

Moteur Audio 3D

pour l'environnement Unity 3D

1. Origine du projet

Besoin

Disposer d'un moteur de spatialisation WFS adaptable à n'importe quelle configuration de haut-parleurs et intégré dans un environnement de design visuel, sonore et interactif (i.e. un moteur de jeu vidéo tel que Unity)

Contrainte

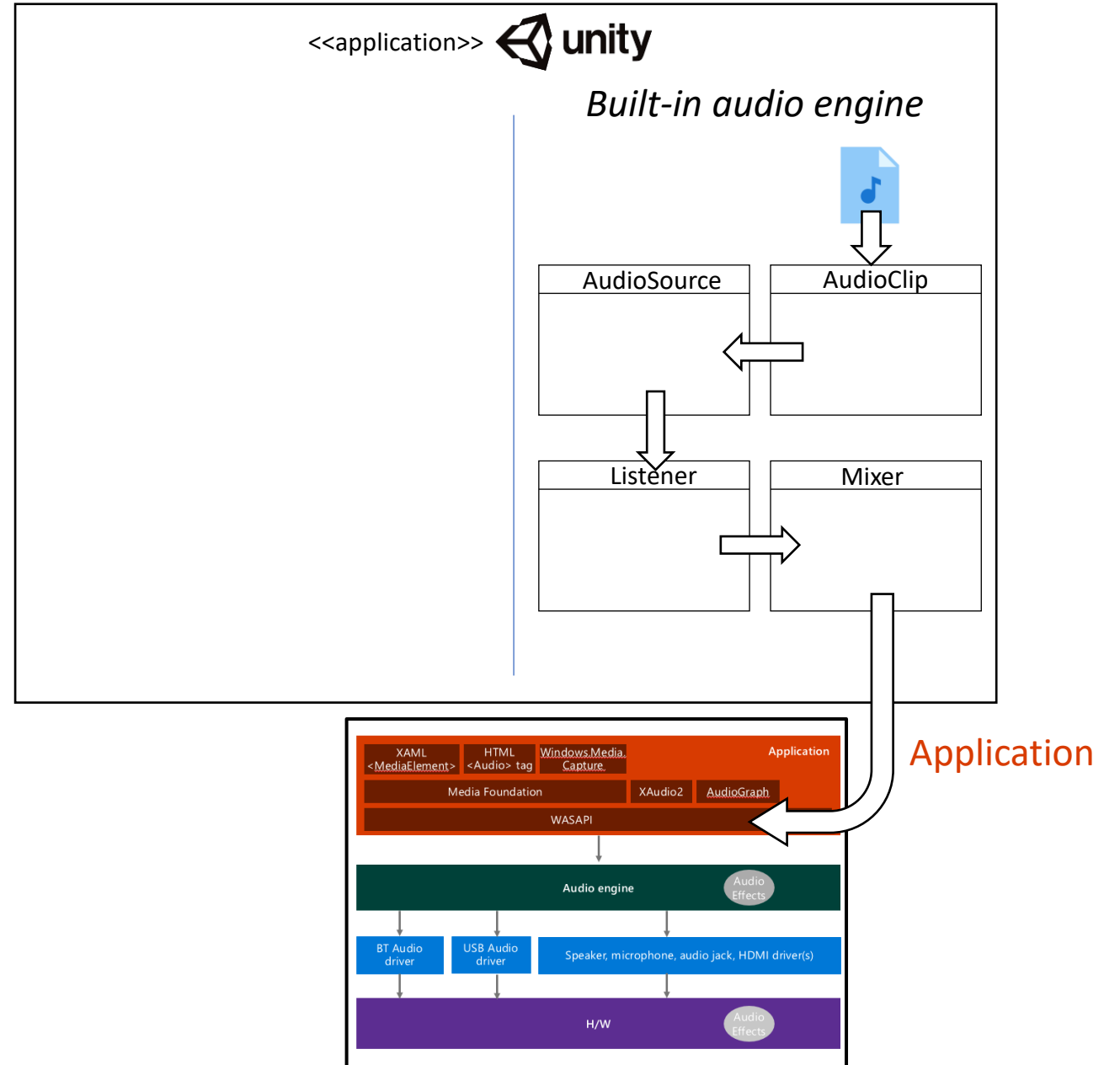
Comme la plupart des stations audionumériques, le graphe audio des moteurs de jeu ne permet pas d'utiliser des bus possédant plus de 8 canaux (correspondant à une configuration 7.1)

Cahier des charges

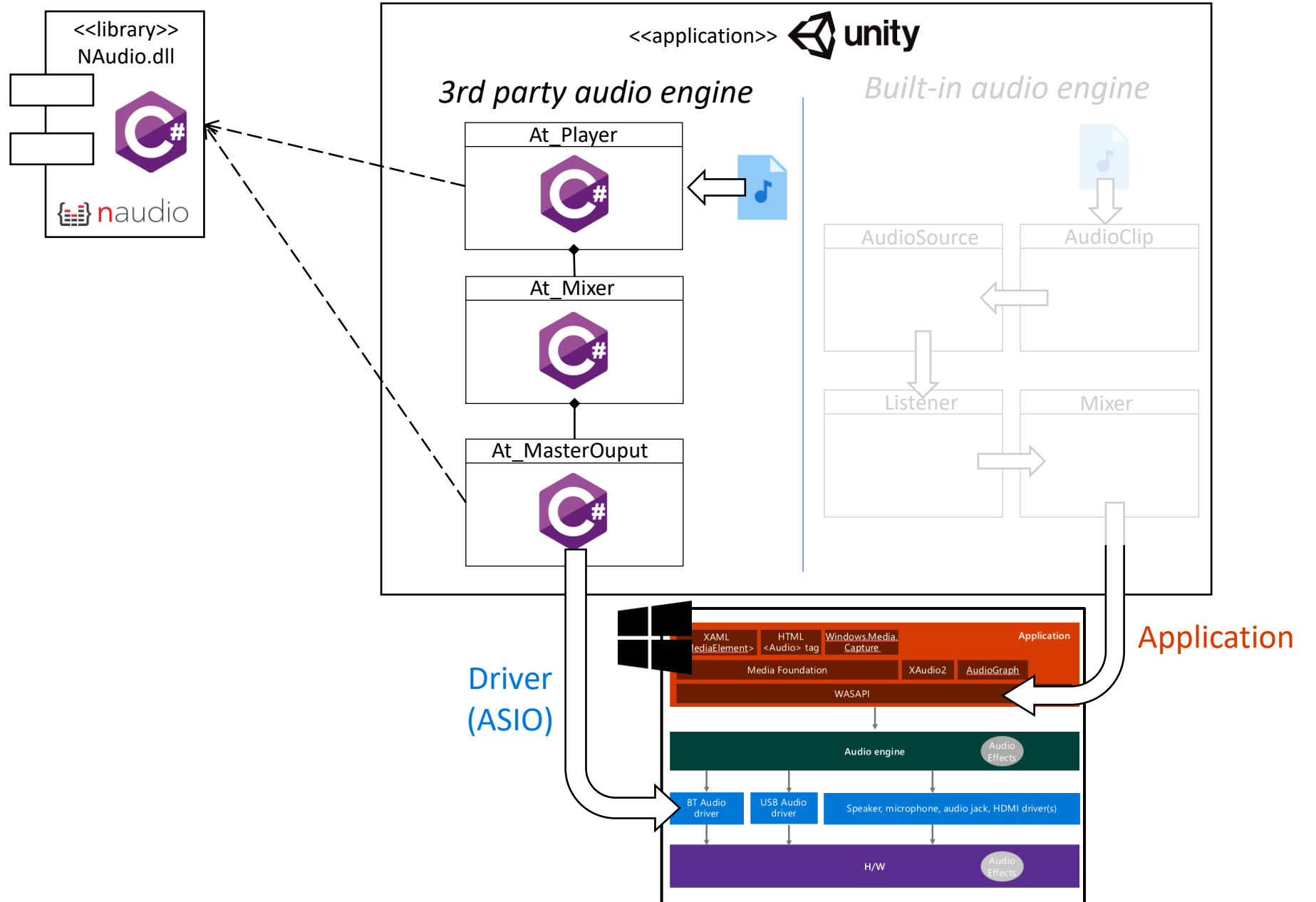
Développer un moteur audio complet pour Unity, ce qui implique la définition d'une double architecture :

- *L'architecture du moteur de spatialisation développé en C++ (bibliothèque de lien dynamique)*
- *L'architecture de l'intégration Unity développée en C# qui implémente le moteur de spatialisation et offre les fonctionnalités essentielles à son bon usage dans le moteur de jeu (sérialisation des données, interface graphique, etc.)*

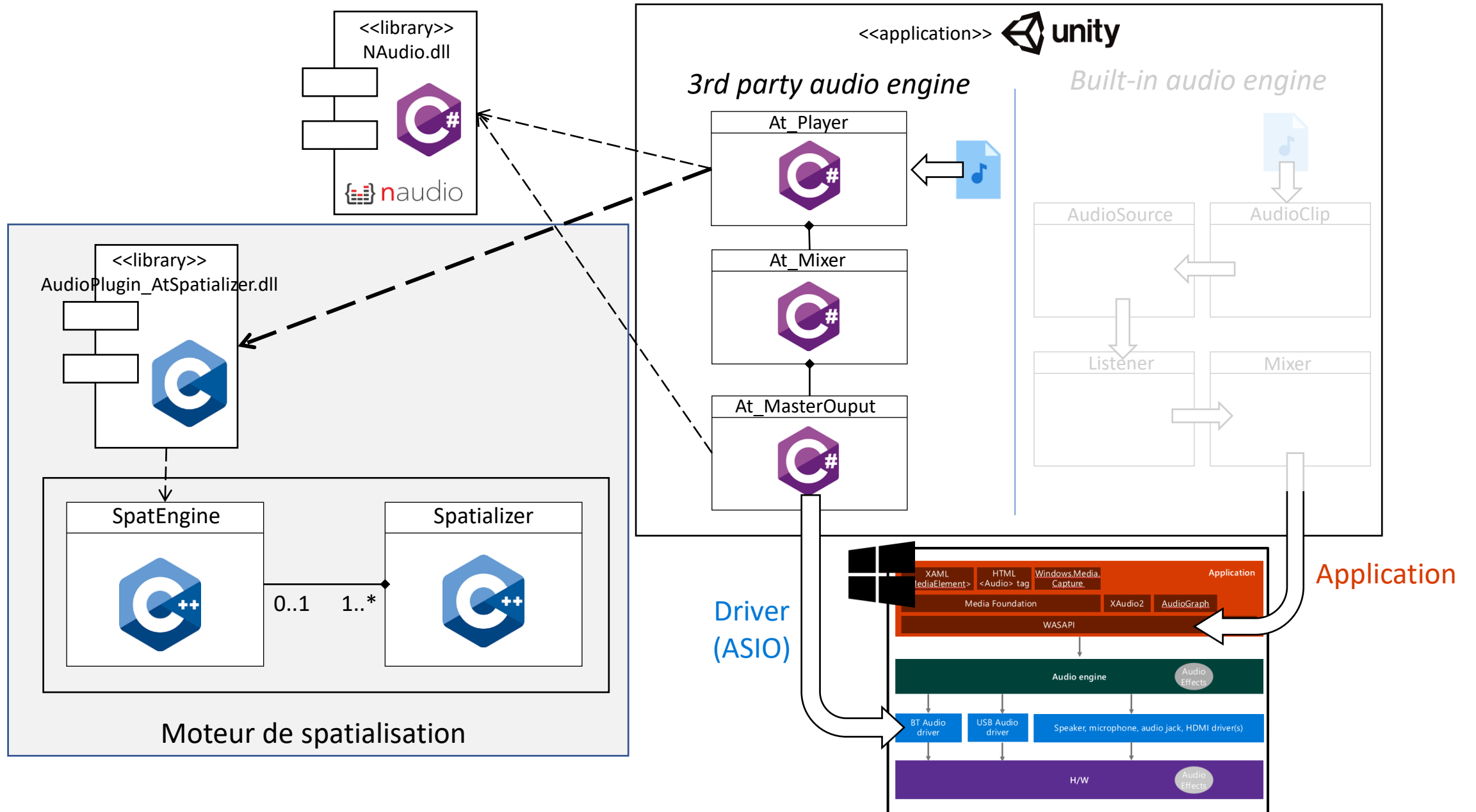
1. Aperçu de l'architecture du moteur audio



1. Aperçu de l'architecture du moteur audio



1. Aperçu de l'architecture du moteur audio

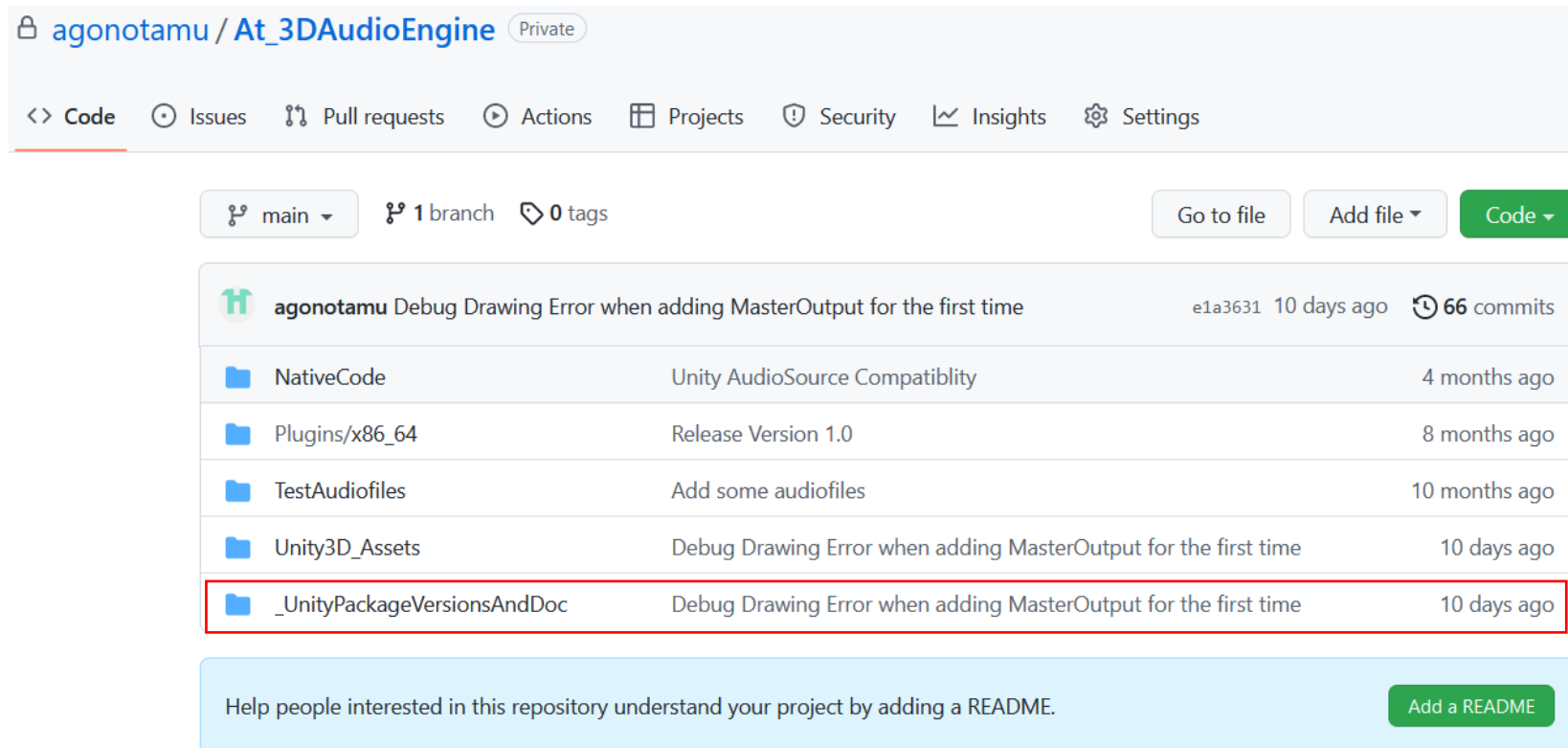


1. Aperçu de l'architecture du moteur audio

Intégration du moteur audio dans Unity

Un fichier « .unitypackage » est disponible sur le Github du projet à l'adresse suivante

https://github.com/agonotamu/At_3DAudioEngine



agonotamu / At_3DAudioEngine Private

<> Code Issues Pull requests Actions Projects Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

