchements de blocs au sein d'une séquence ne sont pas autorisés et sont considérés comme une erreur. Le statut d'erreur signifie que les responsables de la mise en œuvre doivent considérer que les données d'entrée sont fausses. Il faut alors réparer le système qui a produit ces données. Dans la mise en œuvre de référence, la gestion des erreurs consiste à arrêter le processus de restitution et à signaler l'erreur à l'utilisateur. D'autres types de mise en œuvre peuvent recourir à des stratégies de gestion des erreurs différentes en fonction de l'environnement de leur application cible.

Cette action est exécutée dans `core.renderer_common.InterpretTimingMetadata`.

6.6 Interprétation des objets `TrackSpec`

L'entrée de données sonores dans le système de restitution passe par un bus multicanal; les données sont lues directement depuis le fichier d'entrée. Les métadonnées d'entrée, sous la forme de l'élément `RenderingItem`, comprennent des objets `TrackSpec`. Il s'agit des instructions pour l'extraction des canaux de ce bus, notamment par l'application du prétraitement par matrice, qui mélange plusieurs canaux.

Le traitement de chaque type d'objet `TrackSpec` est mis en œuvre par `core.track_processor`.

À partir d'un élément `TrackSpec`, il est possible de créer un objet `TrackProcessor`, qui emploie une méthode unique, `process(sample_rate, input_samples)`, consistant à appliquer le traitement spécifié aux échantillons entrés (`input_samples`) pour obtenir un résultat formé d'un seul canal (au taux d'échantillonnage donné).

6.6.1 SilentTrackSpec

Pour *n* échantillons entrés, la fonction `process` appliquée à un objet `SilentTrackSpec` donne *n* échantillons à valeur nulle.

6.6.2 DirectTrackSpec

`La fonction process` pour un élément `track_spec DirectTrackSpec` renvoie les échantillons entrés sur la piste spécifiée dans l'élément `track_spec.track_index` (l'indexation part de zéro).

6.6.3 MixTrackSpec

`La fonction process` pour un élément `track_spec MixTrackSpec` renvoie la somme des résultats de l'exécution de la fonction `process` sur un élément `TrackProcessor` pour chaque sous-piste de l'élément `track_spec.input_tracks`.

6.6.4 MatrixCoefficientTrackSpec

La fonction `process` pour un élément `track_spec MatrixCoefficientTrackSpec` applique le traitement par matrice spécifié dans l'élément `track_spec.coefficient` (qui représente les paramètres d'un seul élément de *coefficient* d'une matrice) à un canal unique spécifié dans l'élément `track_spec.input_track`.

Si l'élément `track_spec.coefficient.gain` a une valeur différente de `None`, les échantillons sont multipliés par le `gain`.

Si l'élément `track_spec.coefficient.delay` a une valeur différente de `None`, les échantillons sont différés de n échantillons et du nombre de millisecondes indiqué par `delay`, arrondi à l'échantillon le plus proche (si deux échantillons sont possibles, on choisit le plus proche de 0):

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

Certains paramètres ne sont pas pris en charge. Si la valeur de `gainVar`, `delayVar`, `phaseVar` ou `phase` est différente de `None`, ou si la valeur de `delay` est négative, une erreur est signalée.

6.7 Angle relatif

`relative_angle`(x, y) permet de trouver un angle équivalent à y et supérieur ou égal à x. Cela permet d'éviter les cas limites lors des activités impliquant des arcs de cercle.

`relative_angle`(x, y) donne $y' = y + 360n$, où n est le plus petit nombre entier tel que $y' \geq x$.

6.8 Transformations des coordonnées

La fonction `cart` permet de transformer des coordonnées polaires en coordonnées cartésiennes, conformément au paragraphe 2.2:

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

où:

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

Les transformations inverses, permettant d'obtenir l'azimut et l'élévation à partir des coordonnées cartésiennes, sont également définies:

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

La fonction `local_coordinate_system` produit une matrice de rotation qui associe {0,1,0} à un azimut et une élévation donnés:

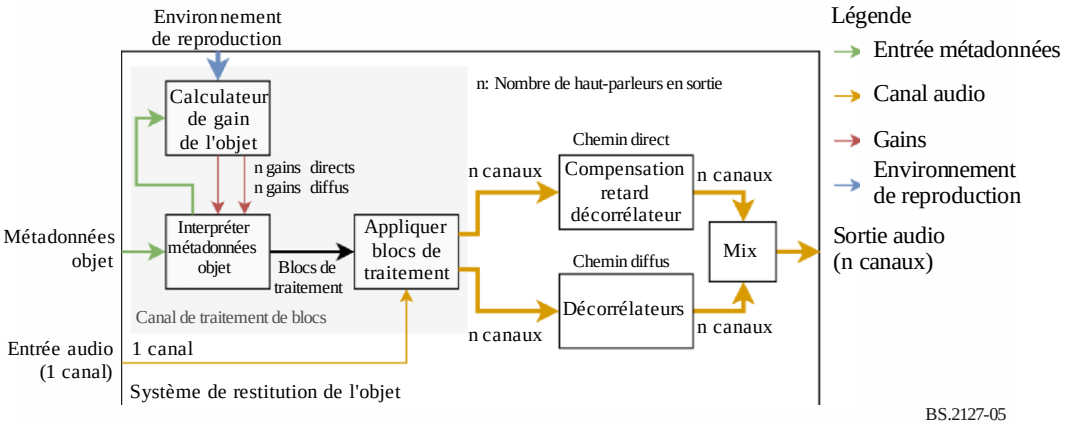
$$\texttt{local\_coordinate\_system}(\varphi,\theta) = \begin{bmatrix} \texttt{cart}(\varphi - 90,0,1) \\ \texttt{cart}(\varphi,\theta,1) \\ \texttt{cart}(\varphi,\theta + 90,1) \end{bmatrix}$$

7 Restitution d'éléments avec l'attribut typeDefinition==Objects

7.1 Structure

FIGURE 5

Structure du système de restitution des objets



La Figure 5 montre la structure du système de restitution de l'attribut *typeDefinition==Objects*. Elle présente la procédure de restitution d'un seul élément ; la restitution de plusieurs éléments se déroule comme s'il y avait une structure pour chaque élément, et les produits sont mélangés.

Les métadonnées sont introduites dans le système de restitution sous la forme d'un objet `ObjectRenderingItem`, comportant un index des pistes, ainsi qu'une source d'objets `ObjectTypeMetadata` qui représentent les paramètres de restitution limités dans le temps pour la piste identifiée.

On applique la méthode décrite dans le paragraphe 7.2 à chaque objet `ObjectTypeMetadata`. Cela permet d'interpréter les métadonnées temporelles et de calculer les vecteurs de gain à l'aide du calculateur de gain présenté dans le paragraphe 7.3. Cette méthode produit des objets `ProcessingBlock`, qui appliquent des opérations de traitement du signal limitées dans le temps aux données sonores d'entrée afin d'obtenir un bus direct et un bus diffus, dont chacun contient un canal par haut-parleur. Cette approche et la classe `BlockProcessingChannel` dans laquelle elle s'inscrit sont décrites dans le paragraphe 6.4.

On passe le bus diffus dans une banque de filtres de décorrélation par canal, et on retarde le bus direct de la même durée, avant de les mélanger pour obtenir la sortie. Le paragraphe 7.4 décrit les délais et les filtres de décorrélation.

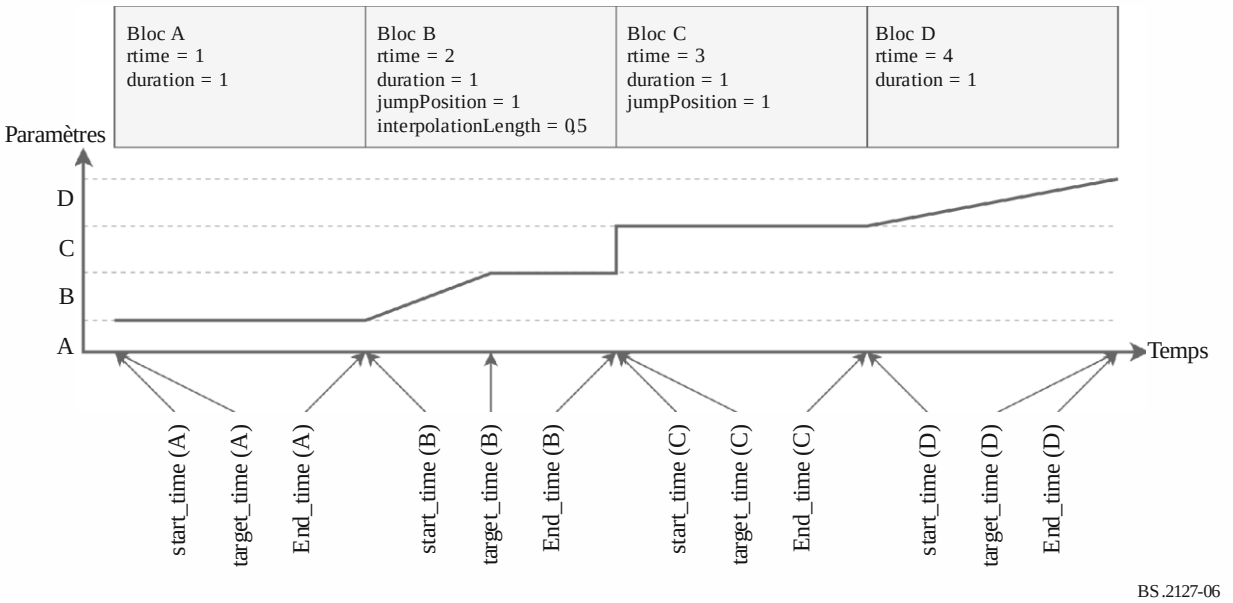
Cette structure est exécutée dans `core.objectbased.renderer.ObjectRenderer`.

7.2 InterpretObjectMetadata

Les métadonnées temporelles relatives aux objets sont interprétées dans la classe `InterpretObjectMetadata`, qui s'intègre dans la structure du canal de traitement des blocs.

FIGURE 6

Exemples de formats audioBlockFormat et courbes d'interpolation associées interprétées



Pour chaque objet `ObjectTypeMetadata` à l'entrée, on suit la procédure ci-dessous:

– Les moments de début `start_time` et de fin `end_time` du bloc sont déterminés conformément au paragraphe 6.5.

– Le moment de fin de l'interpolation dans le bloc `target_time` est déterminé conformément aux cas suivants, illustrés par les blocs correspondants de la Figure 6:

A

S'il s'agit du premier bloc, ou si le moment `end_time` du bloc précédent est inférieur au moment `start_time` du bloc actuel, alors:

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

B

Si l'élément bf.jumpPosition.flag est défini et que la valeur de l'élément bf.jumpPosition.interpolationLength n'est pas "None", alors:

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

C

Si l'élément bf.jumpPosition.flag est défini et que la valeur de l'élément bf.jumpPosition.interpolationLength est "None", alors:

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

D

Si l'élément `bf.jumpPosition.flag` n'est pas défini, alors l'interpolation est réalisée sur l'ensemble du bloc:

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

– Le vecteur de gain `interp_to` est calculé à l'aide d'une instance `GainCalculator` pour le bloc en cours; `interp_from` est le vecteur de gain calculé pour le bloc précédent.

– Si `start_time` < `target_time`, un élément `ProcessingBlock InterpGains` est créé. Il réalise une interpolation d'`interp_from` à `interp_to` entre le moment `start_time` et le moment `target_time`.

– Si `target_time` < `end_time`, un élément ProcessingBlock FixedGains est créé. Il applique le vecteur de gain interp_to entre le moment start_time et le moment target_time.

Cette action est exécutée dans `core.objectbased.renderer.InterpretObjectMetadata`.

7.3 Calculateur de gain

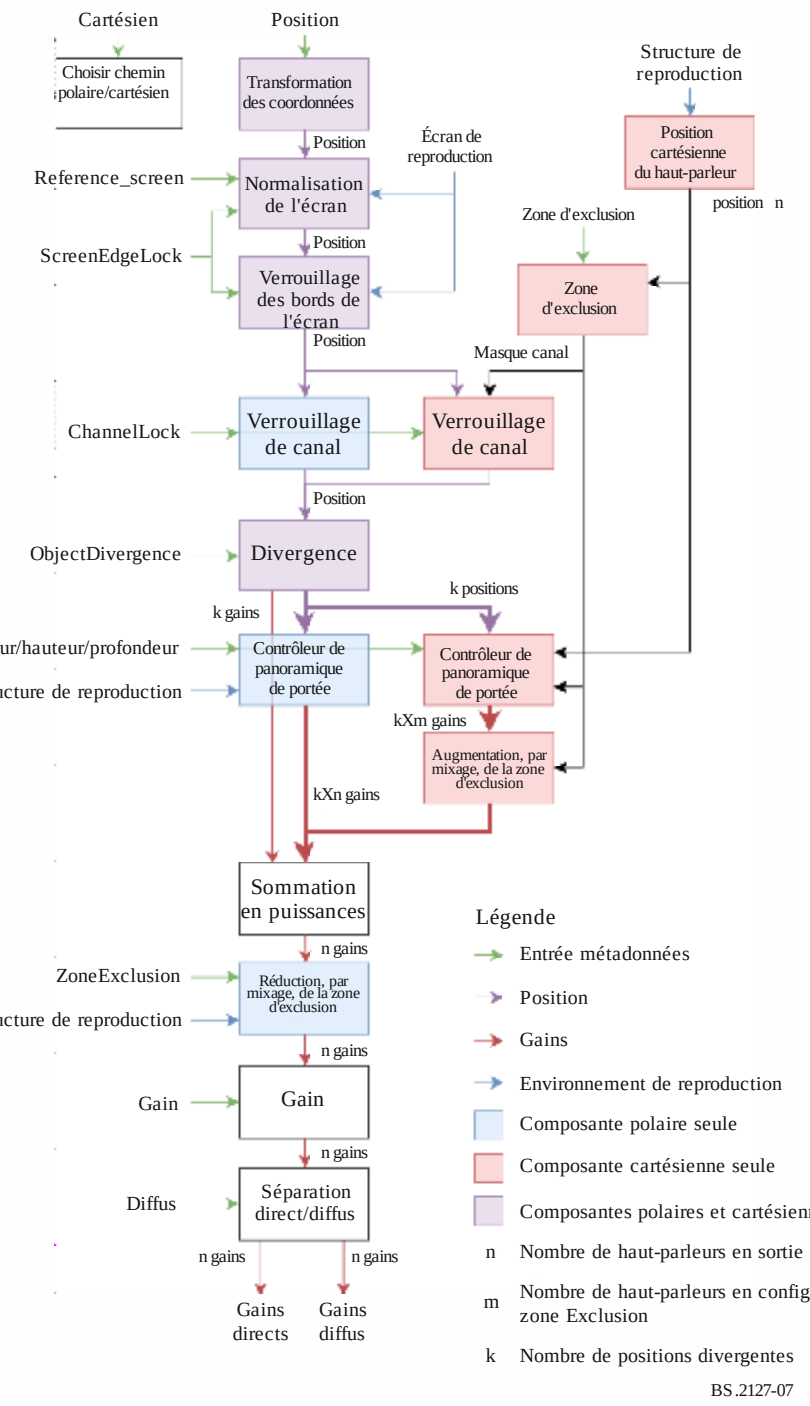
Pour un objet `ObjectTypeMetadata` donné, cet objet calcule un gain pour chaque haut-parleur sur les trajets diffus et direct. L'interface de cette composante est:

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

class GainCalc {
    GainCalc(Layout layout);

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

7.3.1 Structure

FIGURE 7

Structure du calculateur de gain pour le composant *typeDefintion==Objects*



Ce composant est essentiellement constitué à partir des sous-composants évoqués dans cette partie. La Figure 7 présente un diagramme du flux du signal entre ces composants. Le comportement d'un élément `ObjectTypeMetadata (otm)` doté d'un attribut `block_format bf` est le suivant:

– On applique la transformation des coordonnées décrite dans le paragraphe 7.3.2 à l'élément `bf.position` afin d'obtenir l'objet `CartesianPosition` de la position.