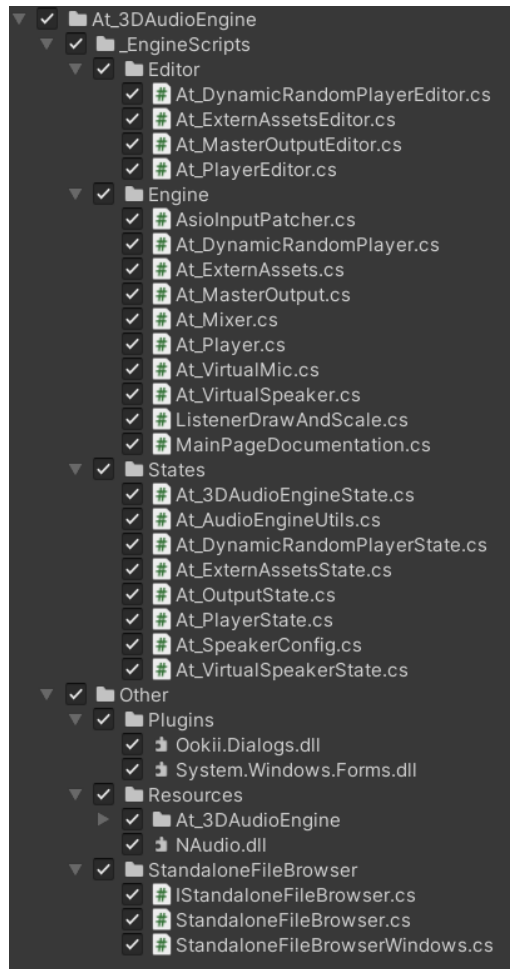
agonotamu	Debug Drawing Error when adding MasterOutput for the first time	e1a3631 10 days ago	66 commits
NativeCode	Unity AudioSource Compatibility	4 months ago	
Plugins/x86_64	Release Version 1.0	8 months ago	
TestAudiofiles	Add some audiofiles	10 months ago	
Unity3D_Assets	Debug Drawing Error when adding MasterOutput for the first time	10 days ago	
_UnityPackageVersionsAndDoc	Debug Drawing Error when adding MasterOutput for the first time	10 days ago	

Help people interested in this repository understand your project by adding a README. Add a README

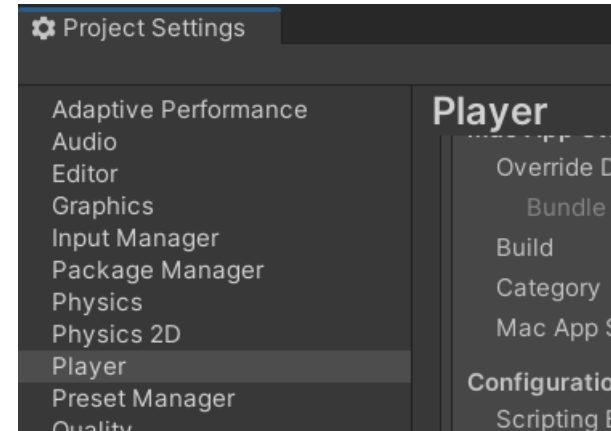
1. Aperçu de l'architecture du moteur audio

Intégration du moteur audio dans Unity

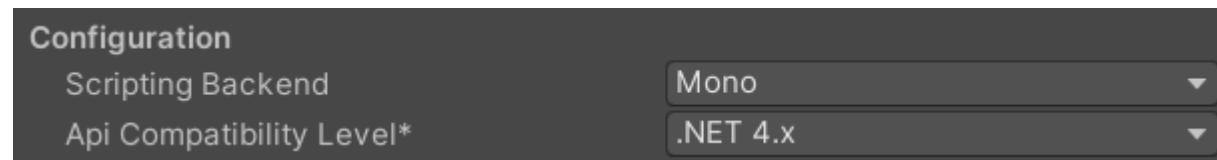
On importe l'ensemble des fichiers du package



Et on modifie quelques paramètres du « Player » dans les « Project Settings »



(1) Utiliser la dernière version du framework .NET :

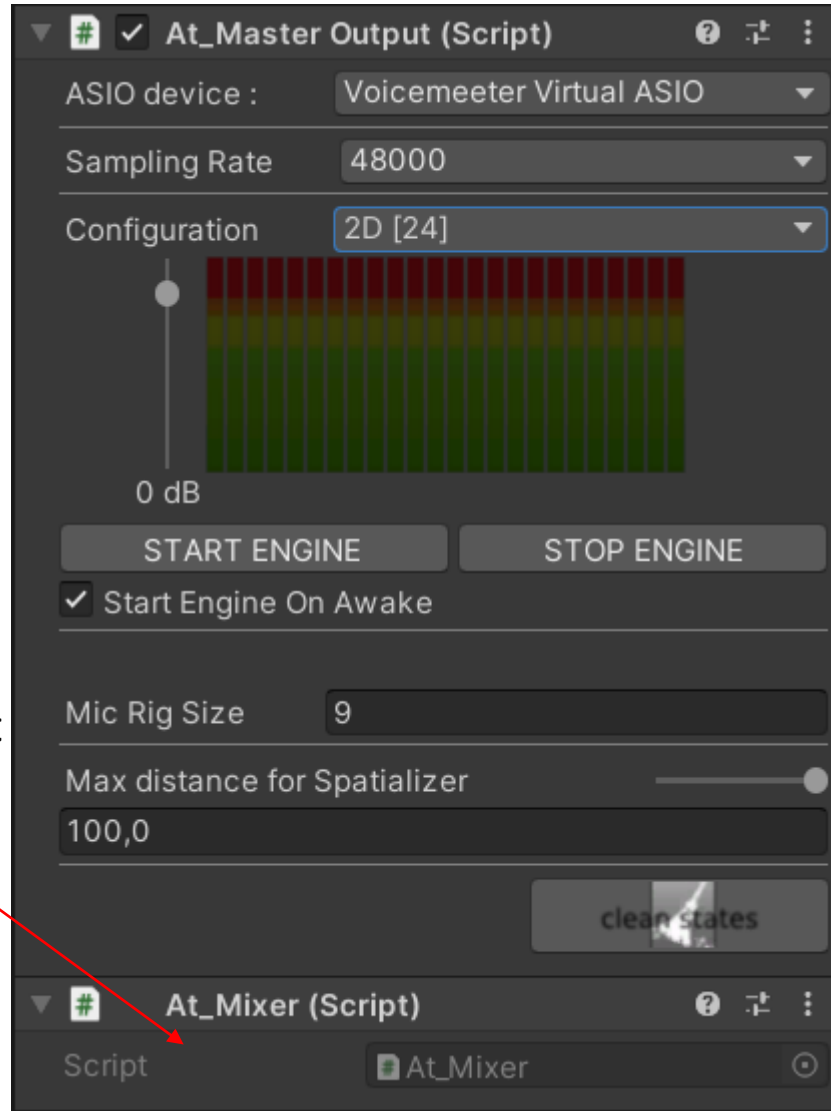


(2) Autoriser le code « unsafe » (utilisation de pointeurs) utilisé par Naudio :



2. Utilisation du moteur audio et aperçu de l'interface des *Components Unity*

a) Le component `At_MasterOutput`



! C'est le premier component que l'on doit ajouter pour l'intégration du moteur !

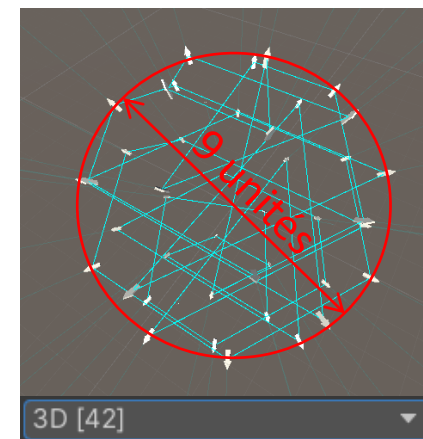
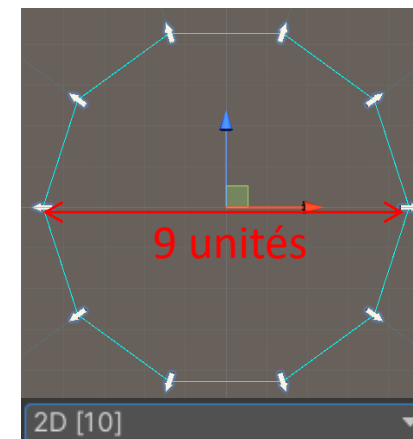
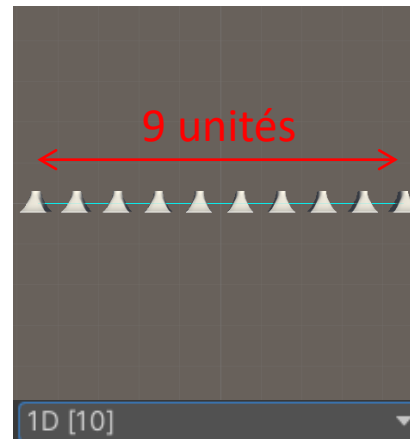
Le component At_Mixer est ajouté automatiquement

Le component « `At_MasterOutput` permet de définir le périphérique ASIO en sortie et la fréquence d'échantillonnage

Il est aussi en charge de l'initialisation de la configuration spatiale du système de haut-parleurs en sortie, soit le **nombre** et la **position** des **microphones virtuels** dans la scène 3D

Pour l'initialisation, on choisit l'une des 15 configurations proposées (8 linéaires, 6 circulaires et 1 géodésique) ainsi que la taille du « rig » de microphone virtuelle.

Exemples :



2. Utilisation du moteur audio et aperçu de l'interface des *Components Unity*

b) Remarques sur l'algorithme de spatialisation WFS

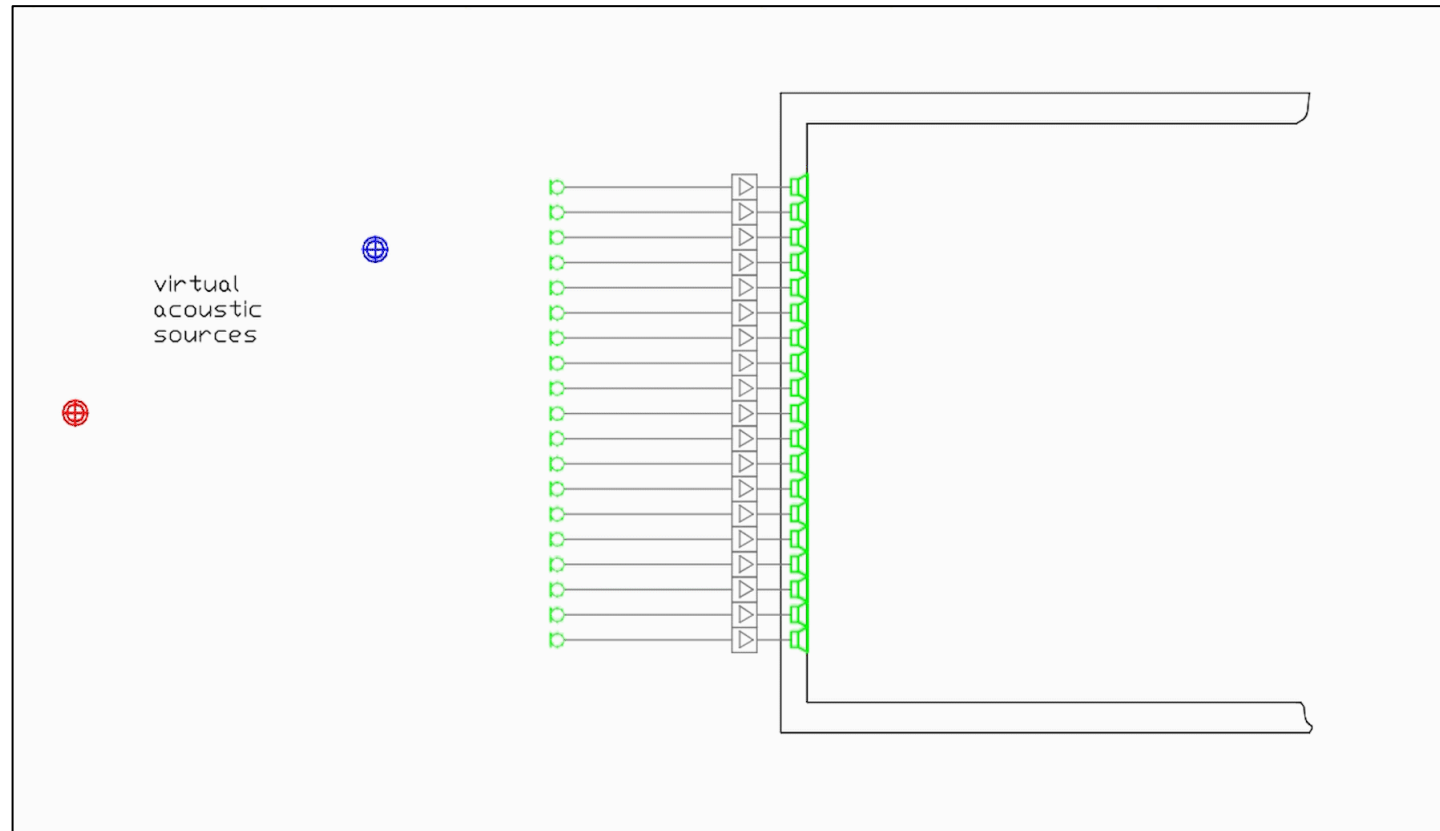
$$p(\vec{r}, \omega) = \iint_{\partial\Omega} \left[(\vec{\nabla} p_0 \cdot \vec{n}) \frac{e^{-ikR}}{4\pi R} - p_0 \cos \alpha (1 + ikR) \frac{e^{-ikR}}{4\pi R^2} \right] dS$$