– La normalisation de l'écran est réalisée selon la méthode présentée dans le paragraphe 7.3.3, en utilisant les paramètres `position`, `bf.screenRef`, `otm.extra_data.reference_screen` et `bf.cartesian` et en mettant à jour l'élément `position`. Ce composant est initialisé avec l'écran de reproduction (`layout.screen`) et la présentation de reproduction (`layout`).

– Le verrouillage des bords de l'écran est réalisé selon la méthode décrite dans le paragraphe 7.3.4, en utilisant les paramètres `position`, `bf.position.screenEdgeLock` et `bf.cartesian` et en mettant à jour l'élément `position` à l'aide du résultat obtenu. Ce composant est initialisé avec l'écran de reproduction (`layout.screen`) et la présentation de reproduction (`layout`).

– Si `bf.cartesian`, alors:

  • La position allocentrique de chaque haut-parleur dans la présentation `layout.without_lfe` est déterminée conformément au paragraphe 7.3.9, ce qui donne le jeu de positions `allo_channel_positions`.

  • L'algorithme de zone d'exclusion décrit dans le paragraphe 7.3.5 est appliqué aux positions `allo_channel_positions` et à l'élément `bf.zone_exclusion` afin d'obtenir un masque booléen des haut-parleurs à exclure, `excluded`.

  • Le verrouillage des canaux dans la configuration allocentrique présentée dans le paragraphe 7.3.6 est appliqué à l'aide des paramètres `position`, `bf.channelLock` et `excluded`, en mettant à jour l'élément `position`.

  sinon:

  • Le verrouillage des canaux dans la configuration égocentrique présentée dans le paragraphe 7.3.6 est appliqué à l'aide des paramètres `position` et `bf.channelLock`, en mettant à jour l'élément `position`.

– On emploie la méthode décrite dans le paragraphe 7.3.7 pour appliquer la divergence, en utilisant les paramètres `position`, `bf.objectDivergence` et `bf.cartesian`. Cela permet d'obtenir jusqu'à trois sources étendues, dont les gains et les positions sont contenus respectivement dans `diverged_gains` et `diverged_positions`.

– Si `bf.cartesian`, alors:

  • On utilise le contrôleur de panoramique de portée décrit dans le paragraphe 7.3.11 sur chaque `p` des `diverged_positions`, à l'aide des paramètres `channel_positions`, `p`, `bf.width`, `bf.height` et `bf.depth`, qui donnent les vecteurs de gain de l'ensemble des haut-parleurs non exclus. La liste `channel_positions` répertorie des positions de canaux non exclus choisis parmi les `allo_channel_positions[i]`, où `excluded[i]` a la valeur `False`.

  • Ces vecteurs de gain sont mélangés conformément à l'élément `excluded`, ce qui donne la valeur de gain pour chaque haut-parleur `i` où `excluded[i]` est `False` et zéro lorsque `excluded[i]` est `True`. Ces valeurs sont inscrites dans l'élément `gains_for_each_pos`.

  sinon:

  • On utilise le contrôleur de panoramique de portée décrit dans le paragraphe 7.3.8 sur chaque `p` des `diverged_positions`, à l'aide des paramètres `p`, `bf.width`, `bf.height` et `bf.depth`, ce qui donne un vecteur de gain pour chaque haut-parleur. Les vecteurs sont contenus dans l'élément `gains_for_each_pos`.

– Les gains inscrits dans l'élément `gains_for_each_pos` sont mélangés à une puissance qui dépend de `diverged_gains`:

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

– Si `bf.cartesian` n'est pas défini, la zone d'exclusion décrite dans le paragraphe 7.3.12 est appliquée aux `gains` et à l'élément `bf.zoneExclusion`, ce qui donne un nouveau vecteur de `gains`. Ce composant est initialisé avec la présentation `layout.without_lfe`.

– On étend `gains` en y ajoutant les gains des canaux LFE dont la valeur est 0 afin d'obtenir `gains_full`, qui comporte une valeur par haut-parleur inclus dans la présentation `layout`.

– En fonction du paramètre `bf.diffuse`, `gains_full` est divisé en un vecteur direct et un vecteur diffus, qui contrôlent respectivement le trajet direct et le trajet diffus. Ils sont donnés sous la forme de `DirectDiffuseGains`, avec les attributs suivants:

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

7.3.1.1 Discussion (pour information)

La structure du calculateur de gain est influencée par les deux principes suivants:

– Si les paramètres sont peu nombreux (c'est-à-dire si seuls quelques-uns des champs de métadonnées possibles sont utilisés), il est souhaitable de garder l'interprétation évidente de ces paramètres.

– En cas d'utilisation de combinaisons de paramètres, c'est l'option qui ouvre à l'utilisateur le plus grand nombre de possibilités de comportements utiles qui est préférée.

Par exemple:

– Le verrouillage de canal est mis en œuvre en tant que modification de la position. S'il est utilisé par lui-même (avec une valeur de *maxDistance* appropriée), alors la source sera verrouillée à un canal, en raison du comportement du contrôleur de panoramique source ponctuelle. Cependant, il peut être employé avec des paramètres étendus en vue de créer une source étendue centrée sur un haut-parleur particulier, par exemple.

– Le caractère diffus n'est pas lié à l'étendue: une source diffuse complètement étendue peut être obtenue grâce à la mise en place de paramètres étendus appropriés, mais on peut également réaliser un filtrage de décorrélation même si la source n'est pas totalement étendue.

7.3.2 Transformation des coordonnées

La fonction `core.objectbased.gain_calc.coord_trans` réalise une simple transformation des coordonnées afin de convertir les positions entrées en coordonnées dans un repère cartésien uniforme. Sa signature est la suivante:

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

La position `position` est d'abord convertie en un vecteur cartésien **p**.

Si la position `position` est de type `ObjectCartesianPosition`, alors les éléments de **p** sont réduits à l'intervalle $[-1,1]$ avant d'être renvoyés:

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

sinon, **p** est renvoyé sans modification.

`clip` est défini pour les nombres réels comme:

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

et est généralement appliqué à chaque élément d'un vecteur:

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

7.3.3 Normalisation de l'écran

La composante de normalisation de l'écran applique une distorsion à la position des sources afin de compenser les différences en matière de géométrie de l'écran entre l'environnement de production et l'environnement de reproduction. L'interface de cette composante est:

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

Les deux définitions d'écran utilisées sont:

Écran de référence

Il s'agit de la valeur `audioProgrammeReferenceScreen` répertoriée dans l'élément `audioProgramme`, ou, en son absence, de la taille d'écran polaire par défaut. Cette géométrie d'écran est utilisée lors de la production des métadonnées.

Écran de reproduction

Il s'agit de la géométrie de l'écran dans l'environnement de reproduction, où sera écoutée la sortie du système de restitution.

Les positions, dans l'écran de référence, sont soumises à une distorsion afin qu'elles apparaissent aux positions correspondantes de l'écran de reproduction.

7.3.3.1 Représentation de l'écran interne

Les informations relatives aux deux écrans peuvent être fournies sous forme de coordonnées polaires ou cartésiennes (c'est-à-dire d'objets `PolarScreen` ou `CartesianScreen`). Contrairement aux positions des sources d'objets, ces deux types d'écrans ne présentent aucune équivalence évidente. Néanmoins, pour simplifier la mise en œuvre, il est nécessaire de les représenter par une seule représentation d'écran. C'est à cela que sert la structure `PolarEdges`, qui contient l'azimut des bords gauche et droit de l'écran ainsi que l'élévation de ses bords supérieur et inférieur:

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

Un objet `PolarEdges` est créé à partir d'un objet `PolarScreen` ou `CartesianScreen` donné en transformant l'écran en la position centrale d'un repère cartésien et deux vecteurs (dans les directions x et z) qui définissent la surface de l'écran, puis en déterminant l'azimut et l'élévation de chacun des bords de l'écran.

Pour un écran `PolarScreen`, où:

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

La procédure ci-dessous s'applique:

– La position centrale correspond à une simple conversion en coordonnées cartésiennes de la position centrale:

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

– On calcule une largeur et une hauteur cartésiennes:

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

– On utilise le repère `local_coordinate_system` pour déterminer les vecteurs x et z de l'écran:

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

Pour un écran `CartesianScreen`, où:

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

La procédure ci-dessous s'applique:

– La position centrale est directement utilisée:

$$centre = screen.centrePosition$$

– La hauteur et la largeur sont calculées:

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

– Les vecteurs x et z de l'écran sont définis par:

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

Pour les deux types d'écran, un objet `PolarEdges` peut alors être construit avec:

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

7.3.3.2 Compensation de position

Pour certaines présentations de sortie où `cartesian==true`, le panoramage vertical en face de l'auditeur risque de subir une distorsion. Ceci est compensé par l'utilisation de la fonction `core.screen_common.compensate_position`:

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

où:

– φ_r est formé par interpolation linéaire par morceaux θ de:

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

à:

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

– φ' est formé par interpolation linéaire par morceaux φ de:

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

à:

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

7.3.3.3 Déformation de trajectoires

La déformation des positions est définie dans l'élément `core.screen_scale.PolarScreenScaler.scale_az_el`, qui déforme séparément l'azimut et la valeur d'élévation. Selon la `PolarEdges ref` de l'écran de référence et la `PolarEdges rep` de l'écran de reproduction, le processus se déroule comme suit:

– Une interpolation linéaire par morceaux est appliquée à l'azimut, avec mise en correspondance des valeurs

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

jusqu'à

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

– Une interpolation linéaire par morceaux est appliquée à l'élévation, avec mise en correspondance des valeurs

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

jusqu'à

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

Ceci est enveloppé dans l'élément `core.screen_scale.PolarScreenScaler.scale_position`, qui applique la fonction `scale_az_el` aux composantes d'azimut et d'élévation d'un vecteur cartésien, sans modification de distance.

7.3.3.4 Interprétation des métadonnées

Si la fonction `screenRef` est activée et que l'écran de reproduction est fourni, la position passe par l'élément `PolarScreenScaler.scale_direction`, avec les écrans de référence et de reproduction en marche. Dans le cas contraire, la position est renvoyée telle quelle.

Si la fonction `screenRef` n'est pas activée ou si aucun écran de reproduction n'est fourni, la position est renvoyée telle quelle. Dans le cas contraire, son comportement dépend du fanion `cartésien`:

– Si la fonction `cartésienne` est activée, une normalisation et compensation polaires sont appliquées en utilisant la conversion décrite au paragraphe 10.1, ce qui modifie la position $\{x', y', z'\}$:

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

– Dans le cas contraire, la fonction `scale_az_el` est appliquée aux composantes d'azimut et d'élévation de la position.

7.3.4 Verrouillage des bords de l'écran

La composante de verrouillage des bords de l'écran déforme les positions source afin de placer la source sur le bord de l'écran indiqué. Son interface est la suivante:

```
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
  );
};
```

Au démarrage, cette composante transforme l'élément `reproduction_screen` en objet `PolarEdges` de l'élément `polar_edges`, tel qu'indiqué au paragraphe 7.3.3.1.

La fonction `handle_az_el` modifie indépendamment l'azimut et l'élévation, établissant alors de nouvelles valeurs d'azimut et d'élévation:

– Si l'élément `screen_edge_lock.horizontal` est défini par `GAUCHE`, l'azimut est alors défini comme `polar_edges.left_azimuth` ; s'il est défini par `DROITE`, l'azimut est alors défini comme `polar_edges.right_azimuth` ; sinon, l'azimut reste inchangé.

– Si l'élément `screen_edge_lock.vertical` est définit par `HAUT`, alors l'élévation est définie comme `polar_edges.top_elevation` ; s'il est défini par `BAS`, alors l'élévation est définie comme `polar_edges.bottom_elevation` ; sinon, l'élévation reste inchangée.

Si l'élément `reproduction_screen` n'est pas fourni, aucune modification de position n'intervient.

Le traitement a lieu dans le domaine polaire, ce qui explique pourquoi les positions cartésiennes doivent être converties au préalable. La conversion aller-retour est appliquée lorsque la méthode `handle_vector` est utilisée à la place de la méthode `handle_az_el`.

– Si la fonction `cartésienne` est activée, une normalisation et compensation polaires sont appliquées en utilisant la conversion décrite au paragraphe 10.1, ce qui modifie la position $\{x', y', z'\}$:

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

– Dans le cas contraire, la fonction `handle_az_el` est appliquée aux composantes d'azimut et d'élévation de la position.

Cette composante est exécutée dans `core.screen_edge_lock.ScreenEdgeLockHandler`.

7.3.5 Zone d'exclusion cartésienne

L'algorithme de zone d'exclusion cartésienne débute avec la structure de reproduction complète des éléments `channel_positions` et traite les objets de l'élément `ExclusionZone` afin d'identifier quels haut-parleurs doivent être retirés, selon l'algorithme présenté au paragraphe 7.3.12.1. Tout haut-parleur s'avérant se trouver dans une région définie comme `ExclusionZone` est retiré. Si ce retrait entraîne la réduction d'une rangée de haut-parleurs (partageant les mêmes coordonnées y et z) à un seul haut-parleur, tous les haut-parleurs de la rangée sont alors retirés afin d'assurer le maintien des propriétés élémentaires requises par le contrôleur de panoramique à source ponctuelle exposées au paragraphe 7.3.10.

Si le processus d'application d'une zone d'exclusion entraîne le retrait de tous les haut-parleurs, alors aucun haut-parleur n'est retiré.

Enfin, une matrice de conversion montante est créée pour mettre les canaux de la structure réduite en correspondance avec leur canal initial dans la structure complète avec gain d'unité.

Cette action est exécutée dans `core.allocentric.apply_zone_exclusion`.

7.3.6 Verrouillage de canal

Le verrouillage de canal est exécuté comme une modification de position. Si la fonction *channelLock* est activée et qu'un haut-parleur se trouve dans l'intervalle indiquée par l'attribut *maxDistance*, la position sera alors modifiée afin de refléter la position du haut-parleur le plus proche de la position initiale. En l'absence de divergence, d'étendue, de zone d'exclusion et de métadonnées diffuses, la source sera reproduite directement par le haut-parleur sélectionné.

Verrouillage `objectbased._gain_calc.ChannelLockHandlerBase` avec la signature suivante:

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

Le masque d'exclusion de canal `excluded` permet d'indiquer quels haut-parleurs doivent être ignorés. Il est uniquement utilisé en circuit allocentrique, dans la mesure où le verrouillage de canal est alors effectué après la zone d'exclusion.

Pour le circuit égocentrique, `ChannelLockHandlerBase` est configuré dans `core.objectbased._gain_calc.EgoChannelLockHandler`.
Pour le circuit allocentrique, `ChannelLockHandlerBase` est configuré dans `core.objectbased._gain_calc.AlloChannelLockHandler`.

En mode égocentrique, les positions de haut-parleurs sont considérées comme les positions réelles normalisées des haut-parleurs dans la `structure`, tandis qu'en mode allocentrique il s'agit des positions correspondant à l'élément `core.allocentric.positions_for_layout (layout)`, tel que décrit au paragraphe 7.3.9.

Pour appliquer des métadonnées de verrouillage de canal, la procédure à suivre est la suivante:

– Si la valeur `d'exclusion` est différente de `None`, les haut-parleurs ne doivent pas être pris en compte, avec comme valeur `excluded[n] == True` (`n` correspondant au rang du haut-parleur) aux étapes suivantes.

– Si la valeur de `channelLock` est `None`, renvoyer la position `position` initiale.

– Si la valeur de `channelLock.maxDistance` est différente de `None`, calculer la distance ℓ_2 entre chaque position de haut-parleur et la valeur `position`, puis identifier l'ensemble des haut-parleurs (avec un certain degré de tolérance), avec une distance inférieure à la valeur `channelLock.maxDistance` pour le plus de haut-parleurs potentiels.

– Si aucun haut-parleur potentiel n'est identifié, renvoyez la `position`.

– Identifier les haut-parleurs les plus proches de la position `position` parmi les haut-parleurs potentiels. En configuration égocentrique, la distance ℓ_2 entre la position `position` et chaque haut-parleur est utilisée ; en configuration allocentrique, la distance pondérée entre la position `position` et chaque haut-parleur est utilisée. La distance pondérée est calculée comme suit

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

où:

$$\begin{aligned} w_x &= \frac{1}{16} \\ w_y &= 4 \\ w_z &= 32 \end{aligned}$$

– S'il n'existe pas de haut-parleur unique le plus proche (avec un certain degré de tolérance), il convient de choisir le haut-parleur du groupe de haut-parleurs le plus proche avec le plus haut degré de priorité. L'ordre de priorité des haut-parleurs est déterminé par comparaison lexicographique du tuple:

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

Où φ etθ correspondent à l'azimut et à l'élévation réels du haut-parleur. Les tuples inférieurs ont un degré de priorité plus élevé ; les haut-parleurs aux élévations absolues les plus basses ont le degré de priorité le plus élevé, avec les liens brisés par l'élévation, suivis de l'azimut absolu puis de l'azimut.

– La position du haut-parleur choisi est renvoyée.

7.3.7 Divergence

La divergence correspond à l'ajout de deux positions source $\mathbf{p}_l$ et $\mathbf{p}_r$ supplémentaires, à gauche et à droite de la position source $\mathbf{p}_c$ initiale. Les valeurs de gain g_l, g_c et g_r sont associées à chaque position source.

Les métadonnées de divergence sont interprétées dans l'élément `core.objectbased.gain_calc.diverge`, avec la signature suivante:

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

Cette fonction accepte une position en trois dimensions (dans ce cas, le résultat de la fonction de verrouillage de canal) et applique les métadonnées de divergence fournies par l'élément `objectDivergence`. Trois positions source et leurs gains associés sont alors générés et transmis au contrôleur de panoramique de portée pour restitution.

Le calcul de ces gains et positions est décrit ci-dessous.

7.3.7.1 Calcul des gains

Pour chaque valeur `objectDivergence.value` x donnée, les trois gains sont calculés comme suit:

$$g_c = \frac{1-x}{x+1}$$

$$g_l = g_r = \frac{x}{x+1}$$

Cela répond aux exigences suivantes:

– $\forall x,\ g_l + g_r + g_c = 1$

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

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

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

7.3.7.2 Calcul des positions

Les positions générées dépendent du fanion `cartésien` du format de bloc. Un avertissement se déclenche lorsque les fonctions `azimuthRange` et `cartesian` sont activées ou lorsque la fonction `positionRange` est activée et que la fonction `cartesian` ne l'est pas.

7.3.7.2.1 Comportement dans la configuration `cartesian == true`

Pour une valeur `position` de **p** et une valeur `objectDivergence.positionRange` de x, la position centrale est simplement déplacée de gauche à droite par x le long de l'axe x, puis raccourcie en $[-1,1]$:

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

`clip` est défini au paragraphe 7.3.2.

7.3.7.2.2 Comportement dans la configuration `cartesian == false`

Les positions sont calculées pour une valeur `objectDivergence.azimuthRange` donnée a, de telle façon que pour l'auditeur les sources de droite et de gauche sont des degrés de a à gauche et à droite du centre, et que l'ensemble des trois sources sont en ligne droite.

Pour ce faire, il convient de définir trois positions centrées autour de l'axe $+y$ à une distance $d = \| \mathbf{p}_c \|_2$, où $\mathbf{p}_c$ est la position source initiale:

$$'_l = \texttt{cart}(a, 0, d)$$

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

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

Ces positions pivotent ensuite autour de la direction de la source initiale selon la matrice de rotation **M**, qui est définie de telle sorte que $\mathbf{p}_c{}'$ est mis en correspondance avec la position source initiale $\mathbf{p}_c$:

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

7.3.8 Contrôleur panoramique de portée polaire

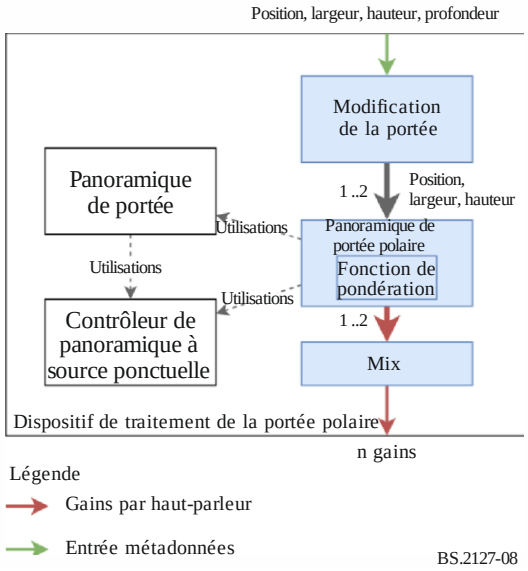
Les paramètres ADM de portée polaire sont gérés dans l'élément `core.objectbased.gain_calc.PolarExtentHandler`, qui utilise les modules décrits ci-dessous afin de produire un vecteur de gain pour la position et les paramètres de portée donnés.

L'interface de cette classe est:

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

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

FIGURE 8

Structure du gestionnaire de portée



La structure de la classe `PolarExtentHandler` est présentée dans la Figure 8.

Cet objet contient une référence vers un élément `PolarExtentPanner`, tel que décrit au paragraphe 7.3.8.2, qu'il utilise pour calculer les vecteurs de gain.

Les paramètres de largeur `width`, de hauteur `height` et de position `position` seront dupliqués et modifiés afin de gérer le paramètre de profondeur `depth` ainsi que la distance entre le composant et la position `position`; ces paramètres sont exécutés dans le contrôleur de panoramique de portée polaire afin de générer pour chacun un vecteur de gain de haut-parleur, puis ces vecteurs de gain sont assemblés. Cette procédure est décrite au paragraphe 7.3.8.2.

Les modes de restitution de portée polaire utilisent le contrôleur de panoramique de répartition pour générer des gains de haut-parleur, tel que décrit ci-dessous.

7.3.8.1 Contrôleur de panoramique de répartition

La forme des sources étendues dans le système de restitution est définie conformément à une fonction de pondération, qui peut calculer une pondération pour une direction en trois dimensions donnée. Cette pondération peut être comprise comme le nombre de fois qu'un objet donné doit être reproduit dans une direction donnée. Par exemple, pour une source plus large que haute, située face à l'auditeur, une fonction de pondération similaire à celle représentée à la Figure 10 peut être utilisée.

En produisant des gains par haut-parleur qui reflètent cette fonction de pondération, en appliquant ces gains au signal mono d'un objet et en appliquant un filtrage de décorrélation aux canaux résultants, il est possible d'obtenir une impression de source étendue ou diffuse avec les paramètres de portée prévus.

La classe de l'élément `SpreadingPanner` est utilisée pour calculer un vecteur de gain pour une fonction de pondération donnée.

L'ensemble de 1 652 positions de sources virtuelles utilisé dans le contrôleur de panoramique de répartition est défini comme suit.

Pour chaque élévation θ entre −90° et 90° compris par pas de 5°, on calcule le nombre de points n devant être répartis régulièrement sur un cercle à l'élévation considérée pour parvenir à une densité approximativement uniforme sur la surface de la sphère unité:

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

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

Puis, pour chaque i dans l'intervalle de 0 à $n - 1$ inclus, on calcule l'azimut φ:

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

On obtient ainsi les coordonnées cartésiennes du point (φ, θ, 1).

Les objets de ce type disposent d'un ensemble de positions de sources virtuelles et d'un vecteur de gain de haut-parleur pour chacune de ces positions.

Au cours du démarrage, le contrôleur de panoramique à source ponctuelle est utilisé pour calculer le vecteur de gain de chaque position.

Pour calculer le vecteur de gain pour une fonction de pondération donnée, la fonction de pondération est appliquée aux positions des sources virtuelles. Le vecteur de gain par source virtuelle subséquent est multiplié par les vecteurs de gain des haut-parleurs pré-calculé afin d'obtenir un vecteur unique de gain avant les haut-parleurs. Il est ensuite normalisé par élévation à la puissance afin d'obtenir le vecteur de gain final.

Cette action est exécutée dans `core.objectbased.extent.SpreadingPanner`.

7.3.8.2 Restitution de la portée polaire

La procédure utilisée pour calculer les gains du haut-parleur pour les paramètres `position`, `width`, `height` et `depth` dans le mode polaire est la suivante:

– Le paramètre `depth` est interprété comme étant deux sources étendues avec la même direction, mais des distances différentes. Les deux distances sont:

$$d_1 = \max\left\{0, \| \texttt{position} \|_2 + \frac{\texttt{depth}}{2}\right\}$$
$$d_2 = \max\left\{0, \| \texttt{position} \|_2 - \frac{\texttt{depth}}{2}\right\}$$

– Pour chaque distance, le contrôleur de panoramique de portée polaire est utilisé pour calculer les vecteurs de gain $\mathbf{g'}_1$ et $\mathbf{g'}_2$ de la position `position`, et les paramètres `width` et `height` modifiés par la fonction de modification de portée polaire, décrite ci-dessous.

– Les vecteurs de gain sont mélangés pour produire le vecteur de gain de sortie $\mathbf{g}$, où $\mathbf{g}_i$ est le gain pour le haut-parleur i:

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

7.3.8.2.1 Fonction de modification de la portée polaire

La fonction de modification de la portée est utilisée pour modifier les paramètres de largeur et de hauteur compte tenu du paramètre de distance:

Ses propriétés sont les suivantes:

– À $\texttt{distance} = 0$, la portée est toujours 360°.

– À $\texttt{distance} = 1$, la portée originale est utilisée.

– À $\texttt{distance} > 1$, la portée est réduite à mesure que la distance augmente.

– Dans la configuration $0 < \texttt{distance} < 1$, les petites portées changent plus fortement à mesure qu'elles se rapprochent de $\texttt{distance} = 0$.

La fonction de modification de la portée pour les éléments `extent` et `distance` se définit comme suit:

– La portée en degrés est mise en correspondance de manière linéaire avec une portée située sur l'axe x, avec une taille minimum de:

$$\texttt{min\_size} = 0{,}2$$
$$\texttt{size} = \texttt{min\_size} + \frac{(1 - \texttt{min\_size}) \times \texttt{extent}}{360°}$$

– Un triangle rectangle est formé, dont le côté adjacent et le côté opposé représentent les distances. L'angle formé est ensuite utilisé pour déterminer une nouvelle portée, qui est calculée pour une distance de 1 et pour l'élément `distance`:

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

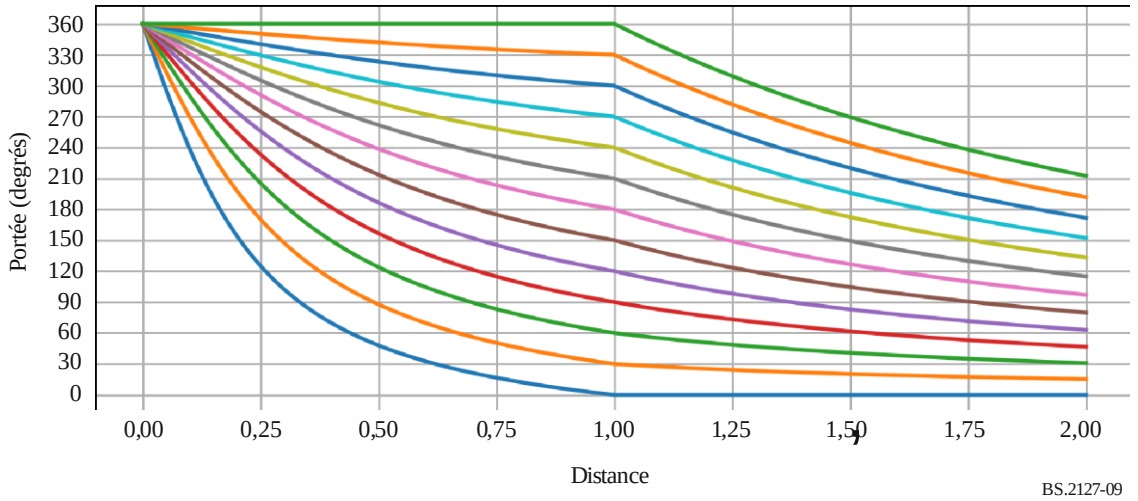
– L'interpolation linéaire par morceaux est appliquée afin de remettre e_d en correspondance avec la portée initiale lorsque $e_d = e_1$:

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

Cette action est exécutée dans `core.objectbased.gain_calc.PolarExtentHandler.extent_mod`. La forme de la fonction de modification de la portée est illustrée à la Figure 9.

FIGURE 9

Fonction de modification pour les sources polaires étendues



NOTE – Chaque ligne illustre comment la portée de sortie varie selon la distance pour une portée de sortie donnée. La portée reste inchangée lorsque `distance` $= 1$. Par exemple, la ligne la plus basse illustre comment la portée modifiée varie selon la distance pour une portée de sortie de 0.

7.3.8.2.2 Contrôleur de panoramique de portée polaire

Afin de gérer l'intervalle complet de positions et de portées autorisées dans le modèle ADM, la taille doit être modifiée avant que la fonction de pondération polaire puisse être appliquée. Les étapes suivantes sont utilisées:

– $max\{width, 5°\}$ et $max\{height, 5°\}$ représentent la largeur et la hauteur modifiées. Ces données sont utilisées avec le contrôleur de panoramique de répartition décrit au paragraphe 7.3.8.1 et la fonction de pondération polaire décrite ci-dessous afin de générer un vecteur de gain de répartition g_s.

– La position est envoyée au contrôleur de panoramique à source ponctuelle afin de générer un vecteur de gain de source ponctuelle g_p.

Les deux vecteurs sont mélangés afin de produire le vecteur g de telle sorte que pour une largeur et une hauteur de zéro, seuls les gains à source ponctuelle sont utilisés, tandis que si la largeur ou la hauteur est supérieure à 5 degrés, seuls les gains de répartition sont utilisés:

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

où:

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

Cette action sert à soutenir les petites portées. Ici, la partie supérieure à zéro de la fonction de répartition doit être suffisamment grande pour couvrir plusieurs points d'échantillonnage afin de produire des gains lisses et pourrait être supérieure au montant désiré.