INTÉGRALE DE KIRCHHOFF-HELMHOLTZ

Monopôle acoustique

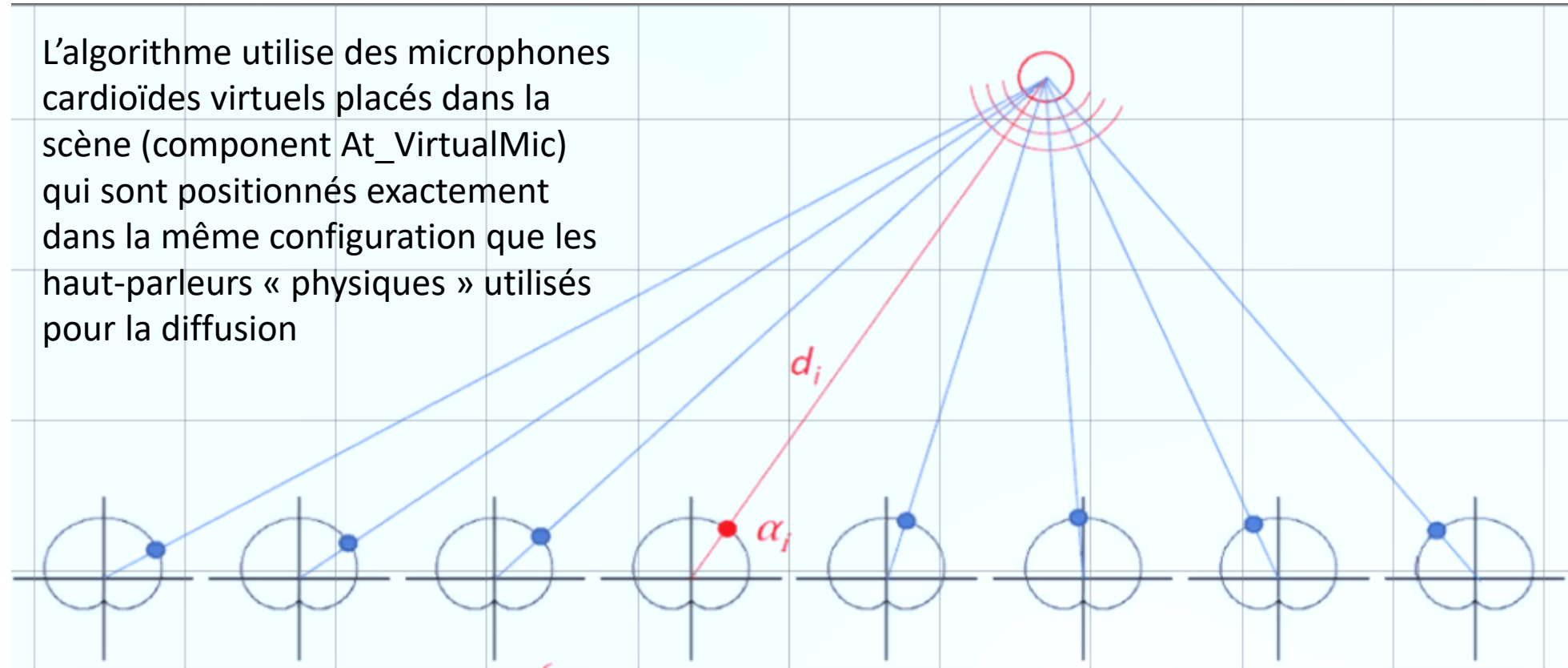
+

Dipôle acoustique



2. Utilisation du moteur audio et aperçu de l'interface des *Components Unity*

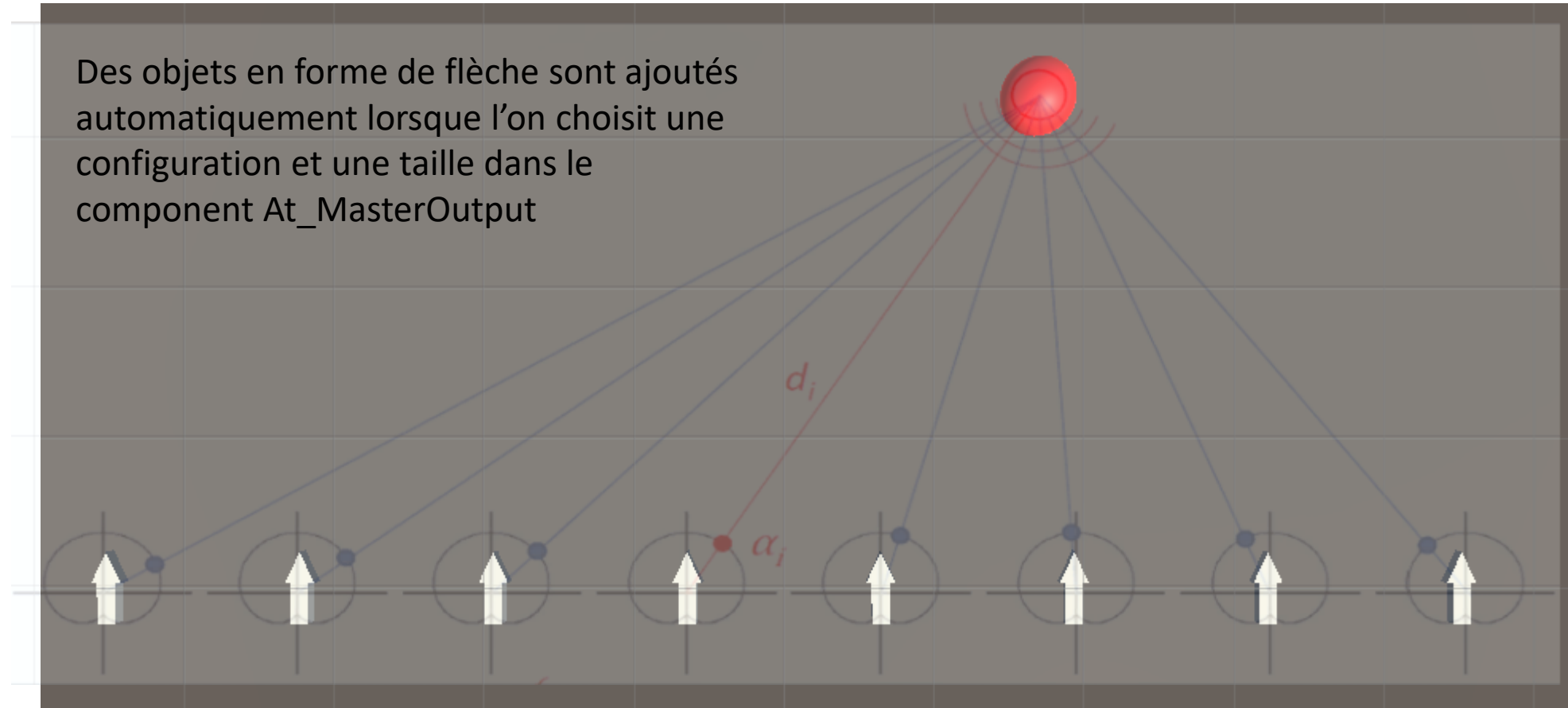
b) Remarques sur l'algorithme de spatialisation WFS et la configuration pour un système de haut-parleurs donné



$$\begin{cases} gain = g(d_i, \alpha_i) \\ retard = r(d_i) \end{cases}$$

2. Utilisation du moteur audio et aperçu de l'interface des *Components Unity*

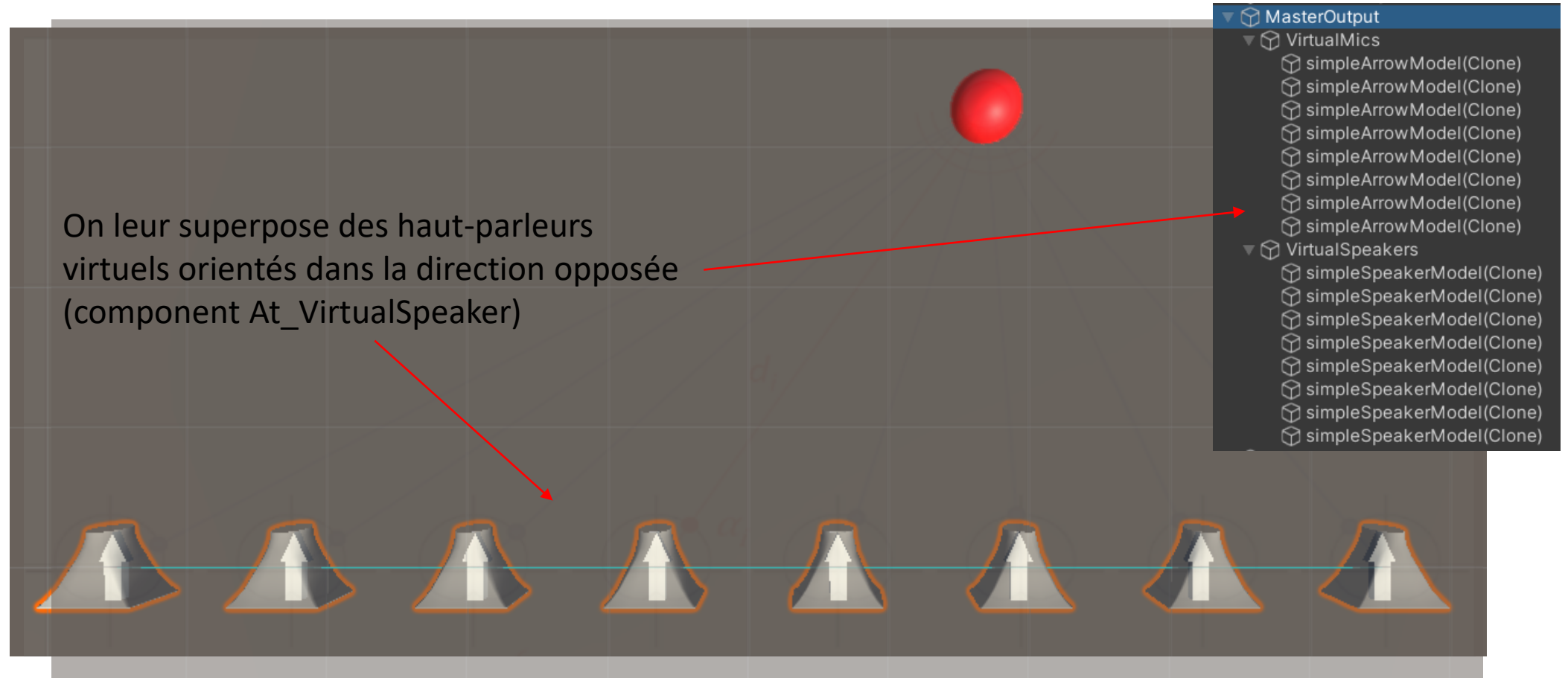
b) Remarques sur l'algorithme de spatialisation WFS et la configuration pour un système de haut-parleurs donné



$$\begin{cases} gain = g(d_i, \alpha_i) \\ retard = r(d_i) \end{cases}$$

2. Utilisation du moteur audio et aperçu de l'interface des *Components Unity*

b) Remarques sur l'algorithme de spatialisation WFS et la configuration pour un système de haut-parleurs donné



$$\begin{cases} gain = g(d_i, \alpha_i) \\ retard = r(d_i) \end{cases}$$

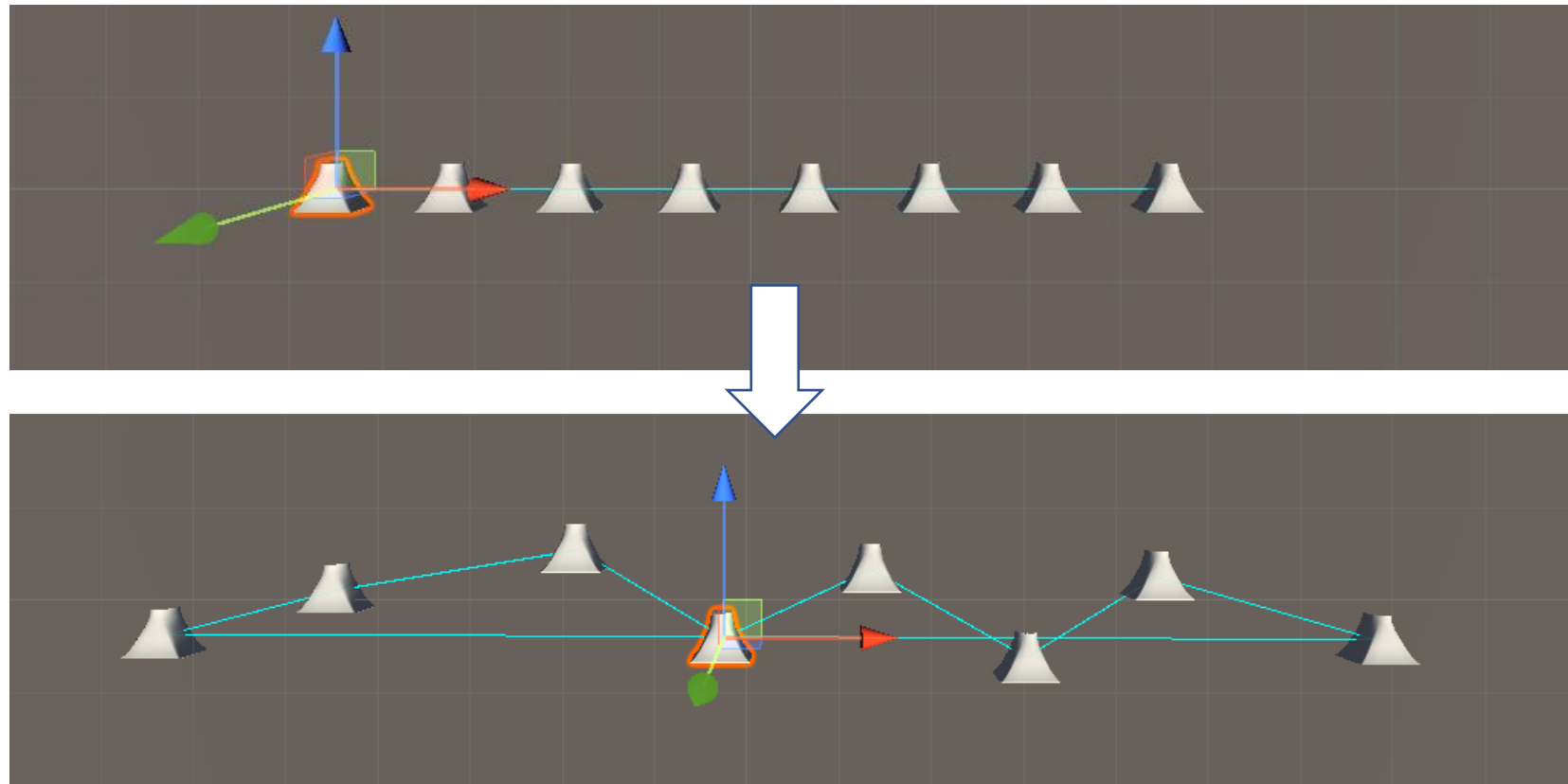
2. Utilisation du moteur audio et aperçu de l'interface des *Components Unity*

b) Remarques sur l'algorithme de spatialisation WFS et la configuration pour un système de haut-parleurs donné

Les configurations créées automatiquement ne correspondent pas nécessairement à la configuration réelle du système de haut-parleurs.

! En les déplaçant, les modèles 3D des haut-parleurs virtuels servent ainsi à calibrer au mieux la configuration des microphones virtuels !

Cas 1D



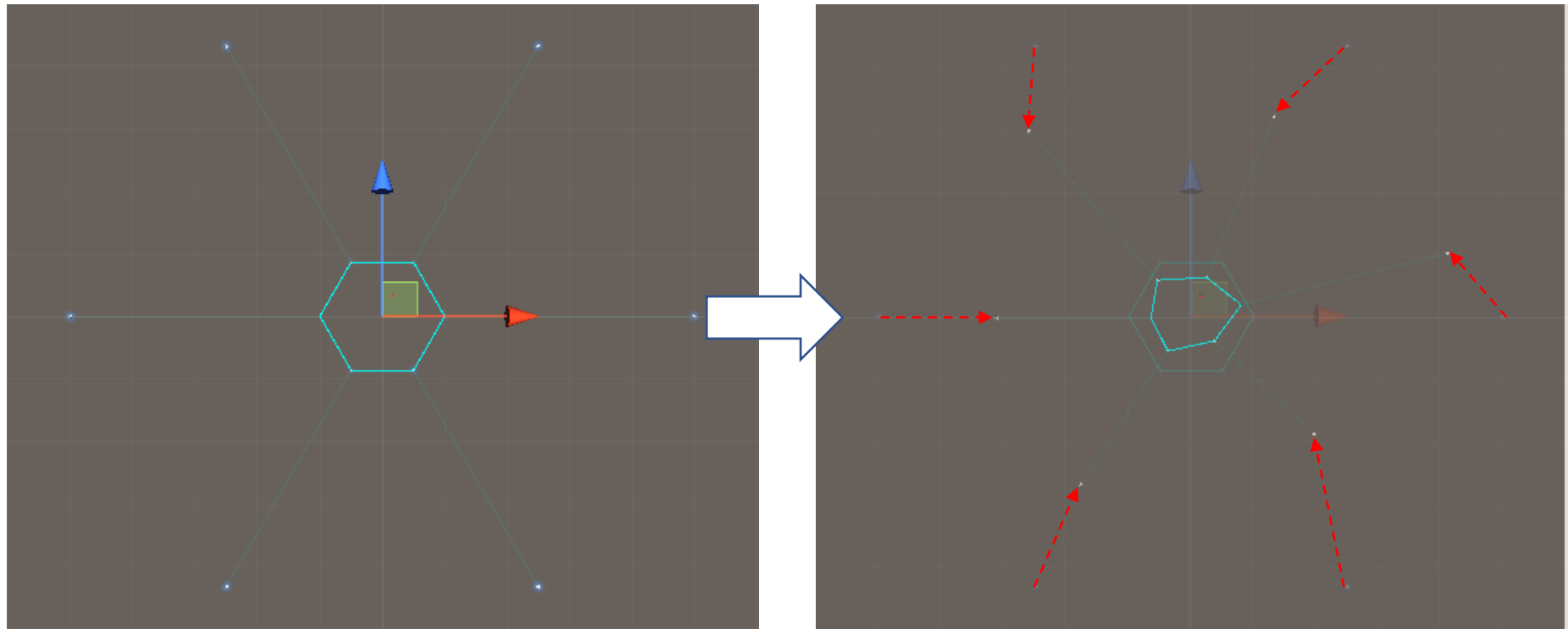
2. Utilisation du moteur audio et aperçu de l'interface des *Components Unity*

b) Remarques sur l'algorithme de spatialisation WFS et la configuration pour un système de haut-parleurs donné

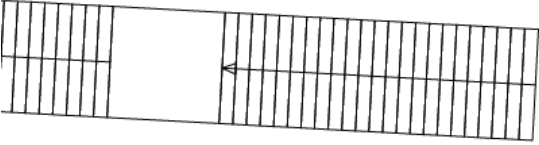
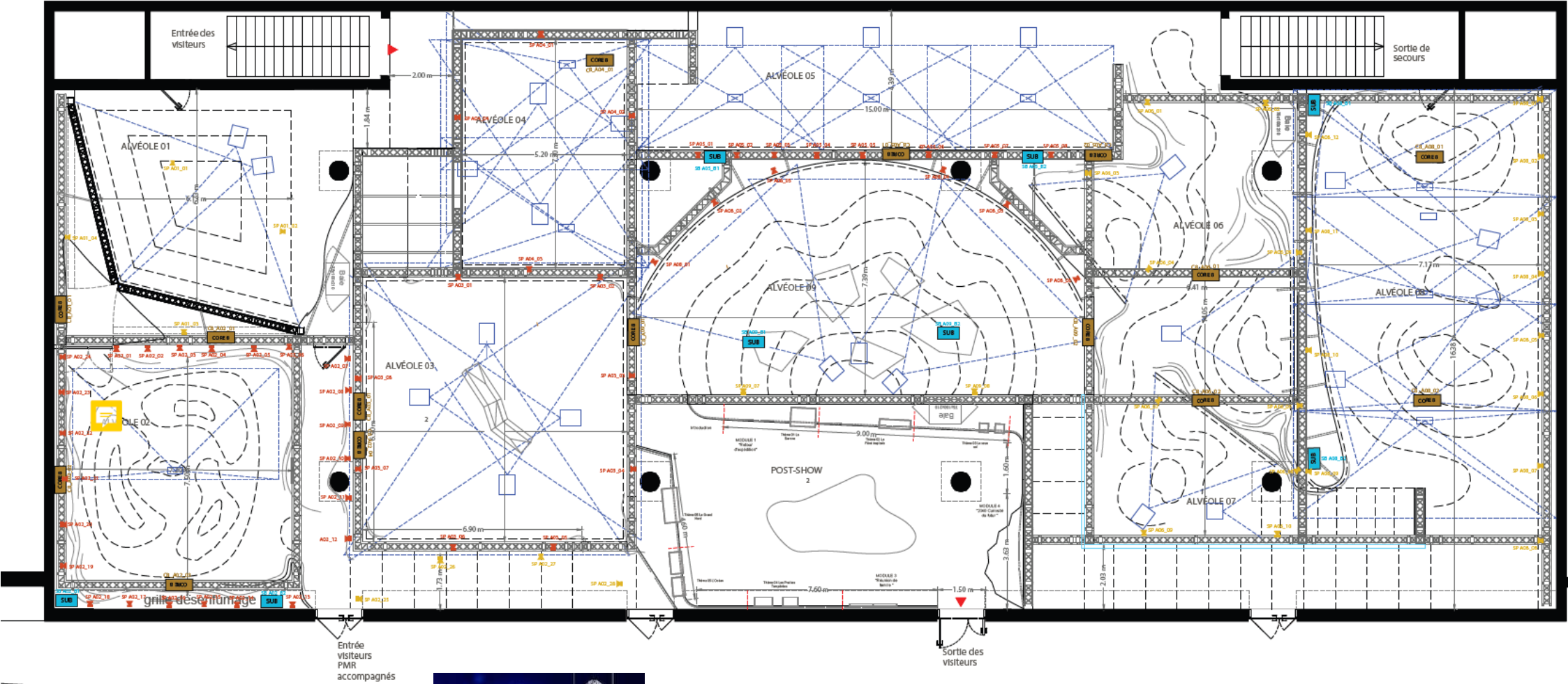
Les configurations créées automatiquement ne correspondent pas nécessairement à la configuration réelle du système de haut-parleurs.