Cette action est exécutée dans `core.objectbased.extent.PolarExtentPanner.calc_pv_spread`.

7.3.8.2.3 Fonction de pondération polaire

La fonction de pondération pour la restitution de portée polaire est paramétrée par une `position` du vecteur cartésien en trois dimensions, et des angles `width` et `height` en degrés. Puisque le composant de distance de la position n'est pas utilisé, il peut être considéré comme une direction.

La fonction de pondération est la suivante:

– Une matrice de rotation est calculée, ce qui met en correspondance la position {0,1,0} (directement devant l'auditeur) avec la position de la source. Cette matrice de rotation prend la forme d'une rotation autour de {1,0,0}, suivie d'une rotation autour de {0,0,1}. Cette action est exécutée dans `core.objectbased.extent.calc_basis`.

– Si la hauteur est supérieure à la largeur, alors le système de coordonnées est inversé afin de simplifier le calcul, car la fonction de pondération pour une source disposant d'une largeur w et d'une hauteur h doit être similaire à la fonction de pondération pour une source disposant d'une largeur h et d'une hauteur w, tournée de 90° par rapport à la position de la source. Il suffit d'échanger les variables de largeur et de hauteur, et d'échanger les rangs x et z de la matrice de rotation. Voir, par exemple, les Figures 10 et 11, qui ont une forme similaire mais sont tournées de 90 degrés (ne pas prendre en compte la déformation due à la projection utilisée).

– La fonction de pondération approximative est désormais de 1 à l'intérieur d'un rectangle `width` × `height` arrondi au maximum (stade) dans un espace azimut-élévation, avec quelques modifications:

  • Les bords arrondis sont circulaires dans l'espace cartésien, car la pondération est calculée sur la base de l'angle des deux vecteurs en leur centre. Lorsque `width` = `height`, la fonction de pondération est circulaire.

  • À `width` > 180°, la largeur est augmentée de sorte que lorsque la largeur atteint 360° les parties arrondies se chevauchent complètement, ce qui forme une "bande", dans laquelle la fonction de pondération a la même valeur pour toutes les positions à la même élévation. Voir les Figures 12 et 13.

  • Une atténuation progressive est ajoutée aux bords de la fonction de pondération. Cette dernière passe de 1 à 0 lorsque la distance angulaire par rapport à la portée atteint 10 degrés.

Cette fonction est exécutée dans `core.objectbased.extent.PolarExtentPanner.get_weight_func`.

FIGURE 10

Fonction de pondération polaire pour `width` = 90° et `height` = 30°

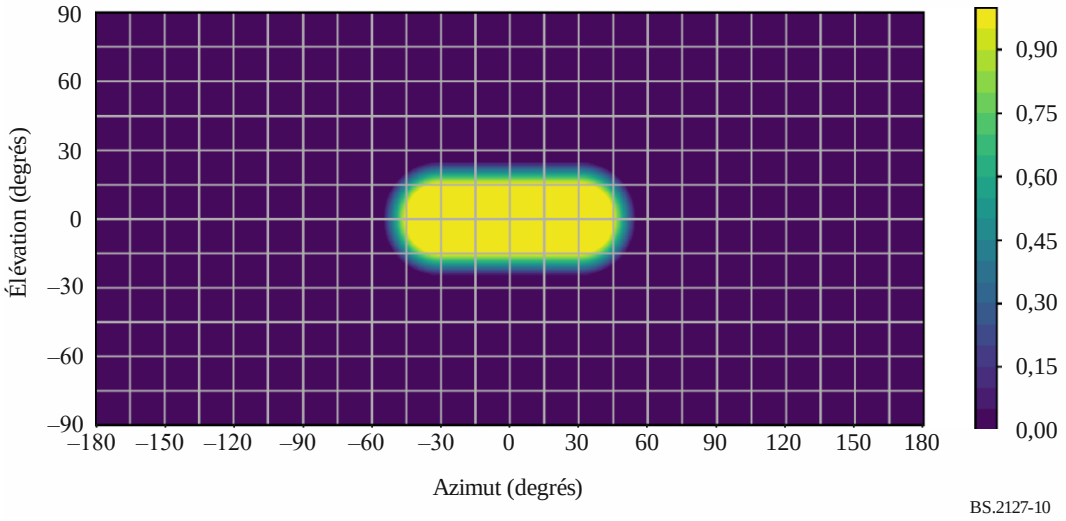


FIGURE 11

Fonction de pondération polaire pour `width` = 30° et `height` = 90°

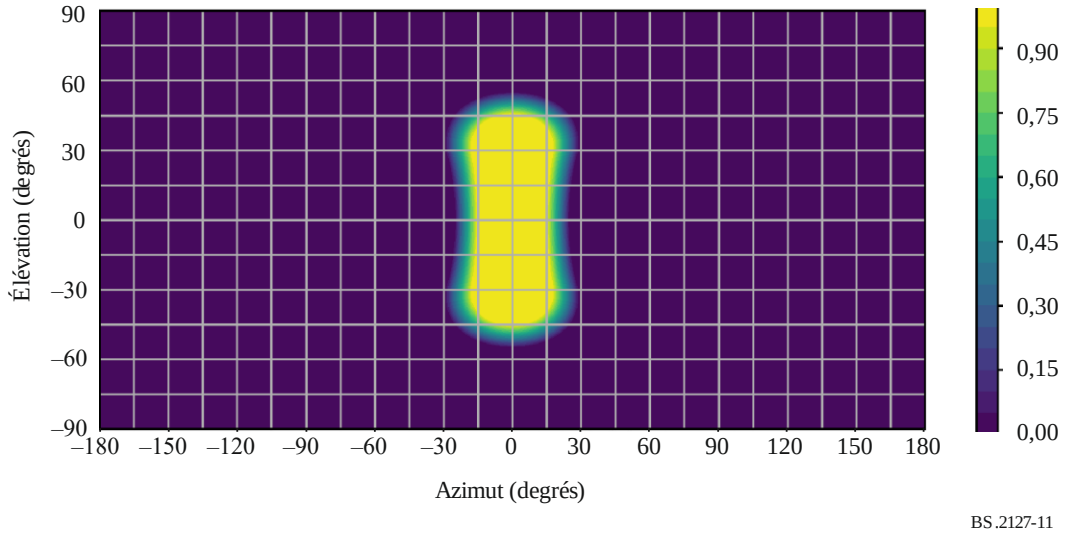


FIGURE 12

Fonction de pondération polaire pour `width` = 300° et `height` = 30°

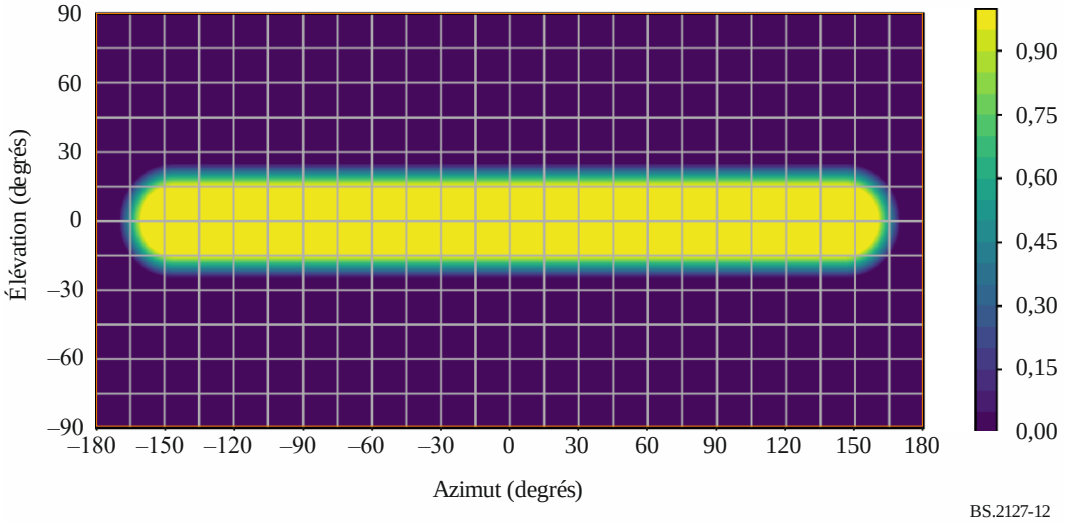
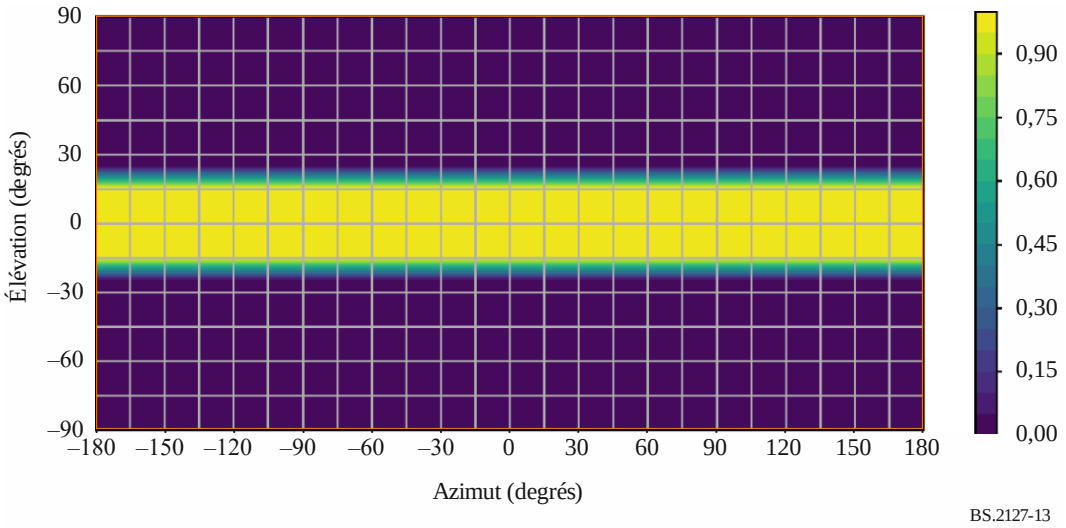


FIGURE 13

Fonction de pondération polaire pour `width` = 360° et `height` = 30°



7.3.9 Positions cartésiennes des haut-parleurs

Pour utiliser le contrôleur de panoramique à source ponctuelle cartésien mentionné au paragraphe 7.3.10, une position cartésienne doit être calculée pour chaque haut-parleur de la configuration.

L'interface de cette composante est la suivante:

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

Tout d'abord, le tableau des positions correspondant à `layout.name` se trouve au paragraphe 11.2.

Pour chaque élément `channel` dans la structure `layout.channels`, les paramètres `x`, `y` et `z` d'une position cartésienne `CatesianPosition` de sortie sont déterminés de la manière suivante:

– Si l'élément `channel.name` est défini par `M+SC` ou `M-SC`, alors:

$$\{x, y, z\} = \texttt{point\_polar\_to\_cart}(\texttt{channel.polar\_position.azimuth}, 0, 1)$$

Veuillez noter que cela suppose une précision infinie dans le `point_polar_to_cart`. En pratique, les positions doivent être modifiées de sorte que:

- $z = 0$.
- Les coordonnées y des deux haut-parleurs de l'écran doivent être identiques.
- Les coordonnées x des deux haut-parleurs de l'écran doivent être parfaitement symétriques par rapport à 0.

– Autrement, les valeurs sont indiquées dans le tableau à la ligne `channel.name`.

Cette action est exécutée dans `core.allocentric`.

7.3.10 Contrôleur de panoramique à source ponctuelle cartésien

L'algorithme de panoramique du point cartésien consiste en un prolongement en trois dimensions du concept de contrôleur de panoramique à "double balance" qui est largement utilisé aux paragraphes 5.1 et 7.1 relatifs à la production de son ambiophonique du canal.

Les données envoyées au contrôleur de panoramique consistent en la position $[p_{ox}, p_{oy}, p_{oz}]$ d'un objet et les positions des haut-parleurs N en sortie, en coordonnées cartésiennes. Les coordonnées $[p_{sx}(j), p_{sy}(j), p_{sz}(j)]$ indiquent la position du haut-parleur j.

En ce qui concerne la configuration des haut-parleurs, le contrôleur de panoramique à source ponctuelle nécessite que les conditions suivantes soient réunies afin de pouvoir placer précisément une image fantôme de l'objet où que ce soit dans la pièce:

– Les haut-parleurs doivent être regroupés en un ou plusieurs plans discrets sur l'axe z.

– Les haut-parleurs de chaque plan doivent être regroupés en un ou plusieurs rangs discrets sur l'axe y-.

– Dans tous les rangs où $-1 < y < 1$ (c.-à-d. tout rang qui ne coupe pas les murs avant ou arrière de la pièce), les haut-parleurs doivent être placés à $x = 1$ et $x = -1$.

– Chaque emplacement des haut-parleurs doit se situer à la surface du cube de la pièce, c'est-à-dire soit au sol, au plafond ou sur les murs.

– Les positions qui réunissent ces conditions peuvent être déterminées en suivant la procédure décrite au paragraphe 7.3.9.

Les gains approximatifs des haut-parleurs pour une position de source donnée sont déterminés:

– En calculant les couches de haut-parleurs au-dessus et en dessous de la source, et en calculant un gain z pour ces deux couches selon la position z des couches et de la source.

– Dans chacune des couches obtenues, en calculant un rang de haut-parleurs devant et derrière la position de la source, et en calculant un gain y pour chacun de ces rangs selon la position y des rangs et de la source.

– Dans chacun des rangs ainsi obtenus, trouvez une paire de haut-parleurs à gauche et à droite de la position de la source, et calculez un gain x pour chacun de ces haut-parleurs selon les positions x des haut-parleurs et de la source.

Un maximum de huit haut-parleurs aura été sélectionné. Pour chacun d'entre eux, le gain est égal à $x \times y \times z$. Les autres haut-parleurs ont un gain de zéro.

Les indications précises sur l'algorithme sont fournies ci-dessous, en calculant le gain $g^{point}(j_x, j_y, j_z)$ pour chaque haut-parleur j. En gardant à l'esprit que chaque axe est indépendant, il est également utile de noter que $g^{point}(x, y, z) = g^{point_x}(x) \times g^{point_y}(y) \times g^{point_z}(z)$, et que les trois gains distincts représentent les valeurs moyenne dans l'algorithme.

```
epsilon = 0.001 //small positive constant

//simplification: Use object-centric coordinates, so that object is
//always at the origin.
for (j = 1 à N)
{
  p_sx(j) -= p_ox
  p_sy(j) -= p_oy
  p_sz(j) -= p_oz
}

for (j = 1 à N)
{
  //Z-gain
  z_this = p_sz(j)
  //find loudspeakers in other plane, on other side of object
  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)
  }

  //Y-gain
  //from among loudspeakers in this plane...
  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)
  //...find loudspeakers in closest row, on other side of object
  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)
  }

  //X-gain
  //from among loudspeakers in this plane...
  p_sx_row = p_sx_plane({i:abs(p_sy_plane(i) - y_this) < epsilon})
  x_this = p_sx(j)
  //find loudspeakers in the closest column
  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
  }
```

Veuillez noter que tout au plus huit haut-parleurs auront des gains supérieurs à zéro, et que la somme des carrés des gains des haut-parleurs sera toujours égale à 1, afin de préserver de l'énergie pendant l'opération de panoramique.

Cette action est exécutée dans `core.point_source.AllocentricPanner`.