! En les déplaçant, les modèles 3D des haut-parleurs virtuels servent ainsi à calibrer au mieux la configuration des microphones virtuels !

Cas 2D



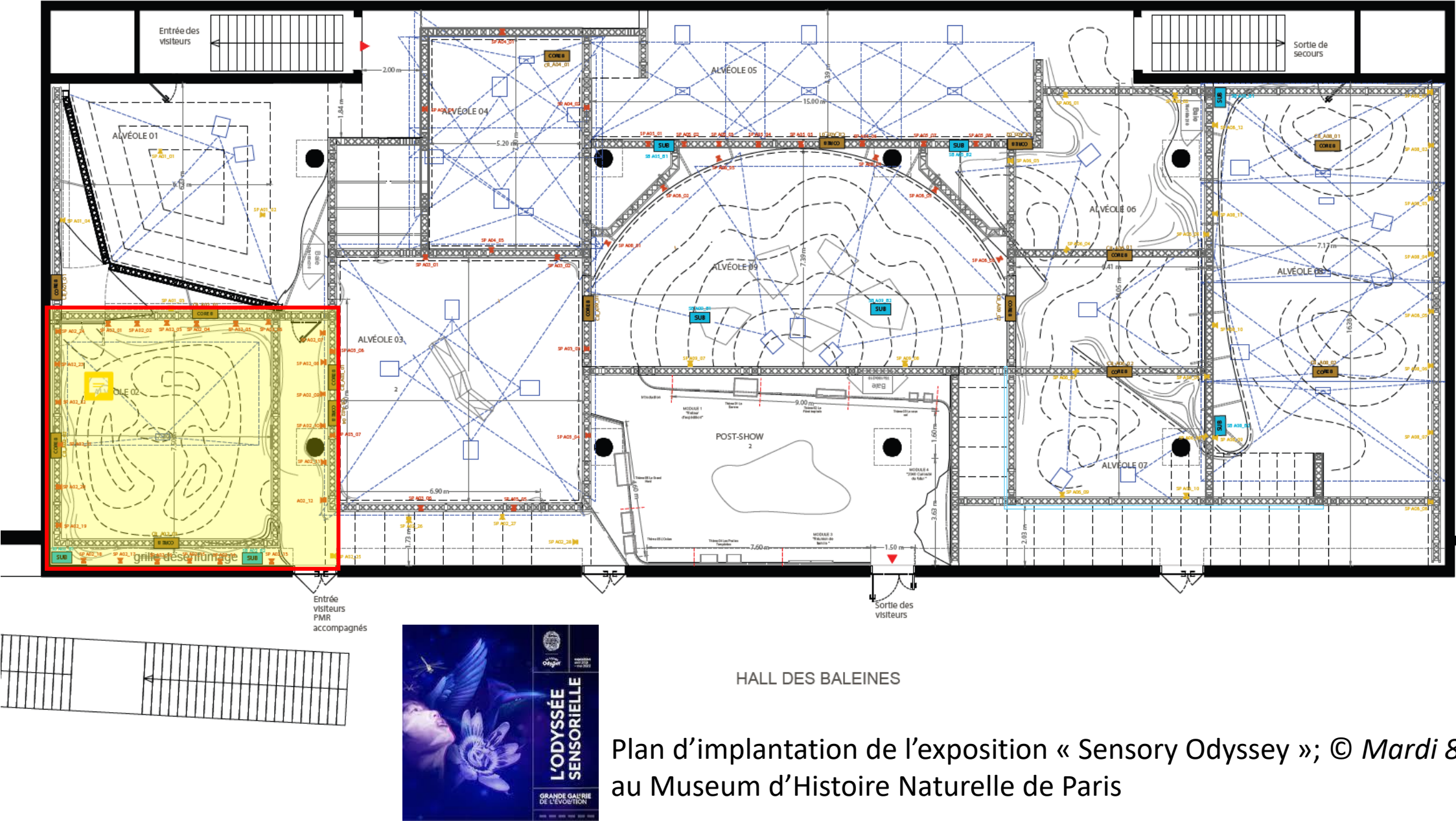
Un cas concret



HALL DES BALEINES

Plan d'implantation de l'exposition « Sensory Odyssey »; © Mardi 8, au Museum d'Histoire Naturelle de Paris

Un cas concret



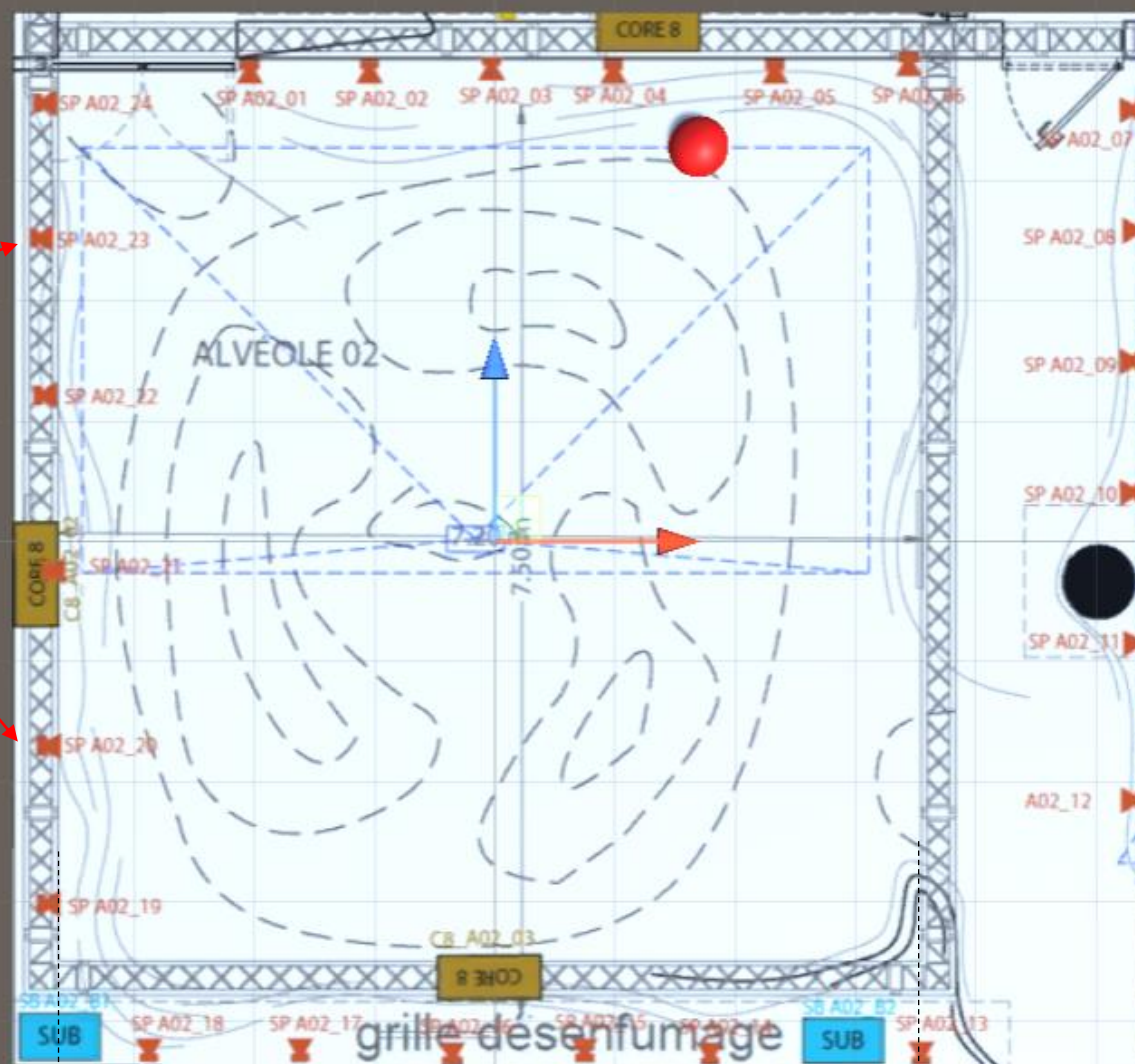
HALL DES BALEINES

Plan d'implantation de l'exposition « Sensory Odyssey »; © Mardi 8, au Museum d'Histoire Naturelle de Paris

Un cas concret

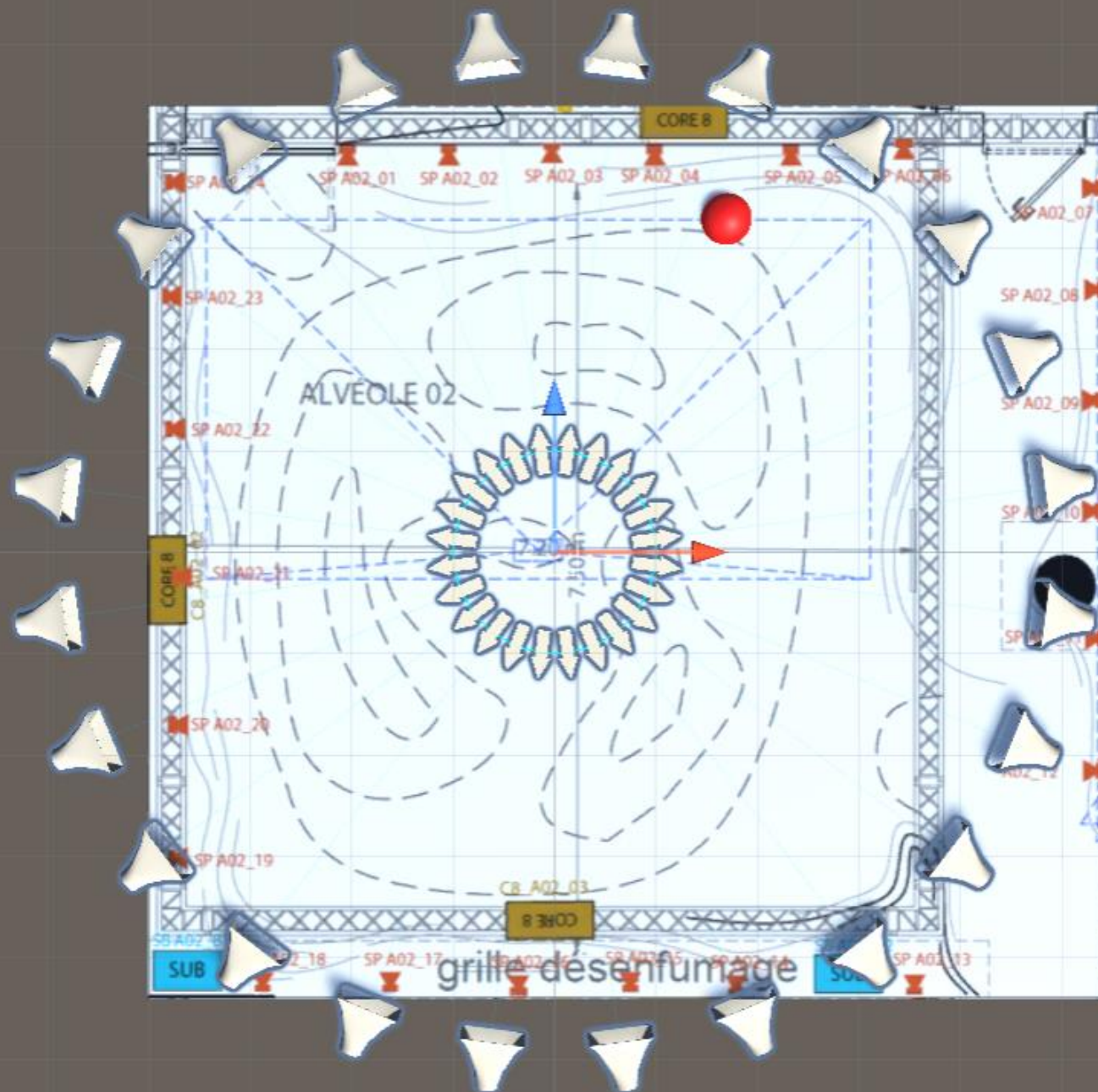
La salle « Savane »
(ALVEOLE_02)

24 haut-parleurs



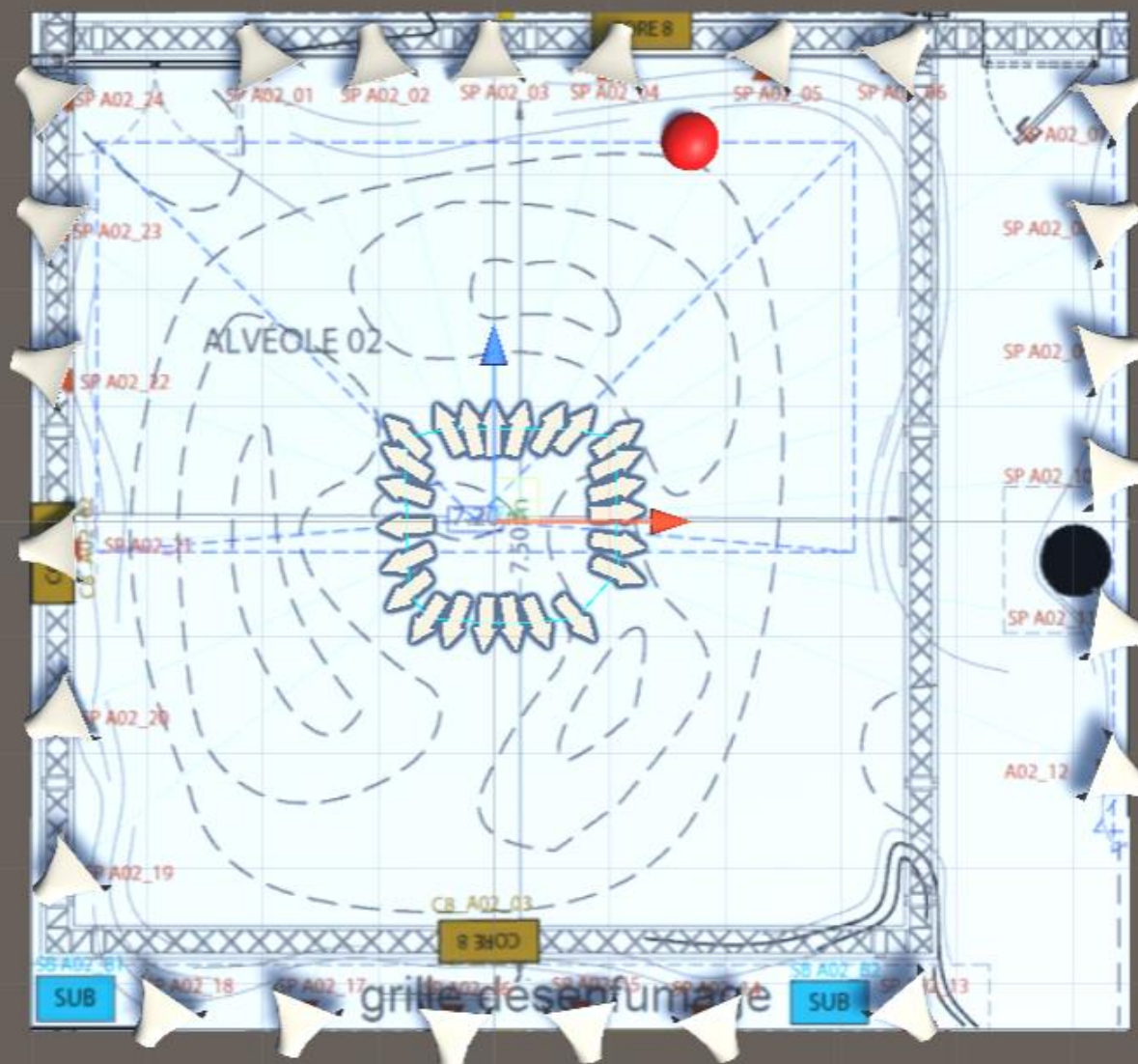
7,20m = 7,2 Unités de Unity

Ajout du Component At_MasterOutput avec une configuration 2D[24]



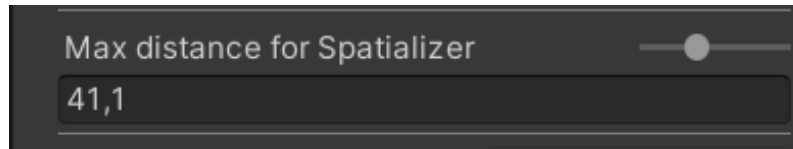
Un cas concret

Déplacement des haut-parleurs virtuels pour adapter la configuration du rig de microphones virtuels à la configuration des haut-parleurs dans le lieu



2. Aperçu de l'interface des objets

a) At_MasterOutput

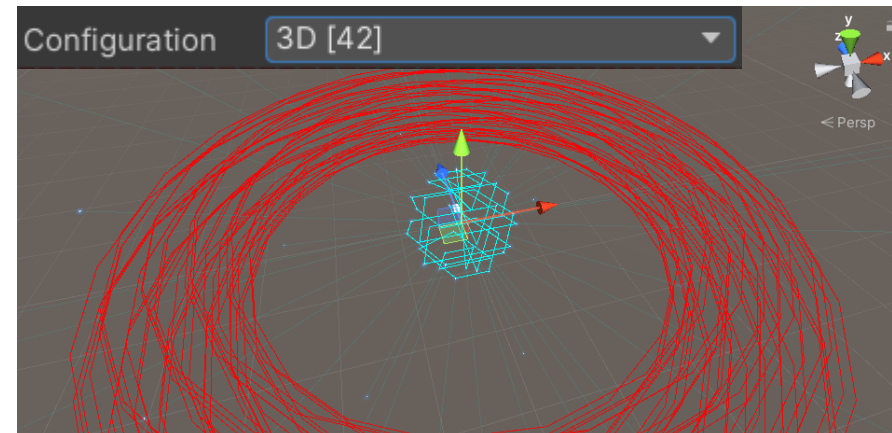
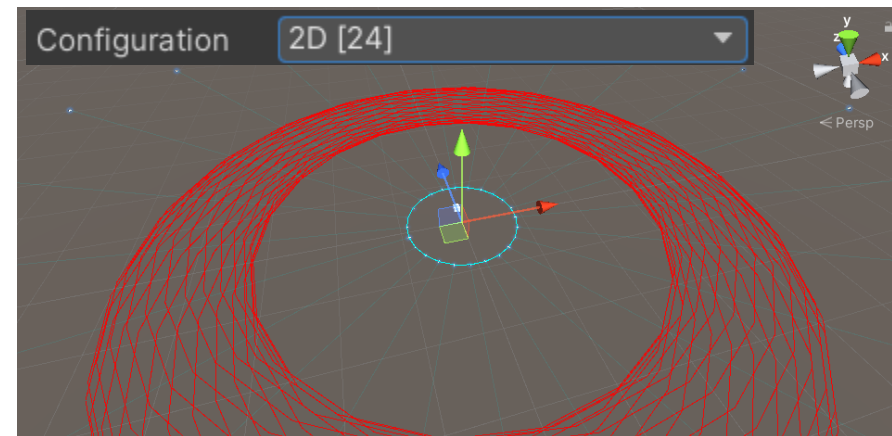
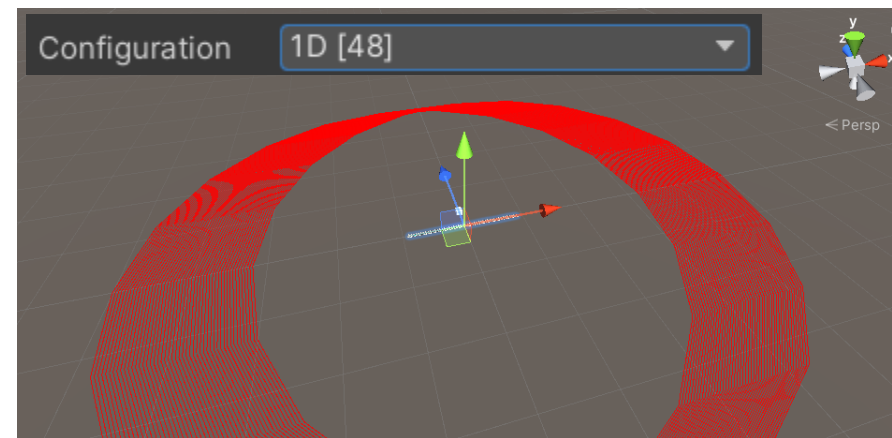


Enfin, il est nécessaire de définir une distance maximum identique pour chaque microphone virtuel

Cette distance convertie en temps (à 340 unités/s) est utilisée pour l'allocation des lignes à retard de chaque canal.

Au-delà de cette distance, les délais sont « seuillés » à la valeur max, donc les effets de spatialisation deviennent aberrant.

Un cercle rouge donnant cette distance est affiché dans l'éditeur. Il convient qu'une source soit dans la zone délimitée par l'intersection de tous les cercles.

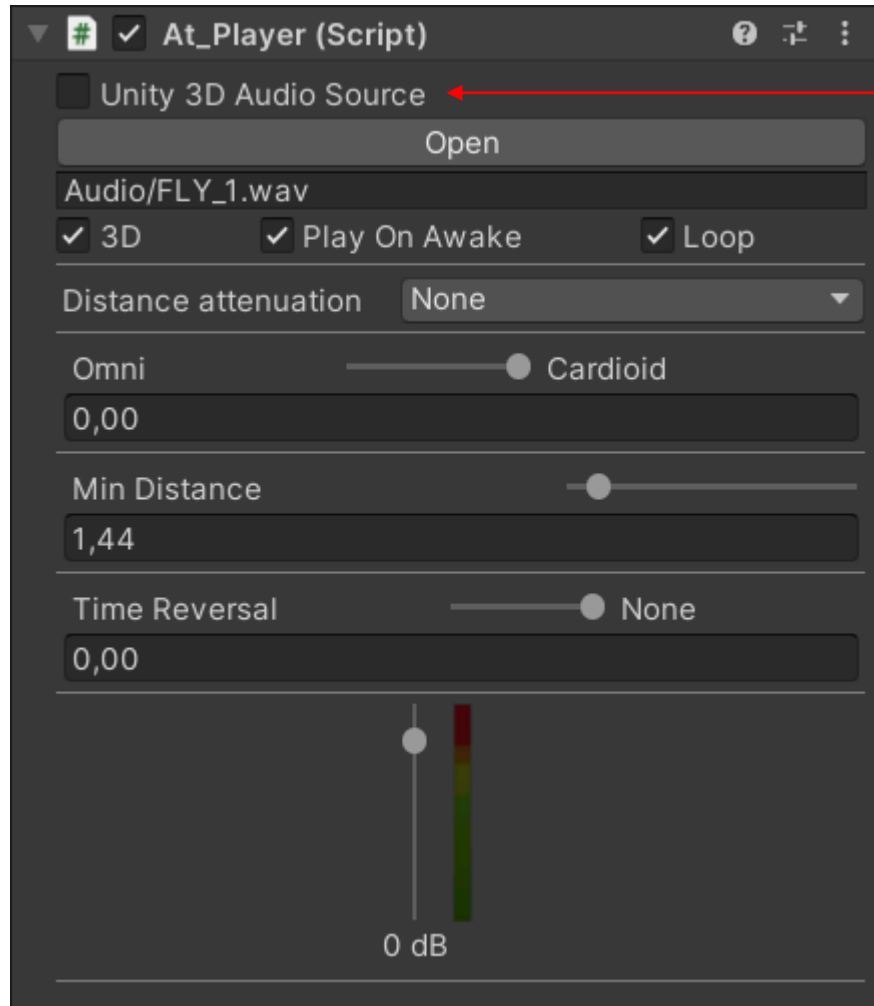


2. Aperçu de l'interface des objets

b) At_Player

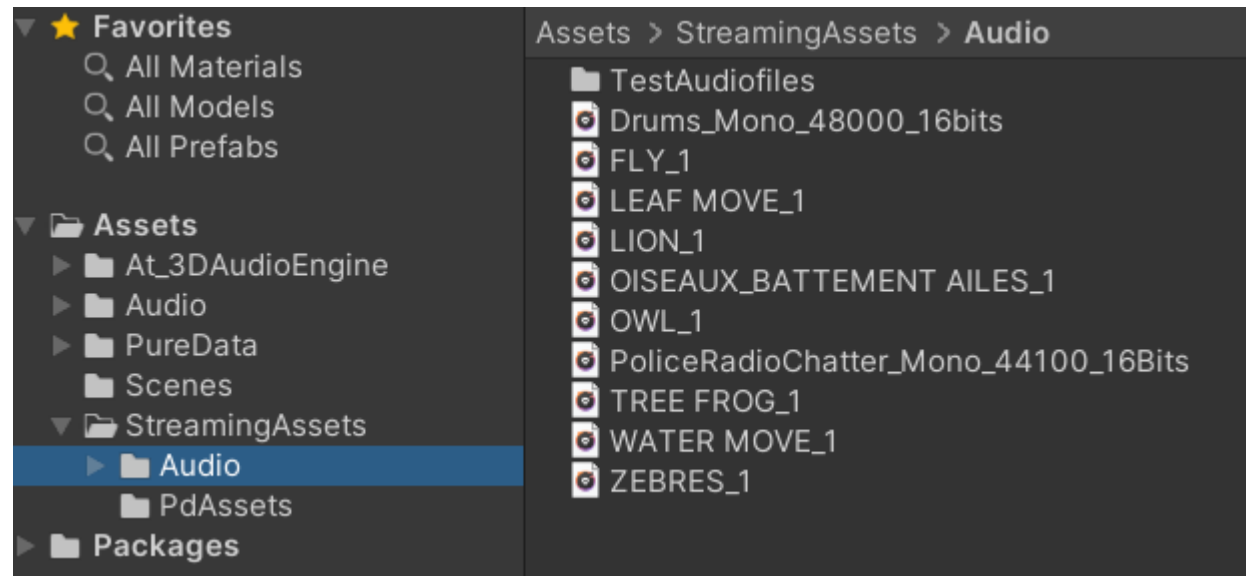
On ajoute un component At_Player à un objet dans la scène 3D pour créer une nouvelle source de son.

- Source 3D



Si cette option est décochée, on utilise le décodeur audio de la librairie NAudio qui est implémenté dans le moteur audio. On peut ainsi lire des fichiers contenant un nombre arbitraire de canaux

Les fichiers audio doivent être dans le dossier \StreamingAssets de Unity (cela signifie qu'ils sont ignorés par le moteur de jeu, comme tout ce qui s'y trouve)

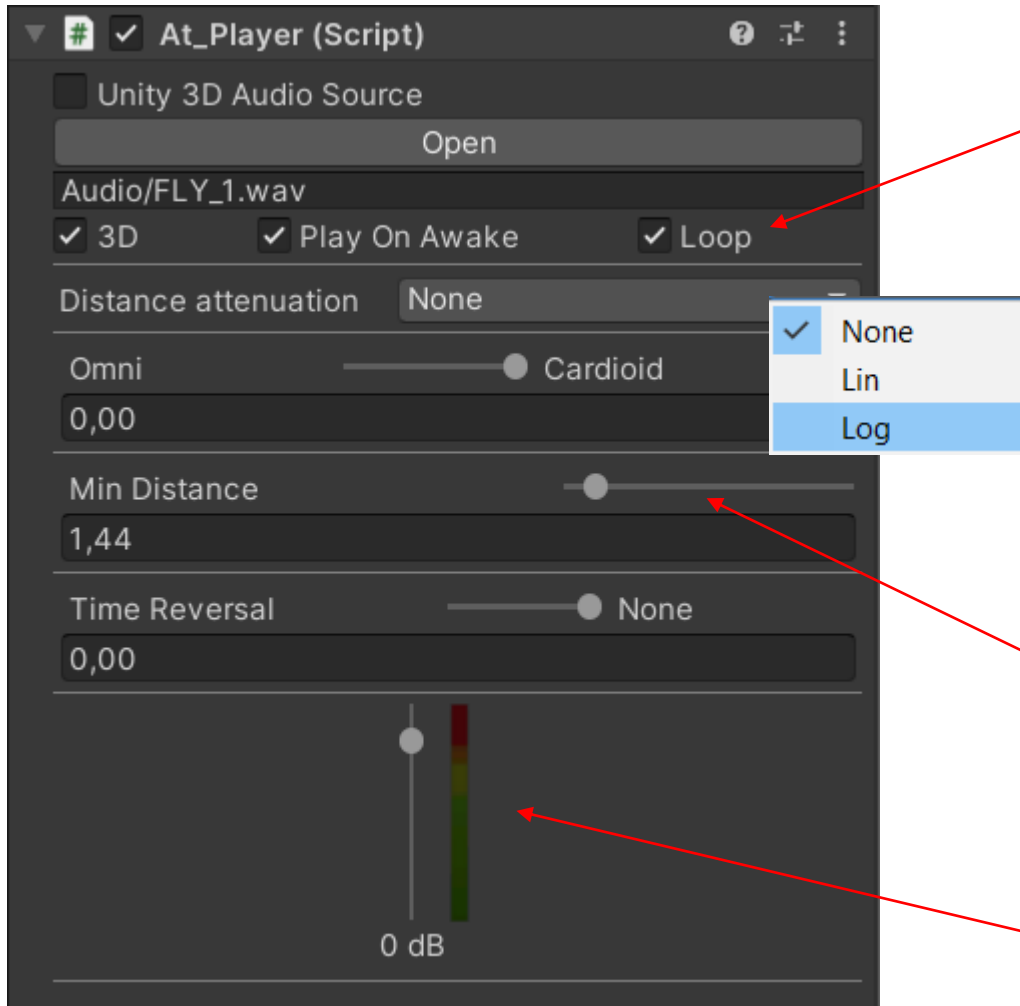


2. Aperçu de l'interface des objets

b) At_Player

On ajoute un component At_Player à un objet dans la scène 3D pour créer une nouvelle source de son.

- Source 3D



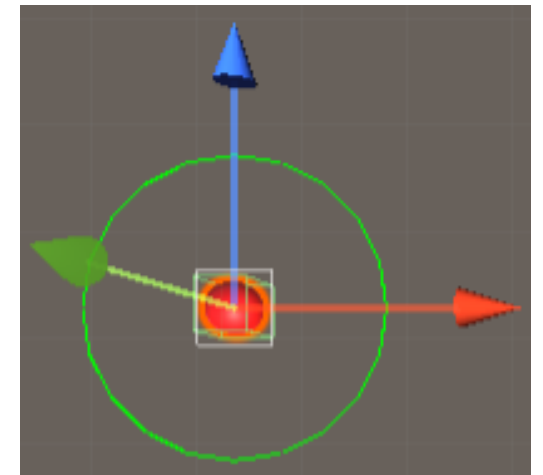
Si la source est « 3D », cela signifie que le premier canal du fichier audio est traité par le moteur de spatialisation

On choisit aussi si la source est lue au démarrage de l'application et si elle doit être jouée en boucle

On sélectionne ici comment le niveau de la source est atténué en fonction de la distance par rapport à l'objet qui porte le component At_MasterOutput (« log » = -6/dec)

On doit aussi définir une distance minimum qui est la distance à partir de laquelle l'atténuation s'applique. Cette distance est représentée par un cercle autour de l'objet.

On règle ici le gain appliqué au fichier audio et on visualise le niveau (RMS sur la durée du buffer)

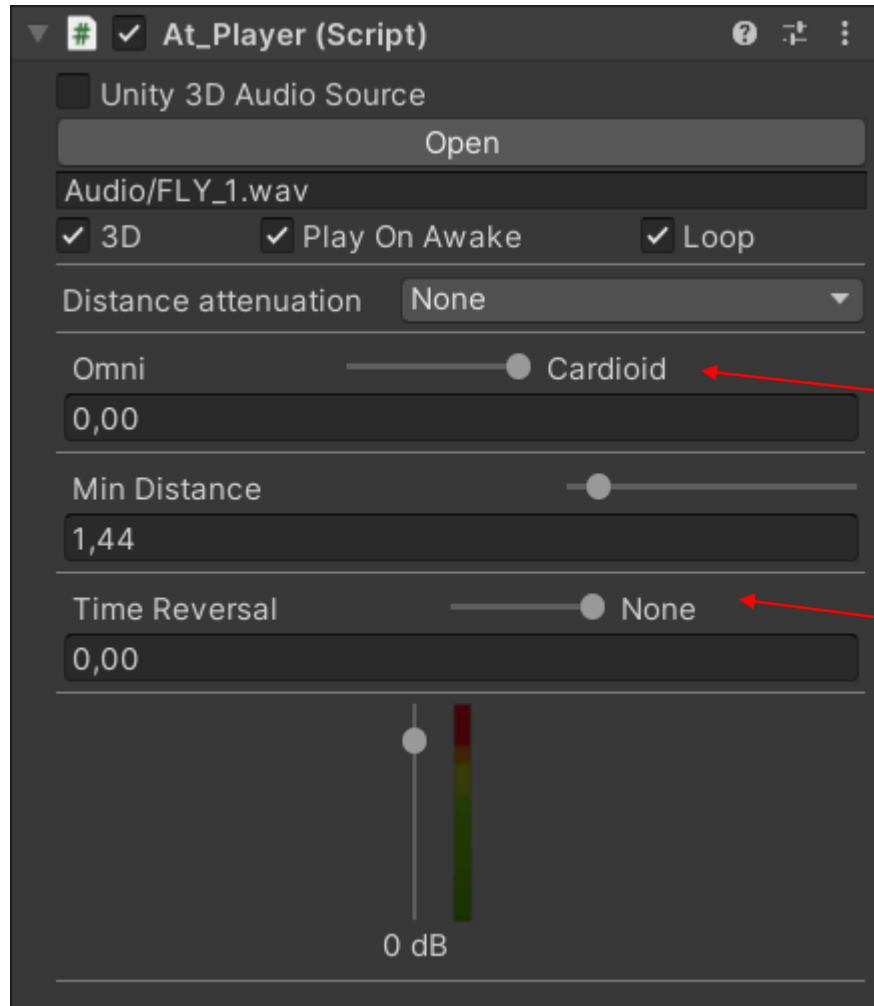


2. Aperçu de l'interface des objets

b) At_Player

On ajoute un component At_Player à un objet dans la scène 3D pour créer une nouvelle source de son.