7.3.11 Contrôleur de panoramique de portée cartésien

L'objectif du contrôleur de panoramique de portée est de calculer le coefficient de gain de chaque haut-parleur dans la configuration des haut-parleurs en sortie, pour une position d'objet et des portées d'objet données. La portée vise à faire paraître l'objet plus grand de sorte que, lorsque la portée est au maximum, l'objet remplisse la pièce, tandis que, lorsqu'elle est à zéro, l'objet est restitué comme un point.

Pour ce faire, le contrôleur de panoramique de portée crée une grille de plusieurs sources virtuelles dans la pièce. Chaque source virtuelle déclenche les haut-parleurs exactement de la même façon que tout objet restitué avec le contrôleur de panoramique à source ponctuelle. Grâce à la position et aux portées d'un objet, le contrôleur de panoramique de portée détermine quelles sources virtuelles, et combien, seront mises à contribution.

Les étapes suivantes sont nécessaires pour calculer les gains pour un objet avec portée. Chaque étape est expliquée plus en détails dans l'une des sous-sections ci-dessous.

1) Préprogrammer les paramètres de portée.
2) Calculer les gains de points pour toutes les sources virtuelles.
3) Combiner tous les gains des sources virtuelles dans la pièce afin d'obtenir les gains de portée internes.
4) Combiner tous les gains des sources virtuelles aux frontières de la pièce afin d'obtenir les gains de portée aux limites.
5) Combiner les gains de portée interne et aux limites afin d'obtenir les gains de portée finaux.
6) Combiner les gains de portée finaux avec les gains de points de l'objet.

Le contrôleur de panoramique de portée cartésien est exécuté dans `core.objectbased.allo_extent.get_gains`.

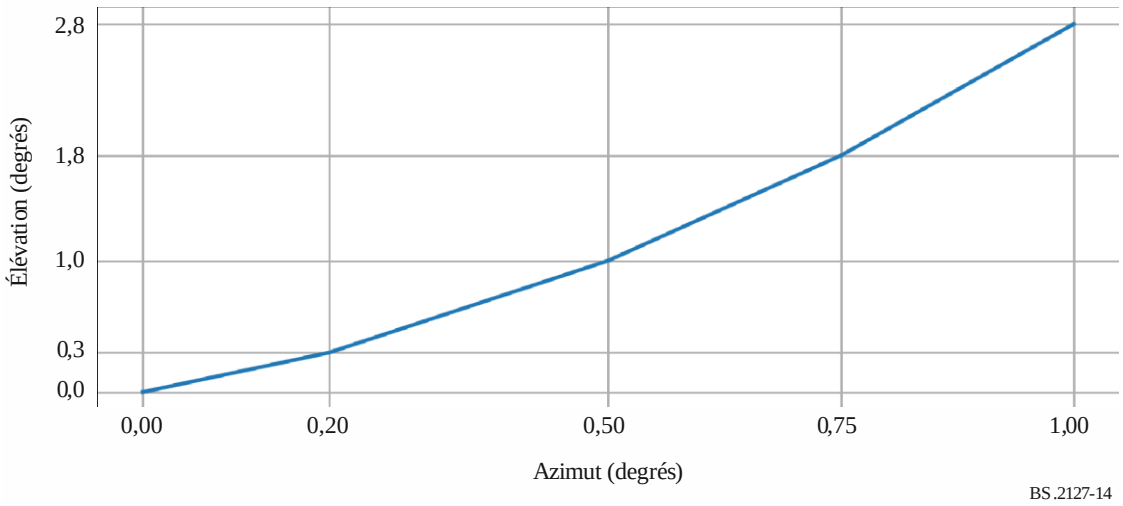
7.3.11.1 Préprogrammation des paramètres de portée

Avant de calculer tout gain, les valeurs des paramètres de portée sont augmentées, de sorte que la fonction de pondération de la source se comporte de façon plus intuitive. L'utilisateur est exposé aux valeurs $s \in [0,1]$, qui sont mises en correspondance dans la portée réelle utilisée par l'algorithme avec l'intervalle $[0,2.8]$. La mise en correspondance est réalisée par une fonction linéaire par morceaux définie par les paires de valeurs $(0,0), (0{,}2, 0{,}3), (0{,}5, 1{,}0), (0{,}75, 1{,}8), (1, 2{,}8)$ et illustrée dans la Figure 14. La valeur maximale de 2,8 garantit que lorsque la portée est fixée à son maximum, elle

occupe réellement toute la pièce. Ci-après, les variables $\hat{s}_x, \hat{s}_y, \hat{s}_z$ correspondent aux valeurs de portée de sortie après mise en correspondance.

FIGURE 14

Mise en correspondance linéaire par morceaux entre les paramètres de la portée de l'ADM et les valeurs de l'algorithme de la portée interne.



Afin de conserver le comportement désiré en cas de valeurs extrêmes de la portée, les valeurs minimales pour $\hat{s}_x, \hat{s}_y, \hat{s}_z$ sont appliquées comme suit:

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

Ces valeurs restreintes s_x, s_y, s_z sont utilisées dans l'intégralité de l'algorithme.

7.3.11.2 Calcul des gains des sources virtuelles

La grille des sources virtuelles est une grille uniforme rectangulaire statique de $N_x \times N_y \times N_z$ points. La grille couvre l'intervalle de positions $[-1,1]$ sur chaque axe. La densité doit être fixée de façon à inclure quelques sources entre les haut-parleurs dans une configuration classique. Des tests empiriques ont montré que $N_x = N_y = N_z = 40$ créait une grille de sources virtuelles adéquate[1]. La notation (x_s, y_s, z_s) sera utilisée pour désigner les éventuelles coordonnées des sources virtuelles. Chaque source virtuelle crée un ensemble de gains $g_j^{point}(x_s, y_s, z_s)$ pour chaque haut-parleur $j = 1, \ldots, N_j$ de la configuration selon l'algorithme du contrôleur de panoramique à source ponctuelle cartésienne décrit au paragraphe 7.3.10. Lorsqu'un haut-parleur a été exclu de la configuration en raison d'un objet Zone d'exclusion (voir le paragraphe 7.3.5), la configuration réduite en haut-parleurs est utilisée lors du calcul des gains.

7.3.11.3 Combinaison des gains des sources virtuelles dans la pièce

La position de l'objet et la portée $(x_o, y_o, z_o, s_x, s_y, s_z)$ sont utilisées pour calculer un ensemble de pondérations qui déterminent dans quelle mesure chaque source virtuelle contribuera aux gains

[1] Pour les configurations des haut-parleurs où il n'y a pas de couche inférieure de haut-parleurs, l'intervalle des sources virtuelles dans l'axe Z est limité à $[0,1]$, et la valeur recommandée de N_z est 20.

finaux[2]. Les pondérations de chaque source virtuelle sont notées $w(x_s, y_s, z_s, x_o, y_o, z_o, s_x, s_y, s_z)$ et sont utilisées pour adapter les gains de points de chaque source virtuelle. Après la pondération, tous les gains de source virtuelle sont additionnés afin d'obtenir les gains de portée interne:

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

Cependant, l'algorithme de portée combine les gains de source virtuelle d'une façon qui varie selon la portée de l'objet. De manière générale, il est décrit comme:

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

L'exposant p indexé sur la portée contrôle la fluidité des gains de tous les haut-parleurs. Il assure la croissance homogène de l'objet à petit s et corrige la distribution de l'énergie dans toutes les directions à grand s. Pour calculer p, il convient, en premier lieu, de classer les coordonnées $\{s_x, s_y, s_z\}$ par ordre décroissant, puis de nommer les trois coordonnées triées: $\{s_1, s_2, s_3\}$ Enfin, les coordonnées sont additionnées afin d'obtenir une portée réelle:

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

Pour les configurations avec un seul plan de haut-parleurs, tel que `0+5+0` ou lorsque la zone d'exclusion réduit la configuration à un seul plan, il convient, en premier lieu, de classer les coordonnées $\{s_x, s_y\}$ par ordre décroissant, puis de nommer les deux coordonnées triées: $\{s_1, s_2\}$, ce qui donne:

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

Pour une configuration `0+2+0` (stéréo) ou lorsque la zone d'exclusion réduit l'ensemble des haut-parleurs à un seul rang, $s_{eff} = s_x$.

La portée réelle est ensuite utilisée pour calculer un exposant défini par morceau:

$$p = \begin{cases} 6 & s_{eff} \le 0.5 \\ 6 - 4 \times \frac{s_{eff} - 0.5}{s_{max} - 0.5} & otherwise \end{cases}$$

où $s_{max} = 2.8$, de sorte que lorsque s est à son maximum, $p = 2$.

La fonction de pondération peut également traiter chaque axe séparément et le calcul de la portée est simplifié si des fonctions de pondération distinctes sont utilisées:

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

Les fonctions choisies ont une forme qui se situe entre cercle et carré (ou sphérique et cubique, en trois dimensions):

---

[2] Pour les configurations de haut-parleurs sans couche inférieure de haut-parleurs, l'algorithme de portée utilise $z_o = max(p_{oz}, 0)$ comme position de l'objet dans l'axe Z. Autrement, $z_o = p_{oz}$. Pour toutes les configurations de haut-parleurs, l'algorithme de portée utilise les mêmes positions X et Y comme contrôleurs de panoramique à source ponctuelle (c.-à-d. $y_o = p_{oy}, x_o = p_{ox}$).

$$w_x(p,o,s) = w_y(p,o,s) \quad = 10^{-\min\left(\left[\frac{3}{2}\left(\frac{p-o}{2s}\right)\right]^4, 6.5\right)}$$

$$w_z(p,o,s) \quad = 10^{-\min\left(\left[\frac{3}{2}\left(\frac{p-o}{s}\right)\right]^4, 6.5\right)} \times \cos\left(s\frac{3\pi}{7}\right)$$

Cela signifie que g_j^{inside} peut être simplifié en

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

où:

$$f_j^x(x_o, s_x) \quad = \sum_{x_s}\left[g_j^{point_x}(x_s) w_x(x_s, x_o, s_x)\right]^p$$

$$f_j^y(y_o, s_y) \quad = \sum_{y_s}\left[g_j^{point_y}(y_s) w_y(y_s, y_o, s_y)\right]^p$$

$$f_j^z(z_o, s_z) \quad = \sum_{z_s}\left[g_j^{point_z}(z_s) w_z(z_s, z_o, s_z)\right]^p$$

Pour les configurations limitées à un seul plan de haut-parleurs, $f_j^z(z_o, s_z) = 1$, et pour un seul rang de haut-parleurs, $f_j^z(z_o, s_z) = f_j^y(y_o, s_y) = 1$.

Par ailleurs, les très petites valeurs de $f_j(c,s)(10^{-6.5})$ sont arrondies à zéro afin d'éviter le soupassement en virgule flottante dans les exécutions.

Une étape de normalisation est appliquée à g_j^{inside}:

$$\tilde{g}_j^{inside} = \begin{cases} \dfrac{g_j^{inside}}{\sqrt{\sum_n\left[g_n^{inside}\right]^2}} & \sqrt{\sum_n\left[g_n^{inside}\right]^2} > tol \\ 0 & otherwise \end{cases}$$

où $tol = 10^{-5}$:

7.3.11.4 Combinaison des gains aux limites

La dernière modification est d'ordre esthétique. Il est important d'avoir un mode où les haut-parleurs opposés ne sont pas activés. Pour ce faire, seules les sources virtuelles situées aux limites sont utilisées. Afin de traiter certaines configurations de haut-parleurs comme des cas spéciaux:

- $dim = 1$ pour les configurations à un seul rang de haut-parleurs après l'application de la zone d'exclusion (p. ex. `0+2+0`),
- $dim = 2$ pour les configurations à un seul plan de haut-parleurs après l'application de la zone d'exclusion (p. ex. `0+5+0`),
- $dim = 4$ pour les configurations avec plus de deux hauteurs de plans de haut-parleurs après l'application de la zone d'exclusion (p. ex. `3+7+0` et `9+10+3`), et
- $dim = 3$ dans tous les autres cas.

Le gain aux limites est alors:

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

où:

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

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

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

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

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

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

7.3.11.5 Combinaison des gains internes et aux limites

Les gains aux limites doivent désormais être combinés aux gains internes. À cette fin, un facteur d'atténuation est introduit pour toutes les sources virtuelles à l'intérieur de la pièce, avec le niveau d'atténuation = "fraction de l'objet hors de la pièce".

Le résultat est:

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

où:

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

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

et $h(c, s)$est une fonction d'atténuation pour un seul axe.

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

Tandis qu'une partie de l'objet étendu commence à se déplacer à l'extérieur de la pièce, toutes les sources virtuelles à l'intérieur de l'objet commencent à s'atténuer, sauf celles près des limites. Lorsqu'un objet atteint une limite, seuls les gains aux limites contribueront aux gains de portée. d_{bound} est la distance minimum des frontières.

Une étape de normalisation est appliquée à g_j^{extent}

$$\tilde{g}_j^{extent} = \begin{cases} \frac{g_j^{extent}}{\sqrt{\sum_n [g_n^{extent}]^2}} & \sqrt{\sum_n [g_n^{extent}]^2} > tol \\ 0 & \text{otherwise} \end{cases}$$

7.3.11.6 Combinaison des gains de portée et de points

Les contributions de la portée sont ensuite combinées aux gains de points, et un fondu enchaîné est appliqué entre les deux, comme fonction de la portée:

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

où:

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

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

et $s_{fade} = 0{,}2$.

Cela assure la fluidité du panoramique et de la croissance de l'objet, assurant ainsi une transition harmonieuse entre la plus petite et la plus grande portée possible.

Enfin, une dernière étape de normalisation est appliquée aux gains:

$$G_j^S = \begin{cases} \frac{g_j^{total}}{\sqrt{\sum_n [g_n^{total}]^2}} & \sqrt{\sum_n [g_n^{total}]^2} > tol \\ 0 & \text{otherwise} \end{cases}$$

7.3.12 Zone d'exclusion polaire

La zone d'exclusion est appliquée en sous-mixant le vecteur de gain du haut-parleur produit plus tôt dans le calculateur de gains afin d'éviter d'envoyer une sortie aux haut-parleurs dans la zone exclue. Elle peut être divisée en deux parties, à savoir: décider quels haut-parleurs se trouvent dans la zone exclue, au paragraphe 7.3.12.1; et calculer le sous-mixage afin de contourner les haut-parleurs exclus, au paragraphe 7.3.12.2.

La sélection des haut-parleurs exclus et le calcul de la matrice de sous-mixage ne prennent en compte que la position nominale des haut-parleurs, de sorte que les changements minimes de position des haut-parleurs n'affectent pas le comportement de la zone d'exclusion.

7.3.12.1 Sélection des haut-parleurs exclus

Les haut-parleurs sont sélectionnés en traitant une liste d'objets `ExclusionZone`, ce qui génère un fanion booléen pour chaque haut-parleur vrai, pour peu que le haut-parleur se trouve dans une zone d'exclusion, et doit, par conséquent, être exclu.

Pour les objets `CartesianZone`, l'expression suivante est utilisée pour déterminer si un haut-parleur se trouve dans la zone, où $\{x, y, z\}$ est la position nominale du haut-parleur, convertie depuis les coordonnées polaires avec un rayon de 1:

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

où $\epsilon = 10^{-6}$ est une marge de sécurité afin de comprendre les erreurs liées à l'arrondissement au moment de la conversion entre les coordonnées polaires et cartésiennes.