- Source 3D



Enfin on peut modifier le rendu des effets de spatialisation sonore grâce à deux paramètres :

La balance cardioïde/omni qui réalise une interpolation linéaire entre une directivité cardioïde et omnidirectionnelle pour les microphones virtuels

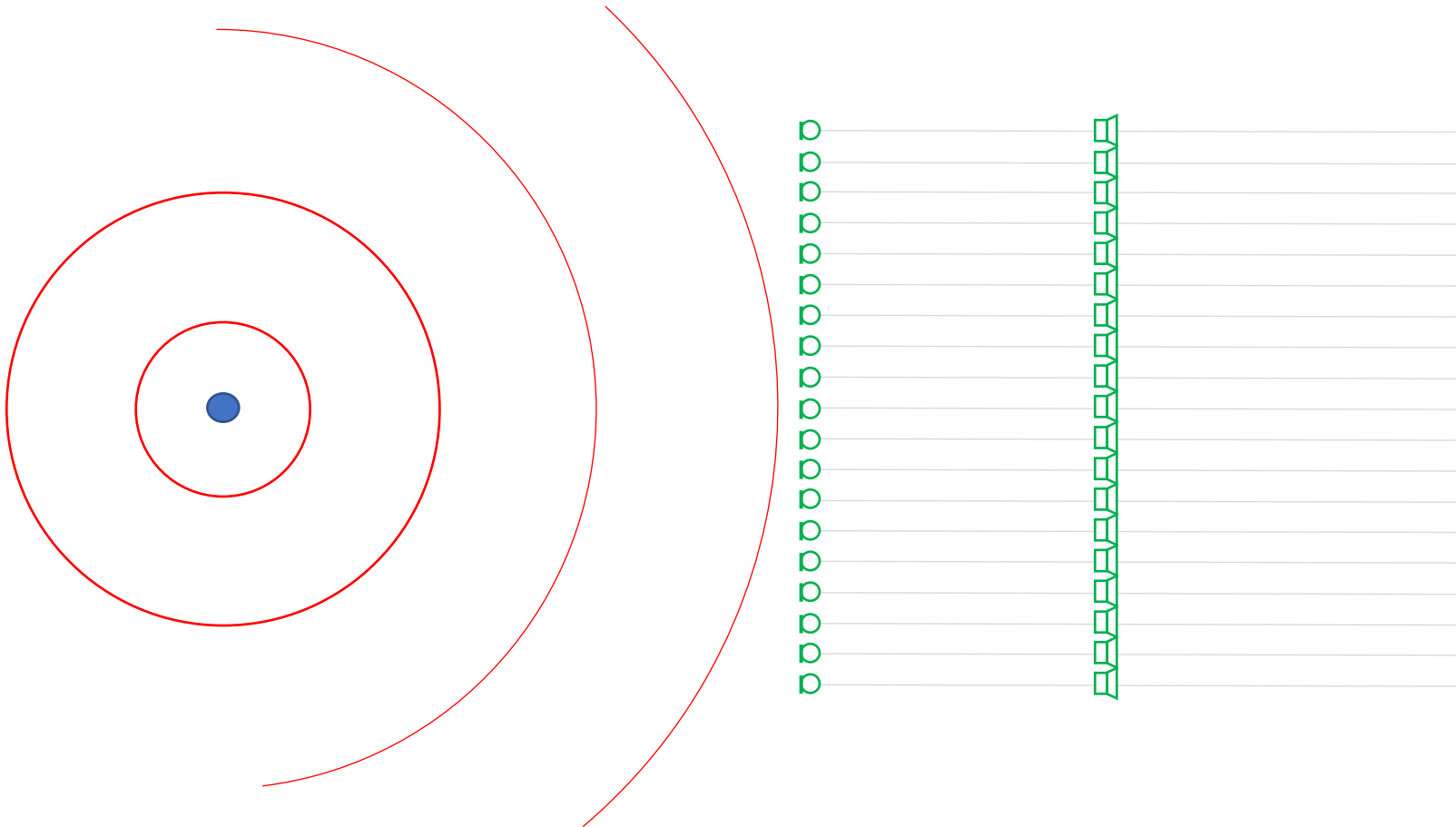
La balance de l'effet de retournement temporel qui crée une source focalisée devant les haut-parleurs à une position symétriquement opposée à la source

2. Aperçu de l'interface des objets

b) At_Player

On ajoute un component At_Player à un objet dans la scène 3D pour créer une nouvelle source de son.

- Source 3D

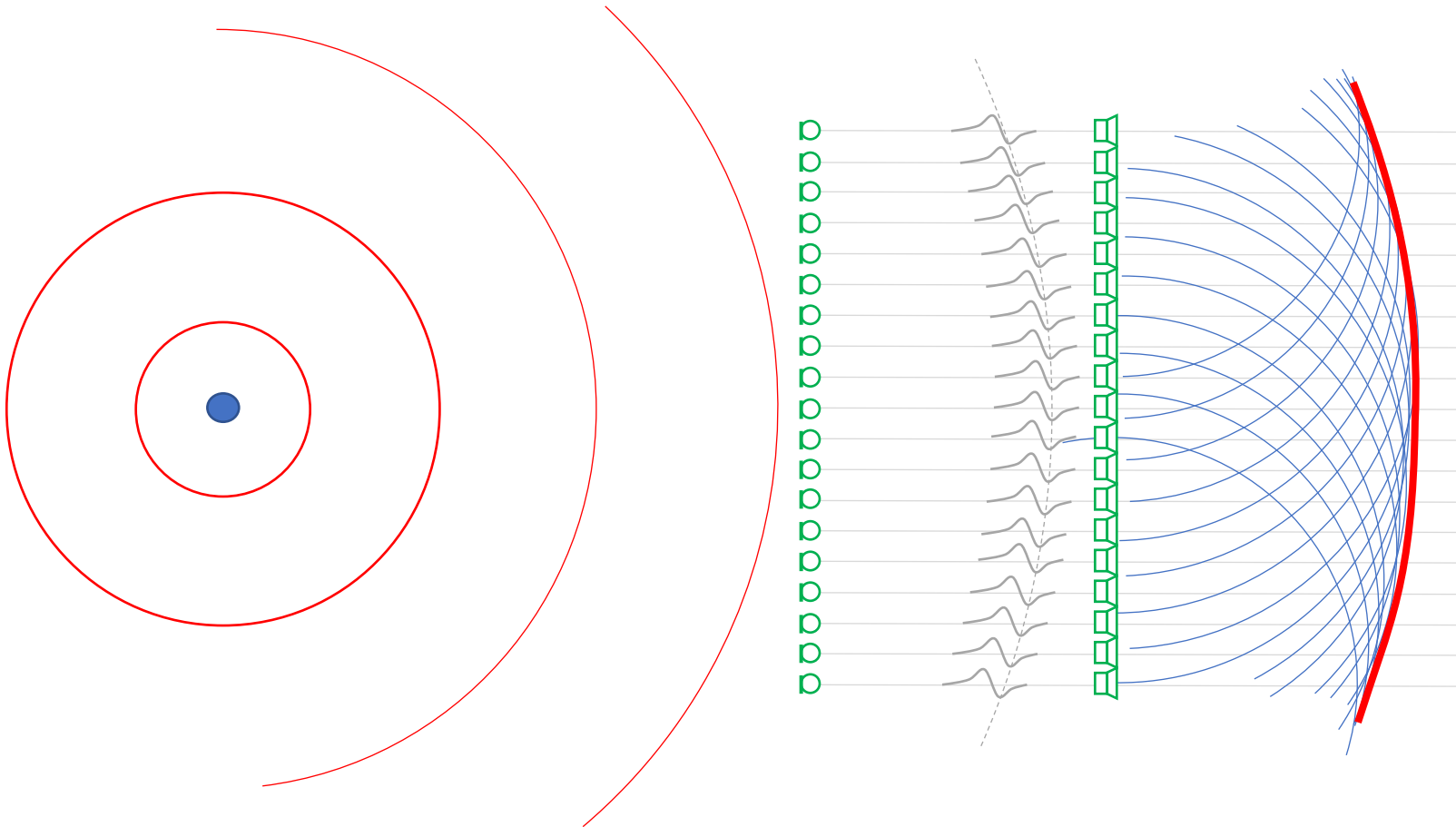


2. Aperçu de l'interface des objets

b) At_Player

On ajoute un component At_Player à un objet dans la scène 3D pour créer une nouvelle source de son.

- Source 3D



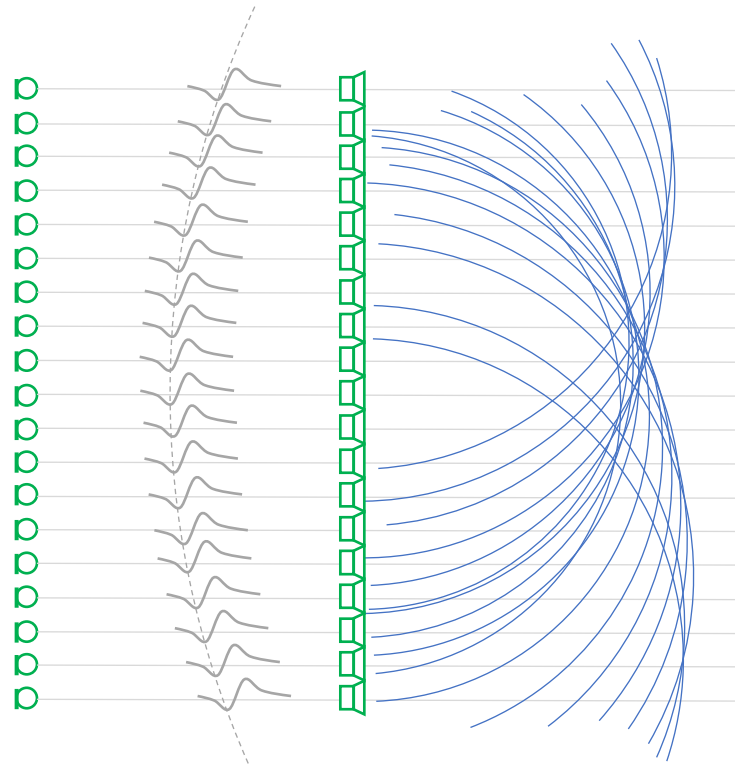
2. Aperçu de l'interface des objets

b) At_Player

On ajoute un component At_Player à un objet dans la scène 3D pour créer une nouvelle source de son.

- Source 3D

Retournement temporel

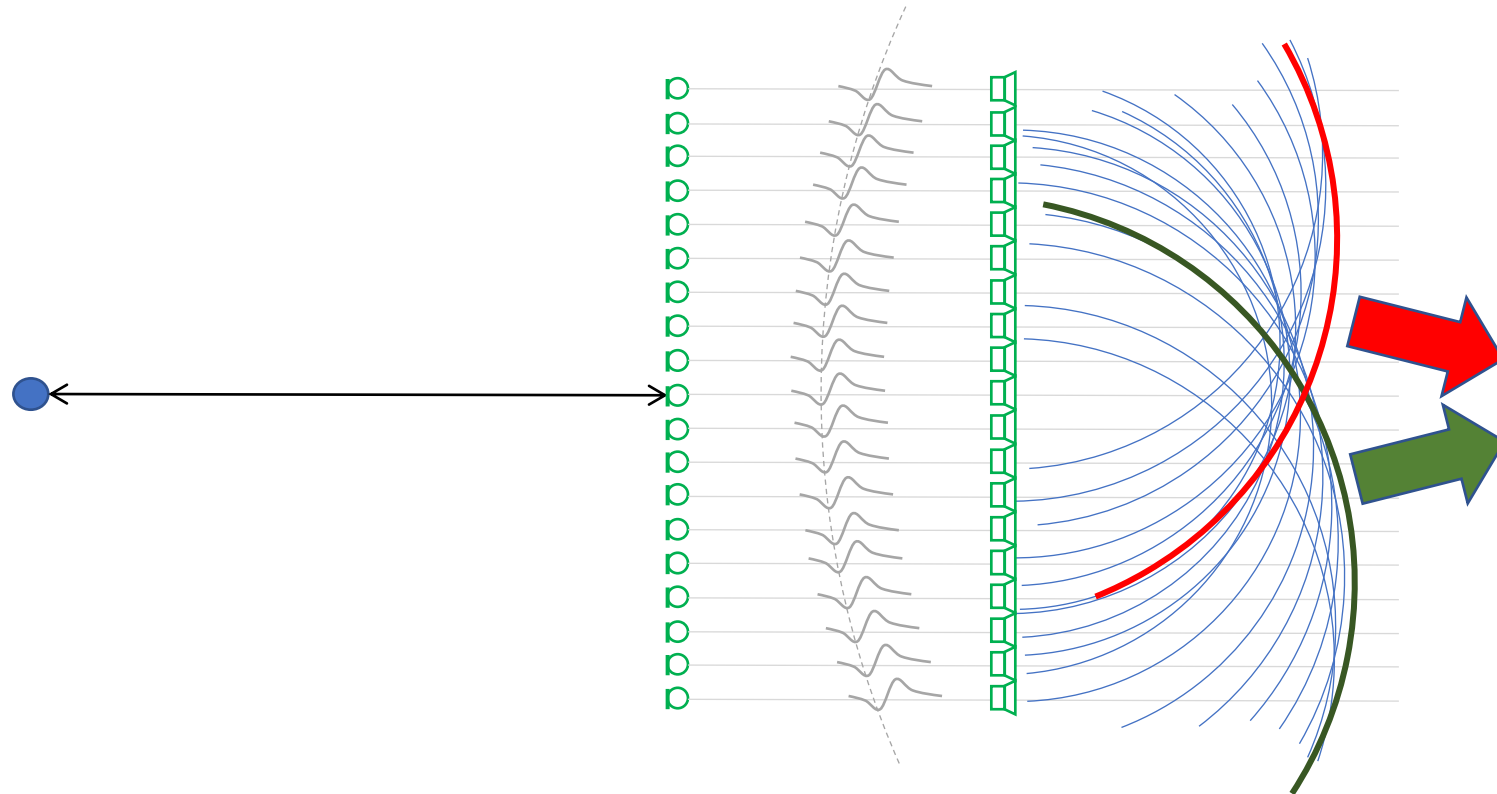


2. Aperçu de l'interface des objets

b) At_Player

On ajoute un component At_Player à un objet dans la scène 3D pour créer une nouvelle source de son.

- Source 3D

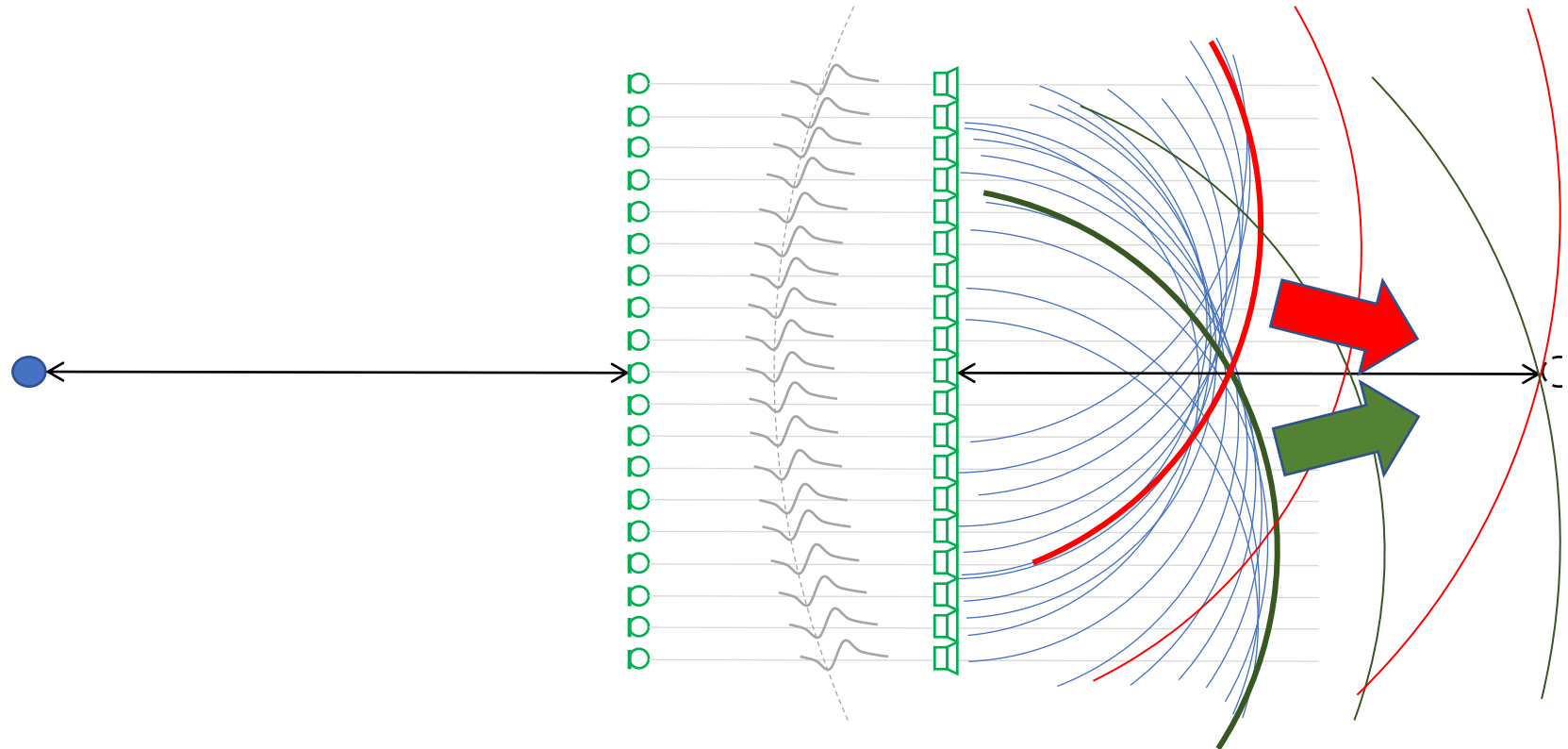


2. Aperçu de l'interface des objets

b) At_Player

On ajoute un component `At_Player` à un objet dans la scène 3D pour créer une nouvelle source de son.

- Source 3D

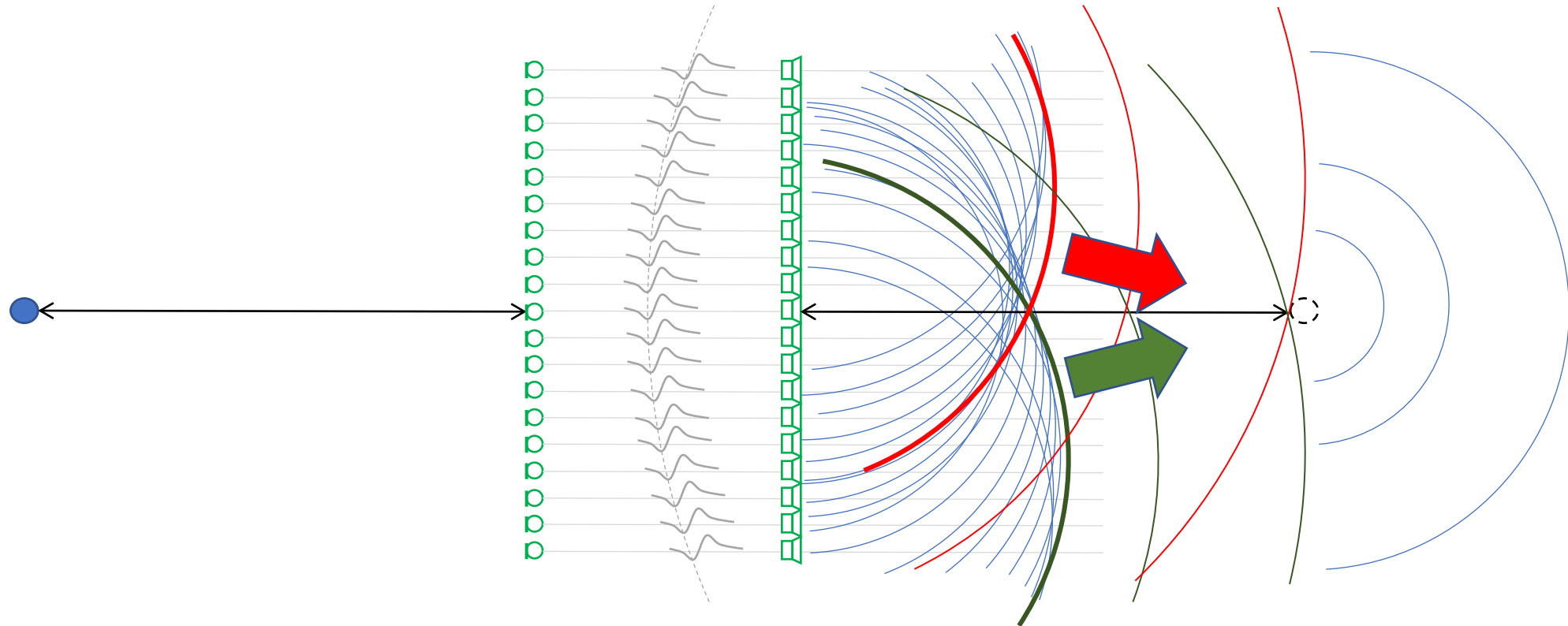


2. Aperçu de l'interface des objets

b) At_Player

On ajoute un component At_Player à un objet dans la scène 3D pour créer une nouvelle source de son.

- Source 3D

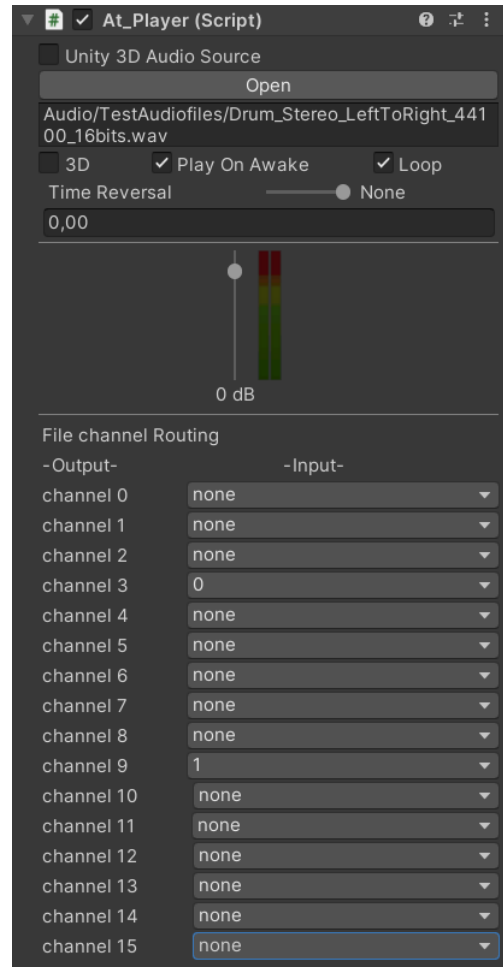
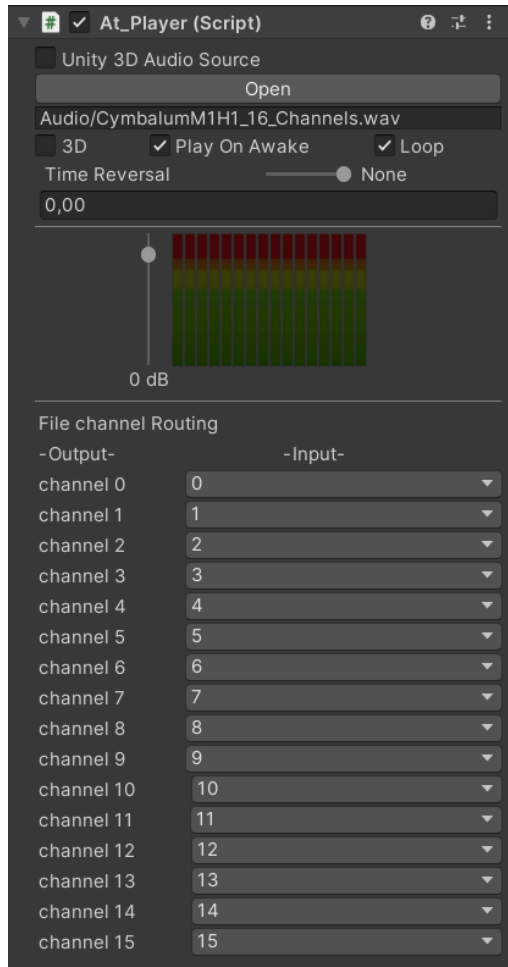


2. Aperçu de l'interface des objets

b) At_Player

On ajoute un component At_Player à un objet dans la scène 3D pour créer une nouvelle source de son.

- Source 2D



Si la source n'est pas « 3D », elle est « 2D ». Cela signifie que les échantillons du fichier audio ne sont pas traités par le moteur de spatialisation.

Dans ce cas les N canaux du fichiers sont assignés à l'un des M canaux du bus Master

- Dans l'image de gauche un fichier 16 canaux est lu avec une configuration de sortie 16 canaux. Le canal 0 est envoyé dans le canal 0 en sortie, le canal 1 dans le canal 1 etc.
- Dans l'image de droite un fichier 2 canaux est lu. Le canal 0 est envoyé dans le canal 3 et le canal 1 dans le canal 9. Aucun son ne sera audible sur les autres canaux (« none ») pour cette source.

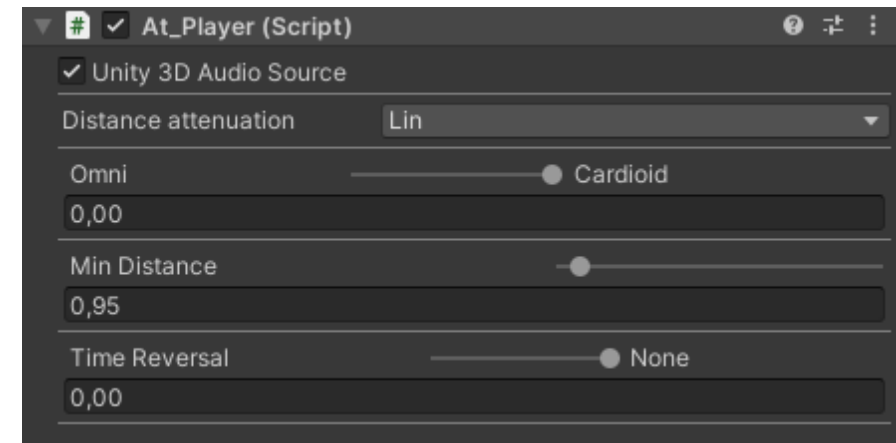
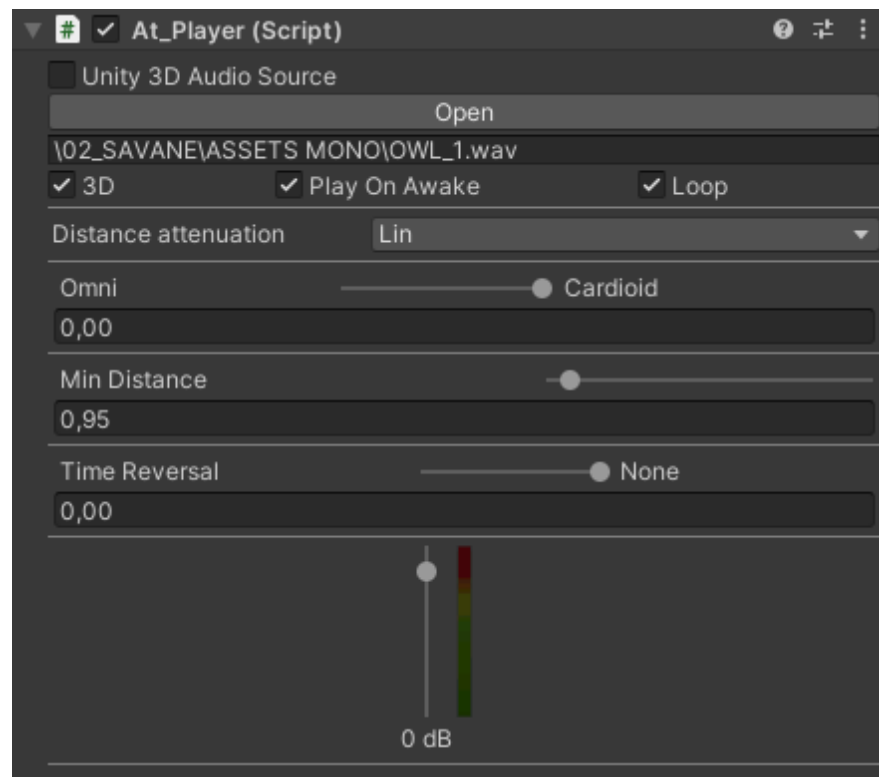
2. Aperçu de l'interface des objets

b) At_Player

On ajoute un component At_Player à un objet dans la scène 3D pour créer une nouvelle source de son.

- Unity 3D Audio Source et LibPD

Lorsque l'option « Unity 3D Audio Source » est cochée, le component At_Player se connecte au graphe audio de Unity grâce à la méthode « OnAudioFilterRead() »



2. Aperçu de l'interface des objets

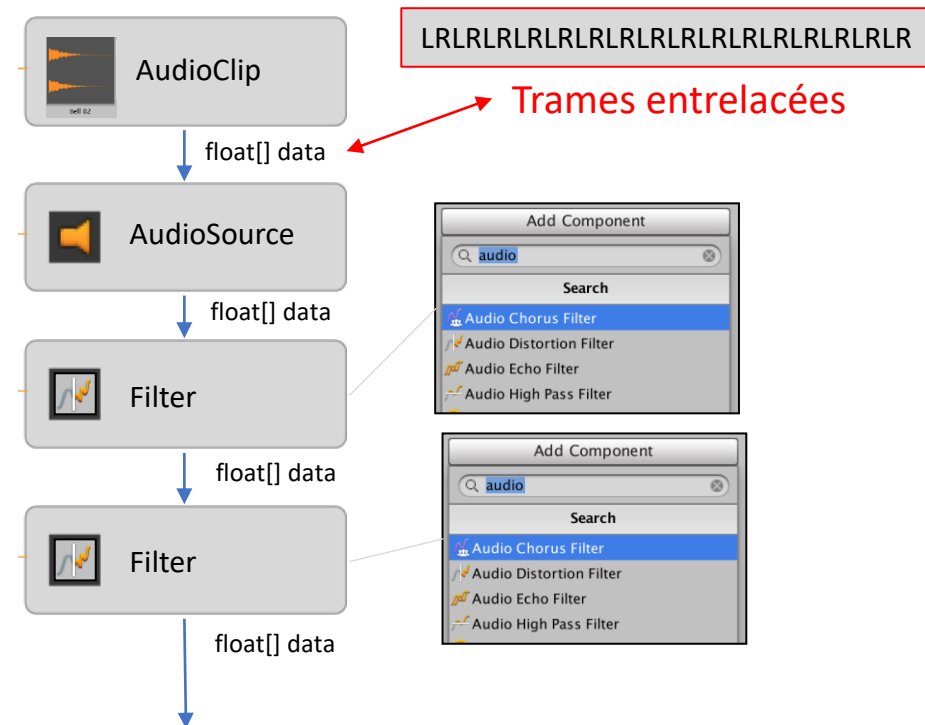
b) At_Player

On ajoute un component `At_Player` à un objet dans la scène 3D pour créer une nouvelle source de son.

- Unity 3D Audio Source et LibPD

<https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnAudioFilterRead.html>

```
private void OnAudioFilterRead(float[] data, int channels)
{
    for (int i = 0; i < data.Length; i += channels)
    {
        for (int j = 0; j < channels; j++)
        {
            // Signal Processing here....
        }
    }
}
```



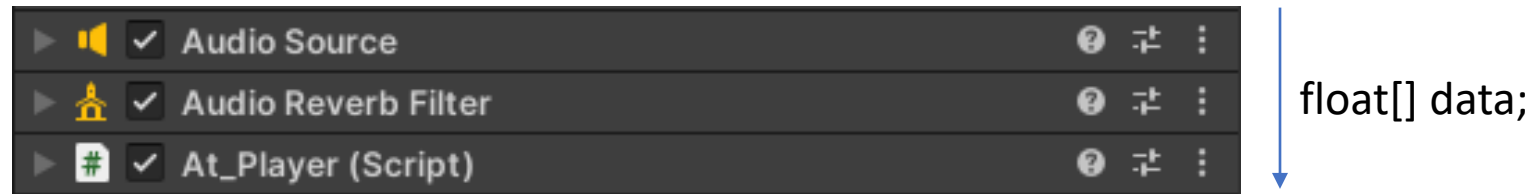
2. Aperçu de l'interface des objets

b) At_Player

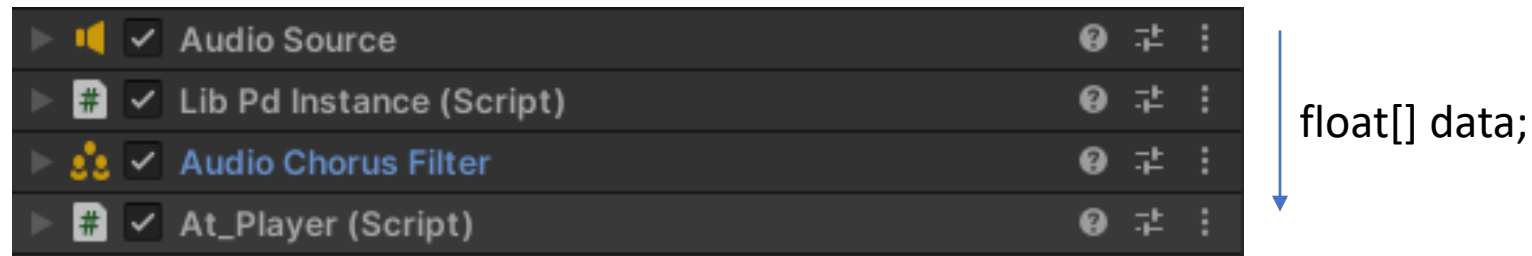
On ajoute un component At_Player à un objet dans la scène 3D pour créer une nouvelle source de son.