Pour les objets `CartesianZone`, l'expression suivante est utilisée afin de déterminer si un haut-parleur se trouve dans la zone, où φ et θ indiquent l'élévation et l'azimut nominaux du haut-parleur.

$$\begin{aligned} & \mathtt{minElevation} - \epsilon < \theta < \mathtt{maxElevation} + \epsilon \\ \wedge\ & \begin{pmatrix} & |\theta| > 90 - \epsilon \\ \vee\ & \mathrm{IAR}(\varphi, \mathtt{minAzimuth}, \mathtt{maxAzimuth}, \epsilon) \end{pmatrix} \end{aligned}$$

IAR est la fonction `inside_angle_range` ; voir le paragraphe 6.2.

L'élévation du haut-parleur doit se trouver dans l'intervalle autorisé, tandis que l'azimut ne doit se trouver que dans l'intervalle autorisé lorsque l'élévation absolue est inférieure à 90 degrés.

Cette action est exécutée dans
`core.objectbased.gain_calc.ZoneExclusionHandler.get_excluded`.

7.3.12.2 Sous-mixage des haut-parleurs exclus

Une fois les haut-parleurs localisés dans la zone, une matrice de sous-mixage est désignée afin de détourner les gains de ces haut-parleurs.

Le contrôleur de panoramique de la zone d'exclusion attribue une liste de groupes de haut-parleurs en sortie à chaque haut-parleur de la configuration. La matrice de sous-mixage détourne les gains d'un haut-parleur vers tous les haut-parleurs non exclus du premier groupe de haut-parleurs disposant de haut-parleurs non exclus. Une description plus détaillée de cette fonctionnalité est fournie dans les deux sections suivantes.

Par exemple, le Tableau 3 montre les groupes pour les haut-parleurs dans 4+5+0. Le premier rang montre que si `M+030` est exclu, la sortie de ce haut-parleur sera détournée vers `M+000`, à moins qu'il ne soit exclu, auquel cas il sera détourné vers `M-030`, etc. jusqu'à `U-110`.

Voici un exemple plus complexe où le regroupement a un certain effet sur `M+000`. S'il est exclu, alors ce canal sera réparti entre les haut-parleurs non exclus dans $\{\mathtt{M+030}, \mathtt{M-030}\}$, à moins que ces deux haut-parleurs ne soient exclus, auquel cas il sera détourné vers les haut-parleurs non exclus dans $\{\mathtt{M+110}, \mathtt{M-110}\}$, etc.

TABLEAU 3

Exemple d'association de haut-parleurs pour 4+5+0

Entrée	Groupes de sortie
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}

Cette fonctionnalité est exécutée dans `core.objectbased.zone.ZoneExclusionDownmix` et `core.objectbased.gain_calc.ZoneExclusionHandler`.

7.3.12.2.1 Détermination des groupes de haut-parleurs

Pendant l'initialisation, les groupes de haut-parleurs en sortie de chaque haut-parleur sont déterminés.

Pour chaque haut-parleur d'entrée, un tuple de nombres réels en virgule flottante, appelé une *clé*, est assigné à chaque haut-parleur en sortie. Les groupes de sortie sont ainsi composés des haut-parleurs en sortie classés par clé et regroupés dans des groupes de clés similaires. Le classement et le regroupement sont par conséquent définis principalement par la fonction de la clé.

La clé pour un haut-parleur en entrée et en sortie comporte quatre clés:

– Une priorité de couche d'un entier relatif, qui est de zéro si les deux haut-parleurs sont sur la même couche, et augmente à mesure que les couches d'entrée et de sortie divergent, privilégiant la sélection d'un haut-parleur d'une couche supérieure avant ceux d'une couche inférieure. Les priorités de couche sont tirées du Tableau 4.

– Une priorité avant/arrière d'un entier relatif, qui est inférieur si les haut-parleurs en entrée et en sortie sont tous les deux à l'avant, sur le côté de ou à l'arrière de l'auditeur. Compte tenu du composant y de la position nominale polaire des haut-parleurs en entrée et en sortie après la conversion en position cartésienne, y_i et y_o, elle est calculée comme suit:

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

– Le vecteur crée une distance entre les positions nominales des deux haut-parleurs, afin de privilégier les petits mouvements.

– La différence absolue dans les coordonnées nominales y entre les deux haut-parleurs, afin de séparer les groupes qui ne sont pas symétriques autour des plans yz ou xz.

TABLEAU 4

Valeur de priorité de couche entre les deux haut-parleurs.

Couche d'entrée	Inférieure	Moyenne	Supérieure	Maximale
Inférieure	0	1	2	3
Moyenne	3	0	1	2
Supérieure	3	2	0	1
Maximale	3	2	1	0

7.3.12.2.2 Application de la zone d'exclusion

La matrice de sous-mixage pour un ensemble de haut-parleurs exclus E est calculée comme suit:

– Pour les haut-parleurs N, commencez par une matrice de sous-mixage $N \times N$, **D**, avec chaque élément initialisé à 0.

– Pour chaque haut-parleur en entrée i, envisager chaque groupe d'indices de candidat de haut-parleur C à la ligne i du tableau de groupe.

- Si tous les haut-parleurs du groupe se trouvent parmi les haut-parleurs ignorés, c'est-à-dire $C \subseteq E$, passer au groupe suivant.
- Autrement, pour chaque j dans $C \setminus E$ (l'ensemble des haut-parleurs dans le groupe qui n'est pas exclu), fixer:

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

et passer au haut-parleur suivant.

Si tous les haut-parleurs sont exclus, **D** est fixé à la matrice identité.

D est ensuite appliqué au vecteur de gain entrant **G** afin de produire $\mathbf{G}'$, en:

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

7.4 Filtres de décorrélation

Lors de la restitution d'objets où le paramètre *diffuse* est supérieur à zéro, le chemin diffuse du système de restitution de l'objet est utilisé, comportant un filtre de décorrélation par sortie de haut-parleur.

Les filtres utilisés sont des filtres $N = 512$ sample long random-phase allpass FIR. Le filtre pour une sortie donnée est généré comme suit:

– Un vecteur pseudo-aléatoire **r** avec des valeurs dans l'intervalle $[0,1)$ de longueur $\frac{N}{2} - 1$ est généré à l'aide du générateur de nombre pseudo-aléatoire MT19937, alimenté par l'indice du nom du canal dans une liste ordonnée de tous les noms de canaux dans la configuration.

– Un vecteur de phase **p** de longueur $\frac{N}{2} + 1$ est défini comme suit:

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

– Le vecteur de fréquence correspondant **x** se définit comme $\mathbf{x}_n = \exp(i\mathbf{p}_n)$.

– Une transformée Fourier inverse à valeur réelle (fonction `irfft`) est tirée des composants de fréquence non négative dans **x** afin d'obtenir le filtre temporel.

Cette action est exécutée dans `core.objectbased.decorrelate.design_decorrelators`.

Le retard introduit par ces filtres est compensé par un retard d'échantillon $\frac{(N-1)}{2}$ dans le chemin direct.

8 Restitution d'éléments de l'attribut typeDefinition==DirectSpeakers

Pour restituer un élément *audioChannelFormats* avec un *typeDefintion==DirectSpeakers*, il est détourné vers un haut-parleur correspondant. Si ce n'est pas possible, le PSP sera utilisé comme solution alternative.

L'algorithme de base est le suivant:

1) Pour les entrées spécifiées à l'aide de définitions communes *audioPackFormats* décrivant les configurations mentionnées dans la Recommandation ITU-R BS.2051-2, les règles de mise en correspondance sont appliquées conformément à l'alinéa 8.1.
2) Déterminez si les métadonnées font référence à un canal LFE (voir l'alinéa 8.2). Si tel est le cas, seules les sorties LFE seront prises en compte, et si ce n'est pas le cas, seules les sorties non-LFE seront prises en compte.
3) Si tout *speakerLabels c*orrespond à un haut-parleur (voir l'alinéa 8.3), le canal est envoyé vers le premier haut-parleur auquel il correspond. Si aucun *speakerLabels* ne correspond, passez à l'étape suivante.
4) Si `screenEdgeLock` est spécifié, la position nominale sera déplacée au bord horizontal ou vertical de l'écran. Les limites minimum et maximum demeurent les mêmes (voir l'alinéa 8.4).
5) Si la position nominale de tout haut-parleur est à l'intérieur des limites de position spécifiées (voir l'alinéa 8.5), dirigez le canal vers le haut-parleur le plus proche de la position nominale spécifiée. Les positions du haut-parleur utilisé sont déterminées par le type de position comme énoncé à l'alinéa 8.5. S'il n'y a aucun haut-parleur à l'intérieur des limites (ou s'il y a plusieurs haut-parleurs dans la position nominale), passez à l'étape suivante.
6) Si les métadonnées font référence à un chemin LFE, dirigez le canal vers `LFE1` (s'il existe) ou ignorez-le. Si les métadonnées font référence à un canal non-LFE, utilisez le PSP correspondant au type de coordonnées utilisé pour définir sa position afin de restituer le canal à sa position nominale.

Les paragraphes suivants décrivent les étapes individuelles plus en détail.

Cette action est exécutée dans `core.direct_speakers.panner.DirectSpeakers Panner`.

8.1 Règles de mise en correspondance

– Si le dernier *audioPackFormat* répertorié dans `type_metadata.audioPackFormats` n'est pas un format du paquet des définitions communes (c.-à-d. qu'il était spécifié dans les métadonnées d'entrée, et non lu dans le fichier des définitions communes), les règles de mise en correspondance ne s'appliquent pas.

– Vérifiez l'identifiant du dernier *audioPackFormat* répertorié dans type_metadata.audioPackFormats dans le Tableau 15 afin de déterminer le `input_layout`. S'il n'est pas répertorié, n'appliquez pas les règles de mise en correspondance.

– Essayez d'appliquer chaque règle du Tableau 16 une par une. Si une règle s'applique, alors les `gains` pour la première règle de correspondance répertoriée sont utilisés afin de reproduire ce canal. Si aucune des règles ne correspond, passez à l'étape suivante. Une règle correspond lorsque toutes ces conditions sont remplies:
  - `rule.speakerLabel` correspond au premier (et seul) *speakerLabel* après l'application de la normalisation décrite à l'alinéa 8.3.
  - `input_layout` (tel que déterminé ci-dessus) est répertorié dans`rule.input_layouts`, s'il est répertorié.
  - Le nom de la configuration du haut-parleur en sortie, `layout.name`, est répertorié dans `rule.output_layouts`, s'il est répertorié.
  - Tous les noms de canaux répertoriés dans `rule.gains` existent dans `layout.channel_names`.

8.2 Détermination LFE

Un canal est considéré comme un canal LFE si la fréquence de l'élément dans l'élément audioChannelFormat a un lowPass de $\leq$ 200 Hz (voir paragraphe 6.3) ou s'il y a un *speakerLabel* qui fait référence à un canal LFE (LFE1 ou LFE2 après l'application du processus de correspondance décrit ci-dessous).

8.3 Correspondance des étiquettes de haut-parleur

La correspondance pour les *speakerLabels* ne fonctionne que pour les étiquettes de la Recommandation ITU-R BS.2051-2 (p. ex. M+030) et les URN utilisés dans le fichier des définitions communes de l'ADM spécifié dans la Recommandation ITU-R BS.2094-1 (p. ex. `urn:itu:bs:2051:0:speaker:M+030`). Les étiquettes des canaux LFE1 et LFE2 sont spécifiées dans la Recommandation ITU-R BS.2051-2. Lorsque les étiquettes *speakerLabels* ci-après sont utilisées dans le fichier ADM, certaines substitutions sont appliquées:

– `LFE → LFE1`

– `LFEL → LFE1`

– `LFER → LFE2`

8.4 Verrouillage des bords de l'écran

L'application de *screenEdgeLock* pour *typeDefintion==DirectSpeakers* réutilise le `ScreenEdgeLockHandler` utilisé pour *typeDefintion==Objects*; décrit en détails au paragraphe 7.3.4. Il permet de transformer la position nominale uniquement; les limites minimum et maximum resteront inchangées.

Cela signifie que si ces limites sont définies, elles sont interprétées comme des limites absolues, peu importe la position de l'écran; la source ne sera verrouillée qu'à un canal situé entre les deux limites. Si ces limites ne sont pas définies, alors le contrôleur de panoramique à source ponctuelle sera activé, et la source se verrouillera sur le côté de l'écran, qu'il y ait un haut-parleur ou non. Il est recommandé de ne pas utiliser *screenEdgeLock* et les limites de coordonnées ensemble.

8.5 Correspondance des limites

Spécifier les limites minimum et maximum permet d'élargir l'intervalle autorisé autour de la position nominale. Si la limite minimum ou maximum n'est pas spécifiée, elle est définie en fonction des coordonnées nominales. Un haut-parleur correspond si toutes les coordonnées se trouvent dans les *limites* spécifiées. Une exception cependant: les haut-parleurs avec des coordonnées polaires aux extrêmes (p. ex., `T+000`) correspondent à tout intervalle azimut, car ils ont un azimut indéterminé.

Un haut-parleur avec une `enceinte` en position polaire nominale correspond aux limites spécifiées en coordonnées polaires si

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

Où IAR est la fonction `inside_angle_range` (voir paragraphe 6.2) et $\epsilon = 10^{-5}$ est une marge de sécurité pour tenir compte des écarts d'arrondis.

Un haut-parleur avec une `enceinte` en position cartésienne qui a été transformée en coordonnées cartésiennes à l'aide du paragraphe 7.3.9 correspond aux limites spécifiées à l'aide des coordonnées cartésiennes si

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

se vérifie.

9 Restitution d'éléments de typeDefinition==HOA

9.1 Formats HOA pris en charge

9.1.1 Rang et degré HOA

Les signaux HOA, définis dans la recommandation UIT-R BS.2076-1, peuvent être restitués jusqu'au rang 50 (voir les détails ci-dessous). Dans le modèle ADM, les canaux HOA sont signalés de manière individuelle par leur *rang* et leur *degré* en fonction des sous-éléments HOA correspondants. Par conséquent, les scènes HOA en trois dimensions (tous les rangs l et degrés m jusqu'à un rang donné L), les scènes HOA 2D (toutes les composantes HOA tel que $|m| = l$ jusqu'à un rang donné L), ainsi que les scènes HOA à rang mixte peuvent être restituées.

Toutefois, si deux signaux HOA partagent le même rang *et* le même degré, une exception est levée et les signaux ne sont pas restitués.

9.1.2 Normalisation

La normalisation des signaux HOA est indiquée via le sous-élément de *normalisation* de type HOA. Les trois types de normalisation (N3D, SN3D et FuMa) sont pris en charge par ce système de restitution. Dans le modèle ADM, la normalisation HOA est spécifiée pour chaque signal HOA de manière individuelle, et il est donc théoriquement possible de définir les scènes HOA en fonction des différents signaux qui utilisent différents types de normalisation. Toutefois, cela n'est pas pris en charge par ce système de restitution; tous les canaux HOA dans un élément *audioBlockFormat* doivent avoir la même normalisation. Enfin, veuillez noter que la normalisation FuMa est prise en charge jusqu'au rang trois seulement.