- Unity 3D Audio Source et LibPD

Il suffit d'ajouter le component AudioSource au GameObject contenant le component At_Player et de sélectionner un fichier audio pour le paramètre « Clip ». Cette fois-ci les fichiers audio ne doivent pas être dans le dossier « StreamingAssets ». Les paramètres relatifs à la lecture du fichier audio disparaissent du component At_Player puisqu'ils sont gérés par le component AudioSource.



Si l'on désire utiliser LibPD pour la génération audio en remplacement d'un clip. LibPdInstance doit être en dessous de AudioSource, les effets ensuite et At_Player en dernier.



2. Aperçu de l'interface des objets

b) At_Player

On ajoute un component At_Player à un objet dans la scène 3D pour créer une nouvelle source de son.

- Unity 3D Audio Source et LibPD

/!\ Le moteur audio de Unity et notre moteur audio 3D ne sont pas dans le même thread ET ne sont pas synchrones.

- Les fréquences d'échantillonnages peuvent être différentes
- La taille du buffer peut être différente
- Les appels aux méthodes « OnAudioFilterRead() » de Unity et « OnAsioOutAvailable() » de NAudio sont asynchrones.

Ainsi, pour copier les échantillons du buffer audio de Unity dans le buffer d'entrée du component At_Player, nous utilisons un buffer temporaire de taille :

$$\frac{10 \times \text{unityOutputBufferSize} \times \text{asioOutputBufferSize}}{\text{gcd}(\text{unityOutputBufferSize}, \text{asioOutputBufferSize})}$$

où $\text{gcd}()$ est la fonction qui calcule le plus grand dénominateur commun entre $\text{unityOutputBufferSize}$ et $\text{asioOutputBufferSize}$

/!\ La taille du buffer peut être très grande si le *pgdc* est proche de 1. Il convient de vérifier la taille du buffer de Unity et du driver ASIO du périphérique de sortie (petite taille et puissance de 2).

3. Architecture du moteur de spatialisation (dll C/C++)

```
extern "C" UNITY_AUDIODSP_EXPORT_API void AT_SPAT_CreateWfsSpatializer(int* id, bool is3D, bool isDirective, float maxDistanceForDelay)
{
    At_SpatializationEngine::getInstance().CreateWfsSpatializer(id, is3D, isDirective, maxDistanceForDelay);
}
//...
extern "C" UNITY_AUDIODSP_EXPORT_API void AT_SPAT_WFS_setVirtualMicPosition(int speakerCount, float virtualMicMinDistance, float*
positions, float* rotations, float* forwards)
{
    At_SpatializationEngine::getInstance().WFS_setVirtualMicPosition(speakerCount, virtualMicMinDistance, positions, rotations,
forwards);
}
//etc.
```

At_SpatializationEngine
Classe

Champs

- incrementalUniqueID
- m_pWfsSpatializerList

Méthodes

- ~At_SpatializationEngine
- At_SpatializationEngine
- CreateWfsSpatializer
- DestroyWfsSpatializer
- findSpatializerWithSpatID
- getInstance
- WFS_cleanDelayBuffer
- WFS_destroyAllSpatializer
- WFS_getDelay
- WFS_process
- WFS_setAttenuation
- WFS_setListenerPosition
- WFS_setMinDistance
- WFS_setSampleRate
- WFS_setSourceOmniBalance
- WFS_setSourcePosition
- WFS_setTimeReversal
- WFS_setVirtualMicPosition

At_WfsSpatializer
Classe

Champs

Méthodes

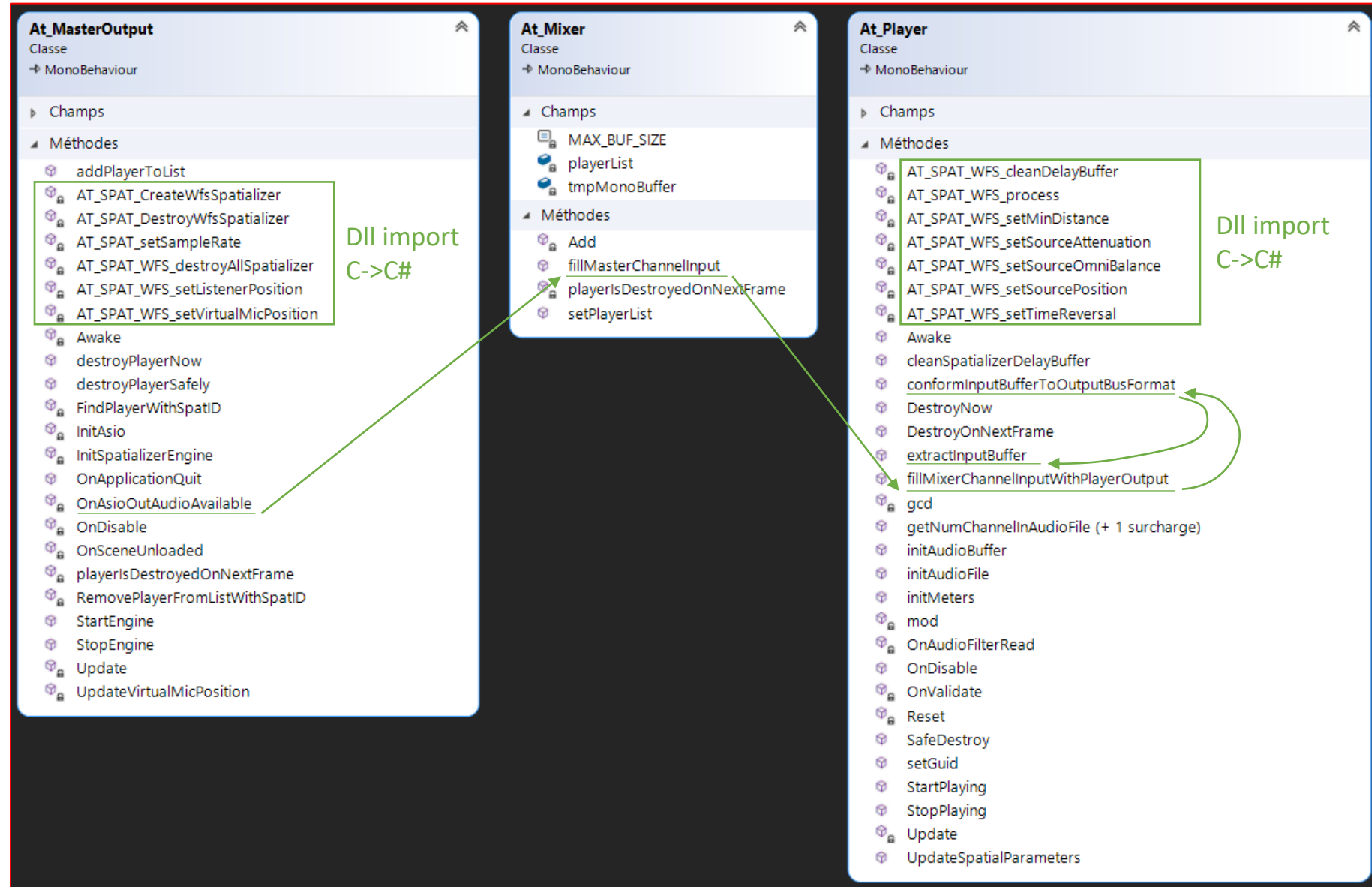
- ~At_WfsSpatializer
- applyWfsGainDelay
- cleanDelayBuffer
- forceMonoInput
- initDelayBuffer
- process
- setIs3DsDirective
- setListenerPosition
- setMinDistance
- setSampleRate
- setSourceAttenuation
- setSourceOmniBalance
- setSourcePosition
- setTimeReversal
- setVirtualMicPosition
- updateDelayBuffer
- updateMixedDirectiveChannel
- updateMixMaxDelay
- updateMultichannelDelayBuffer
- updateWfsVolumeAndDelay
- WFS_getDelay

Plugin_At_Spatializer.cpp
dll

4. Architecture de l'intégration Unity (C#)

Class List

- ▼ N NAudioAsioPatchBay
 - C AsioInputPatcher
 - C At_3DAudioEngineState
 - C At_AudioEngineUtils
 - C At_DynamicRandomPlayer
 - C At_DynamicRandomPlayerEditor
 - C At_DynamicRandomPlayerState
 - C At_ExternAssets
 - C At_ExternAssetsState
 - C At_MasterOutput
 - C At_MasterOutputEditor
 - C At_Mixer
 - C At_OutputState
 - C At_Player
 - C At_PlayerEditor
 - C At_PlayerState
 - C At_SpeakerConfig
 - C At_VirtualMic
 - C At_VirtualSpeaker
 - C At_VirtualSpeakerState
 - C ListenerDrawAndScale



4. Architecture de l'intégration Unity (C#)

Class List

- ▼ N NAudioAsioPatchBay
 - C AsioInputPatcher
- C At_3DAudioEngineState
- C At_AudioEngineUtils
- C At_DynamicRandomPlayer
- C At_DynamicRandomPlayerEditor
- C At_DynamicRandomPlayerState
- C At_ExternAssets
- C At_ExternAssetsState
- C At_MasterOutput
- C At_MasterOutputEditor
- C At_Mixer
- C At_OutputState
- C At_Player
- C At_PlayerEditor
- C At_PlayerState
- C At_SpeakerConfig
- C At_VirtualMic
- C At_VirtualSpeaker
- C At_VirtualSpeakerState
- C ListenerDrawAndScale

The screenshot displays two class inspectors side-by-side. The left inspector is for **At_MasterOutputEditor**, which is a class and an editor. It lists methods including OnEnable and OnInspectorGUI. The right inspector is for **At_PlayerEditor**, also a class and editor, listing methods including OnEnable and OnInspectorGUI.

OnEnable

Méthode utilisé pour l'initialisation de l'interface graphique du component

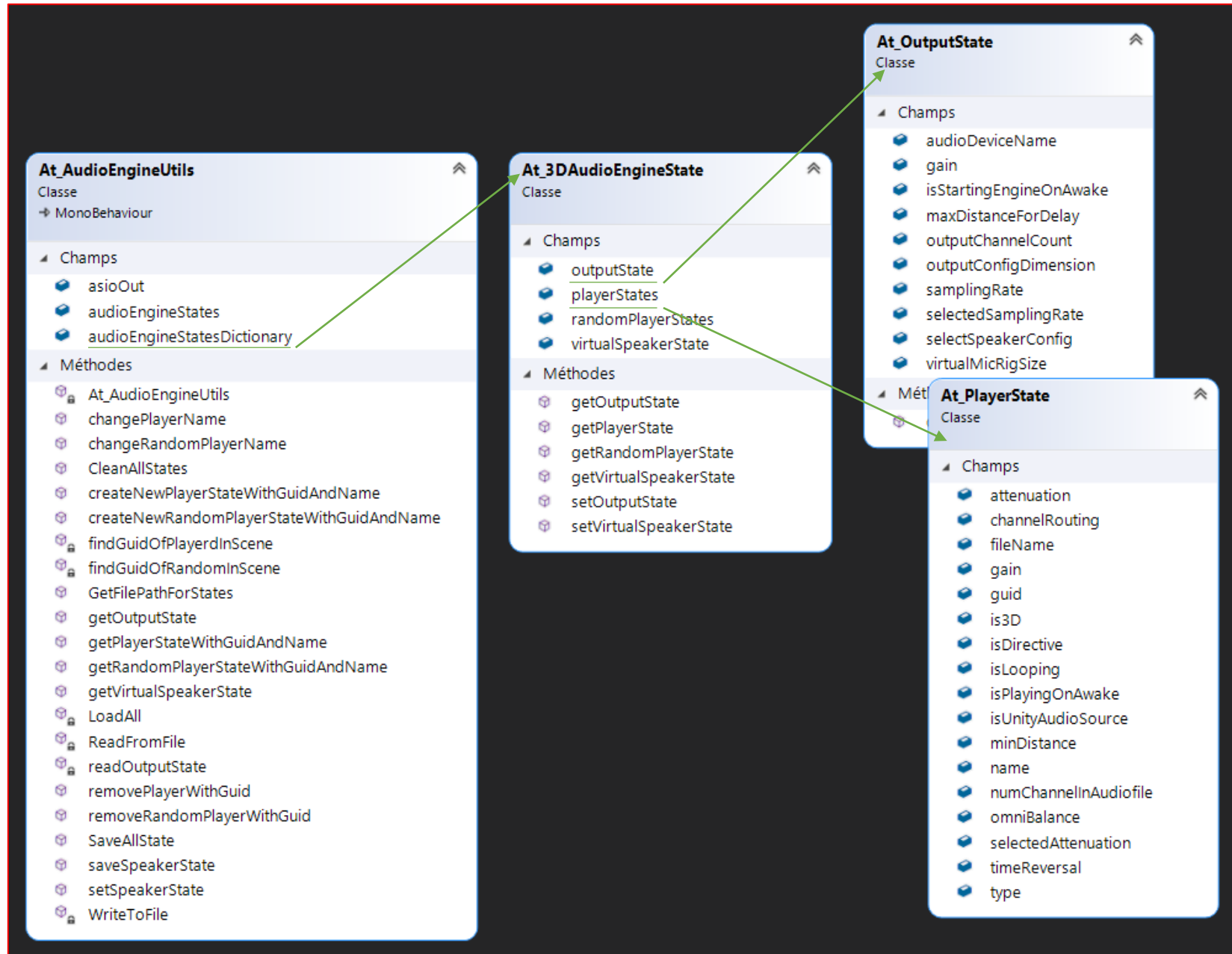
OnInspectorGUI

Méthode utilisé dessiner les éléments d'interface graphique

4. Architecture de l'intégration Unity (C#)

Class List

- ▼ N NAudioAsioPatchBay
 - C AsioInputPatcher
 - C At_3DAudioEngineState
 - C At_AudioEngineUtils
 - C At_DynamicRandomPlayer
 - C At_DynamicRandomPlayerEditor
 - C At_DynamicRandomPlayerState
 - C At_ExternAssets
 - C At_ExternAssetsState
 - C At_MasterOutput
 - C At_MasterOutputEditor
 - C At_Mixer
 - C At_OutputState
 - C At_Player
 - C At_PlayerEditor
 - C At_PlayerState
 - C At_SpeakerConfig
 - C At_VirtualMic
 - C At_VirtualSpeaker
 - C At_VirtualSpeakerState
 - C ListenerDrawAndScale



4. Architecture de l'intégration Unity (C#)

Fichier *json* pour la sérialisation des objets de la scène utilisant le moteur audio (« nom_scene.txt »)

Etat du
At_MasterOutput

```
{"audioDeviceName": "Voicemeeter Virtual ASIO", "outputChannelCount": 16, "gain": -0.9459457993507385, "selectSpeakerConfig": 7, "outputConfigDimension": 1, "selectedSamplingRate": 1, "samplingRate": 48000, "isStartingEngineOnAwake": true, "virtualMicRigSize": 9.0, "maxDistanceForDelay": 100.0}
```

Etat de tous les
At_Player
dans la scène

```
{"type": 0, "guid": "192f36c7-3bac-4d35-b099-b8ff12cbd66f", "name": "Sphere", "fileName": "Audio/TestAudiofiles/Drum_Stereo_LeftToRight_44100_16bits.wav", "gain": -0.6578946113586426, "is3D": false, "isDirective": false, "isPlayingOnAwake": true, "isLooping": true, "omniBalance": 0.0, "timeReversal": 0.0, "attenuation": 0.0, "selectedAttenuation": 0, "minDistance": 1.4414414167404175, "channelRouting": [2, 2, 2, 0, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2], "numChannelInAudiofile": 2, "isUnityAudioSource": false}
```

```
{"type": 0, "guid": "be1aa06f-6b7a-4e8a-bc8d-1a2894fc69d0", "name": "Cube", "fileName": "Audio/LION_1.wav", "gain": 0.0, "is3D": true, "isDirective": false, "isPlayingOnAwake": false, "isLooping": false, "omniBalance": 0.0, "timeReversal": 1.0, "attenuation": 1.0, "selectedAttenuation": 1, "minDistance": 2.9585800170898439, "channelRouting": [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], "numChannelInAudiofile": 1, "isUnityAudioSource": false}
```