9.2 Sous-éléments non pris en charge

Les trois sous-éléments suivants de type HOA ne sont actuellement pas interprétés dans la restitution:

– *nfcRefDist*, qui indique la distance de référence de l'installation des haut-parleurs. L'effet de compensation en champ proche (NFC), qui compense les écarts entre la distance de référence des haut-parleurs et la distance à laquelle les haut-parleurs sont placés dans la configuration de lecture, n'est pas appliqué dans ce système de restitution. Appliquer cet effet dans la restitution HOA augmente de manière considérable la complexité des calculs pour le système de restitution, tout en ayant un impact plutôt mineur sur la perception du contenu audio chez l'auditeur.

– *screenRef*, qui indique si la composante HOA est liée à l'écran. L'utilisation attendue de ce sous-élément est ambiguë dans le contexte HOA; il n'est donc pas pris en compte dans la restitution.

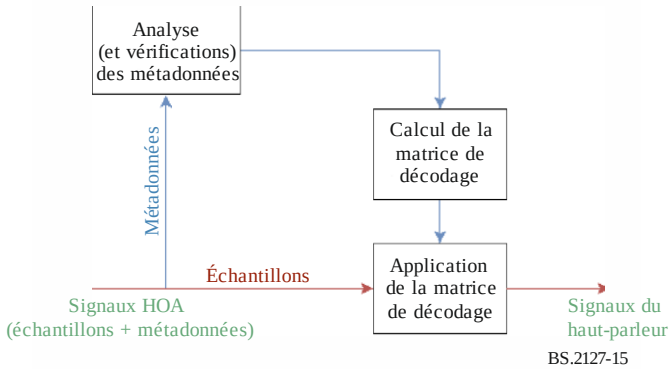
– *equation*, qui est censé remplacer les sous-éléments *rang* et *degré*. La norme ADM actuelle ne fournit pas de règles précises concernant le format utilisé pour définir les formules mathématiques. Par conséquent, ce sous-élément ne peut pas être correctement pris en charge.

Veuillez noter que, comme pour le sous-élément *normalisation*, tous les canaux HOA dans un *audioBlockFormat* doivent partager les mêmes valeurs *nfcRefDist* et *screenRef* pour être restitués.

9.3 Restitution des signaux HOA sur des haut-parleurs

FIGURE 15

Diagramme du flux de restitution HOA



Le processus de restitution des signaux HOA sur des haut-parleurs est présenté à la Figure 15. Premièrement, les métadonnées ADM sont analysées pour identifier le format de l'objet HOA et vérifier si les signaux peuvent être restitués sans ambiguïté. Comme indiqué précédemment, tous les canaux HOA dans un *audioBlockFormat* doivent partager les mêmes valeurs pour les sous-éléments *normalisation*, *nfcRefDist* et *screenRef*. Une matrice de décodage pour le haut-parleur est ensuite calculée et appliquée aux signaux HOA. Ceci est exprimé par l'équation suivante:

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

où:

$\mathbf{S}_{\mathrm{spk}}$ matrice de signaux haut-parleurs, avec les axes $N_{\mathrm{spk}} \times N_{\mathrm{samp}}$

$\mathbf{S}_{\mathrm{HOA}}$ matrice de signaux HOA, avec les axes $N_{\mathrm{HOA}} \times N_{\mathrm{samp}}$

D matrice à valeur réelle, avec les axes $N_{\text{spk}} \times N_{\text{HOA}}$ et à laquelle il est fait référence comme *matrice de décodage HOA*

N_{HOA}, N_{spk} et N_{samp} désignent le nombre de signaux HOA, de signaux haut-parleurs et d'échantillons temporels, respectivement.

Cette section décrit le calcul de la matrice de décodage pour ordonner les canaux avec l'ACN, toutefois l'allocation des canaux en place est spécifiée dans les paramètres *rang* et *degré* dans le *audioBlockFormat*.

La matrice de décodage est appliquée par l'utilisation de la structure de canal de traitement des blocs décrite au paragraphe 6.4. Concrètement, pour chaque objet `HOATypeMetadata` entrant, un bloc de traitement `FixedMatrix` unique est généré, qui applique la matrice de décodage entre les temps déterminés au paragraphe 6.5.

9.3.1 Calcul de la matrice de décodage HOA

Le système de restitution applique la technique de décodage HOA AIIRAD [1]. Cette méthode permet un décodage HOA robuste pour des configurations de haut-parleurs irrégulières, comme celles décrites dans la recommandation UIT-R BS.2051-2. Le calcul de la matrice de décodage se fait dans `core.scenebased.design.HOADecoderDesign`.

Concrètement, la méthode de décodage AIIRAD revient à:

1) Décoder les signaux HOA avec une grille de haut-parleurs virtuels qui sont répartis de manière uniforme sur la sphère, et
2) Contrôle de panoramique des signaux de haut-parleurs virtuels sur les vrais haut-parleurs.

Cette méthode peut être exprimée au moyen de la formule mathématique suivante:

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

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

où $\mathbf{D}'$ désigne la matrice de décodage HOA pour la normalisation *N3D*, $\mathbf{G}$ correspond à la matrice de gain du contrôle de panoramique, $\mathbf{D}_{\text{virt}}$ correspond à la matrice de décodage des haut-parleurs virtuels et ν correspond à un facteur de normalisation énergétique. $\mathbf{D}$ désigne la matrice de décodage complète après application du vecteur de normalisation HOA $\mathbf{n}$ à $\mathbf{D}'$ pour appliquer la normalisation souhaitée.

9.3.1.1 Positions des haut-parleurs virtuels

Afin de faciliter le calcul de la matrice de décodage, les positions angulaires des haut-parleurs virtuels doivent être réparties de manière aussi uniforme que possible sur la sphère. En outre, en règle générale, il doit y avoir deux fois plus de positions de haut-parleurs virtuels que de signaux HOA.

Dans ce système de restitution, les positions des haut-parleurs virtuels représentent un *t-plan sphérique* à 5200 points, ce qui permet de facilement décoder les signaux HOA jusqu'au rang 50.

9.3.1.2 Calcul de la matrice de décodage des haut-parleurs virtuels

Afin de calculer la matrice de décodage, il faut d'abord calculer la matrice pour les coefficients HOA pour les haut-parleurs virtuels, $\mathbf{Y}_{\text{virt}}$. La matrice est obtenue de la manière suivante:

$$\mathbf{Y}_{\text{virt}} = \left[\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{N_{\text{virt}}}\right]$$

$$\mathbf{y}_n = \left[Y_0^0(\theta_n, \phi_n), Y_1^{-1}(\theta_n, \phi_n), \ldots\right]^{\text{T}}$$

où (θ_n, ϕ_n) désigne les angles d'élévation et d'azimut pour le $n$ème haut-parleur virtuel (en utilisant le système des coordonnées HOA et la notation définis dans la Recommandation UIT-R BS.2076-1) et Y_l^m désigne la fonction harmonique sphérique avec le rang-l et le degré-m à valeur réelle avec la

normalisation *N3D*. À noter que la valeur de chaque terme $Y_l^m(\theta, \phi)$ dépend des sous-éléments *rang* et *degré* pour chaque canal HOA.

La matrice de décodage HOA du haut-parleur virtuel est ensuite calculée en transposant $\mathbf{Y}_{\text{virt}}$:

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

Pour choisir la position des haut-parleurs virtuels et la normalisation *N3D*, cela revient à prendre l'inverse de $\mathbf{Y}_{\text{virt}}$.

9.3.1.3 Calcul de la matrice de gain du contrôle de panoramique

Le contrôle de panoramique VBAP est généralement utilisé pour calculer la matrice de gain du contrôle de panoramique dans le cadre de la méthode de décodage HOA AIIRAD. Pour cette forme de restitution, la méthode utilisée pour calculer les gains de contrôle de panoramique est la même que celle qui permet de faire le contrôle de panoramique des objets à source ponctuelle (`core.point_source`).

9.3.1.4 Normalisation énergétique

La matrice de décodage HOA est normalisée; ainsi, si la scène HOA est constituée d'une seule source ponctuelle, la puissance totale des signaux du haut-parleurs est égale à celle du signal source, en moyenne, pour chaque emplacement possible des sources sur la sphère.

Le facteur de normalisation ν est calculé mathématiquement comme suit:

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

Où $\|\cdot\|_{\text{F}}$ désigne la norme de Frobenius.

9.3.1.5 Normalisation HOA

La matrice de décodage est divisée par le vecteur **n** afin de convertir le signal vers la *normalisation N3D* pour laquelle $\mathbf{D}'$ est conçu. **n** est défini pour une `norm` paramètre de normalisation donnée comme:

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

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

10 Conversion des métadonnées

Cette section décrit une méthode pour convertir les paramètres polaires et cartésiens des *audioBlockFormats* avec *typeDefinition==Objects*. La conversion des métadonnées ne peut pas, étant donné sa nature, être exacte; les résultats de la conversion ne correspondront pas exactement aux données non issues d'une conversion. Par conséquent, les résultats de la conversion doivent être suivis. Veuillez noter que le processus de conversion n'est pas réversible, donc il faut éviter de reconvertir des données polaires et cartésiennes.

L'interface de la fonctionnalité de conversion se présente comme suit:

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

Lorsque `to_cartesian` est désigné avec un `AudioBlockFormat input` où `input.cartesian` est défini, `input` est rendu tel quel. À l'inverse, `to_polar` est désigné avec un `AudioBlockFormat input` où `input.cartesian` n'est pas défini, `input` est rendu tel quel.

Autrement, dans tous les cas, `input.cartesian` est inversé, et les changements suivants sont apportés à `input` avant qu'il ne soit rendu:

– `input.position` est converti comme décrit au paragraphe 10.1.
– `input.width`, `input.height` et `input.depth` sont convertis comme décrit au paragraphe 10.2.
– `input.objectDivergence` est converti comme décrit au paragraphe 10.3.

Cette conversion est exécutée dans `core.objectbased.conversion`.

10.1 Conversion de la *position*

Les positions sont converties afin que la position polaire d'un haut-parleurs dans la configuration `4+5+0` corresponde à la coordonnée cartésienne du haut-parleur utilisé dans le contrôleur de panoramique à source ponctuelle, comme décrit au Tableau 8.

Veuillez noter que la même conversion, celle reposant sur la configuration du canal 4+5+0, est utilisée, peu importe la configuration du canal du système de restitution. Cela permet de faire en sorte que les résultats de la conversion soient toujours cohérents, même dans les cas où la configuration de reproduction du système de restitution utilisée n'est pas connue au moment de la conversion. La configuration 4+5+0 a été choisie principalement car elle permet d'assurer une bonne conversion du contenu produit à l'aide de 0+5+0.

Cette section décrit les définitions communes utilisées pour la conversion dans les deux sens; les fonctions de conversion elles-mêmes sont décrites aux paragraphes 10.1.1 et 10.1.2.

`map_linear_to_az` et `map_az_to_linear` définissent une correspondance inversible des positions des sources entre les coordonnées azimuts (φ) et linéaires (x) entre plusieurs haut-parleurs avec des azimuts φ_l et $azimuth_r$, en tenant compte des courbes de contrôle de panoramique à source ponctuelle utilisées pour les coordonnées polaires et cartésiennes.

Par exemple, une position polaire φ_o entre 0° et -30° a une position x donnée par:

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

La correspondance linéaire à azimut est définie comme suit:

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

où:

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

La fonction inverse est définie comme suit:

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

où:

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

La correspondance est appliquée entre les positions de haut-parleurs moyenne, selon les règles suivantes donnant un azimut droit et gauche et une position droite et gauche pour x et y pour un azimut entrant donné:

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

$$find\ \ (\varphi) \begin{cases} \{30,0,\{-1,1\},\{0,1\}\} & IAR(\varphi, 0,45) \\ \{0,-30,\{0,1\},\{1,1\}\} & IAR(\varphi,-45,0) \\ \{-30,-110,\{1,1\},\{1,-1\}\} & IAR(\varphi,-135,-45) \\ \{-110,110,\{1,-1\},\{-1,-1\}\} & IAR(\varphi,135,-135) \\ \{110,30,\{-1,-1\},\{-1,1\}\} & IAR(\varphi,45,135) \end{cases}$$

où IAR est la fonction `inside_angle_range` décrite au paragraphe 6.2.

Les paramètres suivants sont communs aux processus de conversion dans les deux sens:

$$\begin{aligned} \theta_{\texttt{top}} &= 30 \\ \theta'_{\texttt{top}} &= 45 \\ \epsilon &= 1 \times 10^{-10} \end{aligned}$$

10.1.1 Polaire vers cartésien

Pour convertir une coordonnée polaire avec azimut φ, élévation θ et distance d en une coordonnée cartésienne, la fonction

$$\texttt{point\_polar\_to\_cart}(\varphi, \theta, d) = x, y, z$$

est utilisée, où si $|\theta| > \theta_{\texttt{top}}$ alors:

$$\begin{aligned} \theta' &= \theta'_{\texttt{top}} + (90 - \theta'_{\texttt{top}})\frac{|\theta| - \theta_{\texttt{top}}}{90 - \theta_{\texttt{top}}} \\ z &= d\text{sgn}(\theta) \\ r_{xy} &= d\tan\left(\frac{\pi}{180}(90 - \theta')\right) \end{aligned}$$

sinon:

$$\begin{aligned} \theta' &= \theta'_{\mathrm{top}} \frac{\theta}{\theta_{\mathrm{top}}} \\ z &= d\tan\left(\frac{\pi}{180}\theta'\right) \\ r_{xy} &= d \end{aligned}$$

enfin:

$$\begin{aligned} \{\varphi_l, \varphi_r, \{x_l, y_l\}, \{x_r, y_r\}\} &= \mathtt{find\_sector}(\varphi) \\ \varphi' &= \mathtt{relative\_angle}(\varphi_r, \varphi) \\ \varphi'_l &= \mathtt{relative\_angle}(\varphi_r, \varphi_l) \\ p &= \mathtt{map\_az\_to\_linear}(\varphi'_l, \varphi_r, \varphi') \\ x &= r_{xy}(x_l + p(x_r - x_l)) \\ y &= r_{xy}(y_l + p(y_r - y_l)) \end{aligned}$$

`relative_angle` est décrit au paragraphe 6.7.

10.1.2 Cartésien vers polaire

Pour convertir une position cartésienne avec des coordonnées x, y et z vers des coordonnées polaires, la fonction

$$\mathtt{point\_cart\_to\_polar}(x, y, z) = \varphi, \theta, d$$

est utilisée, où si $|x| < \epsilon$ et $|y| < \epsilon$ alors:

$$\{\varphi, \theta, d\} = \begin{cases} \{0,0,0\} & |z| < \epsilon \\ \{0,90\mathrm{sgn}(z), |z|\} & \text{otherwise} \end{cases}$$

sinon, on continue:

$$\begin{aligned} \varphi' &= -\frac{180}{\pi}\mathrm{atan2}(x, y) \\ \{\varphi_l, \varphi_r, \{x_l, y_l\}, \{x_r, y_r\}\} &= \mathtt{find\_cart\_sector}(\varphi') \\ [g_l \quad g_r] &= [x \quad y] \cdot \begin{bmatrix} x_l & y_l \\ x_r & y_r \end{bmatrix}^{-1} \\ r_{xy} &= g_l + g_r \\ \varphi'_l &= \mathtt{relative\_angle}(\varphi_r, \varphi_l) \\ \varphi_{\mathrm{rel}} &= \mathtt{map\_linear\_to\_az}\left(\varphi'_l, \varphi_r, \frac{g_r}{r_{xy}}\right) \\ \varphi &= \mathtt{relative\_angle}(-180, \varphi_{\mathrm{rel}}) \\ \theta' &= \frac{180}{\pi}\arctan\frac{z}{r_{xy}} \end{aligned}$$

Si $|\theta'| > \theta'_{\mathrm{top}}$, alors:

$$\begin{aligned} |\theta| &= \theta_{\mathrm{top}} + (90 - \theta_{\mathrm{top}})\frac{|\theta'| - \theta'_{\mathrm{top}}}{90 - \theta'_{\mathrm{top}}} \\ \theta &= |\theta|\mathrm{sgn}\theta' \\ d &= |z| \end{aligned}$$

sinon:

$$\begin{aligned} \theta &= \theta' \frac{\theta_{\mathrm{top}}}{\theta'_{\mathrm{top}}} \\ d &= r_{xy} \end{aligned}$$

`local_coordinate_system` est défini au paragraphe 6.8.

10.2 Conversion de la portée

La conversion des paramètres liés à la portée se déroule en deux parties:

– $whd2xyz$ et $xyz2whd$: Fonctions qui convertissent les paramètres de portée entre les systèmes cartésien et polaire, en partant du principe que la source se trouve directement en face de l'auditeur, avec un rayon 1.

– `point_polar_to_cart` and `point_cart_to_polar`: fonctions qui assurent la conversion de la position et de la portée. Les positions sont converties à l'aide des méthodes décrites au paragraphe 10.1. La conversion de la portée repose sur $whd2xyz$ et $xyz2whd$, en pivotant la portée cartésienne pour qu'elle corresponde à la position.

Veuillez voter que la conversion de la portée n'est généralement pas réversible.

10.2.1 Polaire vers cartésien

`extent_polar_to_cart` prend une position de source polaire sous la forme d'un azimut, d'une élévation et d'une distance, et une largeur, hauteur et profondeur polaires, et donne des coordonnées cartésiennes x, y et z, et les mesures cartésiennes x, y et z:

$$\texttt{extent\_polar\_to\_cart}(\varphi, \theta, d, \texttt{width}, \texttt{height}, \texttt{depth}) = \{x, y, z, s_x, s_y, s_z\}$$

où:

$$\begin{aligned} \{x, y, z\} &= \texttt{point\_polar\_to\_cart}(\varphi, \theta, d) \\ \{s_{x,f}, s_{y,f}, s_{z,f}\} &= \texttt{whd2xyz}(\texttt{width}, \texttt{height}, \texttt{depth}) \\ [\mathbf{M}_x \quad \mathbf{M}_y \quad \mathbf{M}_z] &= \mathrm{diag}([s_{x,f}, s_{y,f}, s_{z,f}]) \cdot \texttt{local\_coordinate\_system}(\varphi, \theta) \\ s_x &= \| \mathbf{M}_x \|_2 \\ s_y &= \| \mathbf{M}_y \|_2 \\ s_z &= \| \mathbf{M}_z \|_2 \end{aligned}$$

et

$$whd2xyz(width, height, depth) = \{s_{x,w}, max(s_{y,w}, s_{y,h}, s_{y,d}), s_{z,h}\}$$

où:

$$\begin{aligned}
s_{x,w} &= \begin{cases} \sin\frac{\pi}{180}\frac{\texttt{width}}{2} & \texttt{width} < 180 \\ 1 & \text{otherwise} \end{cases} \\
s_{y,w} &= \frac{1-\cos\frac{\pi\ \texttt{width}}{180\ \ 2}}{2} \\
s_{z,h} &= \begin{cases} \sin\frac{\pi}{180}\frac{\texttt{height}}{2} & \texttt{height} < 180 \\ 1 & \text{otherwise} \end{cases} \\
s_{y,h} &= \frac{1-\cos\frac{\pi\ \texttt{height}}{180\ \ 2}}{2} \\
s_{y,d} &= \texttt{depth}
\end{aligned}$$

10.2.2 Cartésien vers polaire

`extent_cart_to_polar` prend une position de source cartésienne sous la forme de coordonnées x, y et z, une portée cartésienne sous la forme de mesures x, y et z, et donne une position et une portée polaires avec un azimut, une élévation et une distance ainsi qu'une largeur, hauteur et profondeur:

$$\texttt{extent\_cart\_to\_polar}(x, y, z, s_x, s_y, s_z) = \{\varphi, \theta, d, \texttt{width}, \texttt{height}, \texttt{depth}\}$$

où:

$$\begin{aligned}
\{\varphi, \theta, d\} &= \texttt{point_cart_to_polar}(x, y, z) \\
[\mathbf{M}_x \quad \mathbf{M}_y \quad \mathbf{M}_z] &= \text{diag}([s_x, s_y, s_z]) \cdot \texttt{local_coordinate_system}(\varphi, \theta)^T \\
s_{x,f} &= \| \mathbf{M}_x \|_2 \\
s_{y,f} &= \| \mathbf{M}_y \|_2 \\
s_{z,f} &= \| \mathbf{M}_z \|_2 \\
\{\texttt{width}, \texttt{height}, \texttt{depth}\} &= \texttt{xyz2whd}(s_{x,f}, s_{y,f}, s_{z,f})
\end{aligned}$$

et

$$xyz2whd(s_x, s_y, s_z) = \{w, h, d\}$$

où:

$$\begin{aligned}
w_{sx} &= 2\frac{180}{\pi}\text{arcsin}s_x \\
w_{sy} &= 2\frac{180}{\pi}\arccos(1 - 2s_y) \\
w &= w_{sx} + s_x\max(w_{sy} - w_{sx}, 0) \\
h_{sz} &= 2\frac{180}{\pi}\text{arcsin}s_z \\
h_{sy} &= 2\frac{180}{\pi}\arccos(1 - 2s_y) \\
h &= h_{sz} + s_z\max(h_{sy} - h_{sz}, 0) \\
\{s_{x,\text{eq}}, s_{y,\text{eq}}, s_{z,\text{eq}}\} &= \text{whd2xyz}(w, h, 0) \\
d &= \max(0, s_y - s_{y,\text{eq}},)
\end{aligned}$$

10.3 Conversion objectDivergence

`azimuthRange` et `positionRange` sont convertis selon la relation suivante:

$$positionRange = tan\frac{270 \times azimuthRange}{\pi}$$

11 Structures et tableaux des données

11.1 Structures de métadonnées internes

11.1.1 Structures partagées

```
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 Input Metadata

```
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 Reproduction Environment Data

```
struct Channel {
  string name;
  /// The real position of the loudspeaker
  PolarPosition polar_position;
  /// The nominal position of the loudspeaker as in bs.2051-2.
```

```
  PolarPosition polar_nominal_position;
  bool is_lfe;
};

struct Layout {
  /// the ITU-format layout name, e.g. "9+10+3"
  string name;
  vector<Channel> channels;
  Screen screen;
};
```

11.2 Positions allocentriques des haut-parleurs

Ces données sont disponibles dans un format lisible à la machine sur `iar/core/data/allo_positions.yaml`, mais sont incluses ici pour référence.

TABLEAU 5

Positions allocentriques des haut-parleurs pour 0+2+0

Canal	X	Y	Z
M+030	−1	1	0
M-030	1	1	0

TABLEAU 6

Positions allocentriques des haut-parleurs pour 0+5+0

Canal	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

TABLEAU 7

Positions allocentriques des haut-parleurs pour 2+5+0

Canal	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

TABLEAU 8

Positions allocentriques des haut-parleurs pour 4+5+0

Canal	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

TABLEAU 9

Positions allocentriques des haut-parleurs pour 4+5+1

Canal	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

TABLEAU 10

Positions allocentriques des haut-parleurs pour 3+7+0

Canal	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

TABLEAU 11

Positions allocentriques des haut-parleurs pour 4+9+0

Canal	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

TABLEAU 12

Positions allocentriques des haut-parleurs pour 9+10+3

Canal	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

TABLEAU 13

Positions allocentriques des haut-parleurs pour 0+7+0

Canal	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

TABLEAU 14

Positions allocentriques des haut-parleurs pour 4+7+0

Canal	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 Correspondance des données DirectSpeakers

Ces données sont disponibles dans un format lisible à la machine sous `core.direct_speakers.panner.itu_packs` et `core.direct_speakers.panner.rules`, mais sont incluses ici pour référence.

TABLEAU 15

Correspondances entre les définitions communes *audioPackFormatID* et le nom de la configuration (voir paragraphe 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

TABLEAU 16

Règles de correspondance pour DirectSpeakers (voir paragraphe 8.1)

Entrée *speakerLabel*	**Gains de reproduction sortie**	`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		

TABLEAU 16 (*suite*)

Entrée *speakerLabel*	**Gains de reproduction sortie**	`input_layouts`	`output_layouts`
M+135	M+110 = 1		
M-135	M-110 = 1		
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	

TABLEAU 16 (*suite*)

Entrée ***speakerLabel***	**Gains de reproduction sortie**	`input_layouts`	`output_layouts`
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 = $\sqrt{\frac{1}{2}}$		
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		

TABLEAU 16 (*suite*)

Entrée ***speakerLabel***	**Gains de reproduction sortie**	`input_layouts`	`output_layouts`
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 = $\sqrt{\frac{1}{2}}$		
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 = $\sqrt{\frac{1}{2}}$		

TABLEAU 16 (*fin*)

Entrée ***speakerLabel***	**Gains de reproduction sortie**	`input_layouts`	`output_layouts`
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 = $\sqrt{\frac{1}{4}}$		
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		

Bibliographie

[1] F. Zotter et M. Frank, "All-round ambisonic panning and decoding". *Journal of the audio engineering society*, vol. 60, n° 10, 2012, p. 807-820.

[2] V. Pulkki, "Virtual sound source positioning using vector base amplitude panning". *Journal of the audio engineering society*, vol. 45, n° 6, 1997, p. 456-466.

Pièce jointe 1 à l'Annexe 1 (informative)

Guide de correspondance des spécifications avec les métadonnées ADM

A1.1 Métadonnées ADM dans le système de restitution ADM UIT-R

L'objectif du tableau ci-dessous est de fournir un résumé des éléments clés du système de restitution et d'indiquer leur emplacement au sein des spécifications données à l'annexe 1. Les spécifications doivent être tirées des références données.

Métadonnées ADM ***sous-élément (attribut)[système de coordonnées)***	**Recommandation UIT-R BS.2076-1**	**Annexe 1 à cette Recommandation**
typeDefinition == "DirectSpeakers"	§ 5.4.3.1 Tableau 11	§ 8
speakerLabel		§ 8.2
position (azimut, élévation, distance, screenEdgeLock)		§ 8
typeDefinition == "Matrix"	§ 5.4.3.2	§ 5.2.6.1.1 § 5.2.6.4
outputChannelIDRef	Tableau 12	§ 5.2.6.1.1
matrice → coefficient (gain, gainVar, phase, phaseVar, delay, delayVar)	Tableau 13	§ 5.6.4
input / outputPackFormatIDRef	§ 5.5.5.1	§ 5.2.6.1.1
encode / decodePackFormatIDRef		§ 5.2.6.1.1
typeDefinition == "Objects"	§ 5.4.3.3	§ 7
position (azimut, élévation, distance, screenEdgeLock) [*polaire*]	Tableau 14	§ 6.1 § 7 § 7.3.4
position (X, Y, Z, screenEdgeLock) [*cartésien*]	Tableau 15	§ 6.1 § 7 § 7.3.10
largeur, hauteur, profondeur [*polaire*]	Tableau 14	§ 7.3.8
largeur, hauteur, profondeur *[cartésien]*	Tableau 15	§ 7.3.11
cartésien	Tableau 16	§ 7.3.1 § 7.3.2
gain		§ 7.3.1

Métadonnées ADM ***sous-élément (attribut)[système de coordonnées)***	**Recommandation UIT-R BS.2076-1**	**Annexe 1 à cette Recommandation**
diffuse		§ 7.3.1 § 7.4
channelLock (maxDistance)		§ 7.3.6
objectDivergence (azimuthRange, positionRange) [*polaire*]		§ 7.3.7 § 7.3.1
objectDivergence (azimuthRange, positionRange) *[cartésien]*		§ 7.3.7 § 7.3.1
jumpPosition (interpolationLength)		§ 7.2
zoneExclusion → zone (minX, maxX, minY, maxY, minZ, maxZ, minElevation, maxElevation, minAzimuth, maxAzimuth)		§ 7.3.5 § 7.3.12
screenRef		§ 7.3.3
importance		§ 5.3.1 § 5.2.7.1.1
typeDefinition == “HOA”	§ 5.4.3.4	§ 9 § 5.2.7.3
équation	Tableau 17	§ 9.2
rang		§ 9.1.1 § 9.3.1.2
degré		§ 9.1.1 § 9.3.1.2
normalisation	§ 5.4.3.4	§ 9.1.2 § 9.3.1.5
nfcRefDist	Tableau 17	§ 9.2
screenRef		§ 9.2
typeDefinition == “Binaural”	§ 5.4.3.5	–

Pièce jointe 2
à l'Annexe 1
(informative)

Configuration alternative des haut-parleurs virtuels

A2.1 Spécifications concernant la configuration alternative des haut-parleurs virtuels

Une configuration alternative VBAP des haut-parleurs virtuels, par rapport à celle spécifiée au paragraphe 6.1.3.1, décrit la position des haut-parleurs virtuels non situés aux pôles et leurs coefficients dérivés. Le traitement des métadonnées ADM est le même que celui décrit dans le texte principal de cette recommandation, et aucune métadonnée supplémentaire n'est requise. Les positions alternatives des haut-parleurs virtuels et de leurs coefficients dérivés reposent sur des optimisations réalisées à l'oreille. Vous trouverez ci-dessous la description de la configuration alternative des haut-parleurs virtuels.

A2.1.1 Procédure de configuration

La procédure de configuration suit les étapes décrites au paragraphe 6.1.3.1, à l'exception de la deuxième étape, qui est la suivante:

2) Les haut-parleurs virtuels sont déterminés en regardant d'abord les tableaux définis au paragraphe A2.1.2. Chaque sous-section du paragraphe A2.1.2 définit une configuration de haut-parleurs virtuels et leurs coefficients pour une configuration spécifique définie dans la recommandation UIT-R BS.2051-2.

Les autres étapes du processus de configuration, les étapes une (1) et trois (3) à six (6), sont les mêmes que celles décrites au paragraphe 6.1.3.1.

A2.1.2 Haut-parleurs virtuels et tableaux dérivés

Dans les tableaux ci-dessous, les haut-parleurs virtuels (spécifiés par azimut et élévation) se trouvent au premier rang et les haut-parleurs physiques à la première colonne. Les emplacements des haut-parleurs virtuels ont la même position nominale et réelle. Le tableau montre les coefficients dérivés entre les haut-parleurs virtuels et les haut-parleurs physiques.

Système A: 0+2+0

Pour le système A:0+2+0, la méthode utilisée est le sous-mixage du système B:0+5+0 au système A:0+2+0 comme décrit au paragraphe 6.1.2.4. Pour obtenir les canaux 0+5+0, les haut-parleurs virtuels pour le système B:0+5+0 sont utilisés comme décrit ci-dessous.

Système 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

Système 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						

Système 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				

Système E: 4+5+1

Cette configuration comporte à la fois des haut-parleurs hauts et bas. Pas besoin de haut-parleurs virtuels car la configuration est complète.

Système 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						

Système 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				

Système H: 9+10+3

Contient des haut-parleurs dans les hémisphères hautes et basses; la configuration est complète donc pas besoin de haut-parleurs virtuels.

Système 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	

Système 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				