FMOD pour les architectures faiblement couplées.

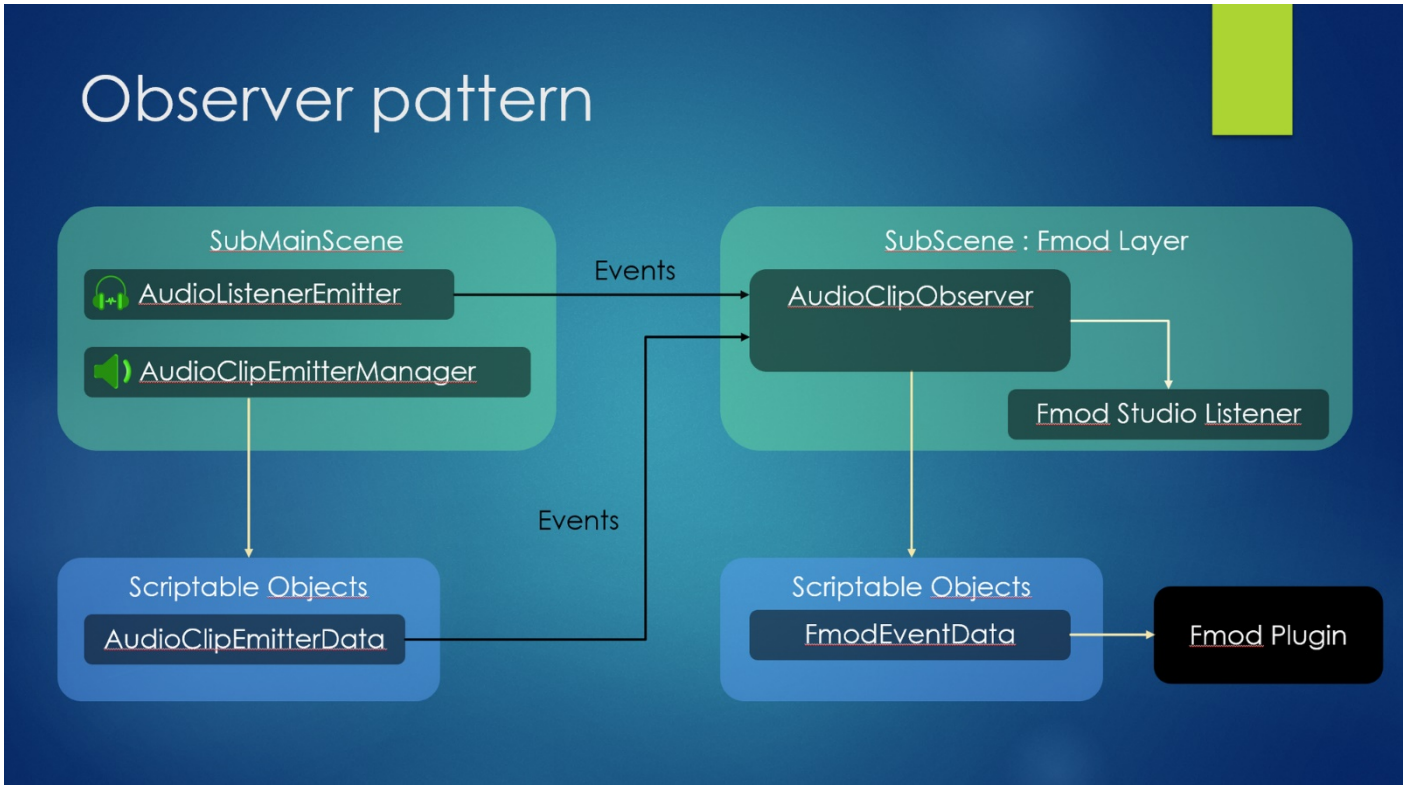
Le découplage permet le développement du jeu sans nécessiter l'installation préalable d'un moteur audio. Avec l'essor du télétravail et du travail à distance, cela facilitera considérablement votre travail en tant que développeur, car vous pourrez travailler en parallèle avec l'équipe dédiée au son. Ils pourront gérer l'aspect sonore en parallèle sans attendre votre validation pour l'intégration.

Autrice :
Aude Valfroy

Table des matières

Présentation de l’extension

1. Schéma Synthétique du fonctionnement :



2. Quelques explications :

Dans un projet d'architecture découplée ou avec plusieurs sous-scènes, où chaque sous-scène est dédiée à une fonctionnalité précise, l'idée principale est d'avoir une sous-scène dédiée uniquement au son (*que vous utilisiez FMOD, WWISE ou tout autre moteur sonore*). Ce "layer" (*sous-scène*) est chargé dès le lancement du jeu afin de fournir en permanence l'audio aux autres sous-scènes de votre jeu qui en dépendent.

Un événement dédié sert de moyen de communication entre le reste du jeu et cette sous-scène sonore.

Le découplage permet le développement du jeu sans nécessiter l'installation préalable d'un moteur audio. Avec l'essor du télétravail et du travail à distance, cela facilitera considérablement votre travail en tant que développeur, car vous pourrez travailler en parallèle avec l'équipe dédiée au son. Ils pourront gérer l'aspect sonore en parallèle sans attendre votre validation pour l'intégration.

Ce module comprend **deux parties distinctes** :

1. La première partie est consacrée au fonctionnement de la scène dédiée à la sonorisation de votre projet (*c'est dans cette partie que FMOD ou WWISE interviennent*).
2. La deuxième partie sert de lien de communication entre la scène sonore et le reste du jeu.

La scène appelée **FMODLayer** permettra l'utilisation de **composants spécifiques** pour placer des sons dans la scène. Cependant, il ne s'agira pas de composants liés au moteur audio comme le FMOD Event Emitter de FMOD. Les composants à utiliser seront les suivants :

- **AudioClipEmitterManager** : Ce composant doit être attaché à tous les éléments à sonoriser dans votre scène (*il remplace le FMOD Event Emitter*).
- **AudioListenerEmitter** : Ce composant doit être attaché à la caméra de votre projet ou au personnage du jeu afin d'entendre les sons que vous intégrez.

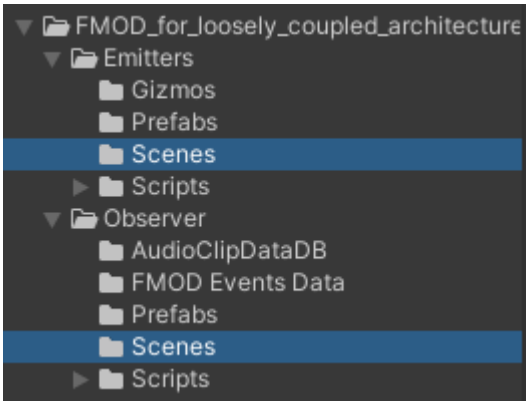
Ces deux composants serviront à envoyer des messages événementiels à destination du **FMODLayer**, qui les interprétera et les exécutera.

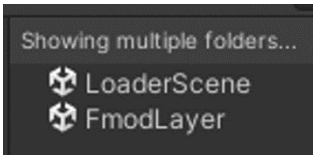
Configurer son projet pour intégrer l’extension :

Une fois que vous aurez téléchargé et importé l’extension, vous devriez avoir un dossier nommé **FMOD_for_lossely_coupled_architecture**.

1. Intégrer les 2 scènes à votre projet :

Il vous faudra intégrer ces deux **sous-scènes** dans votre projet :

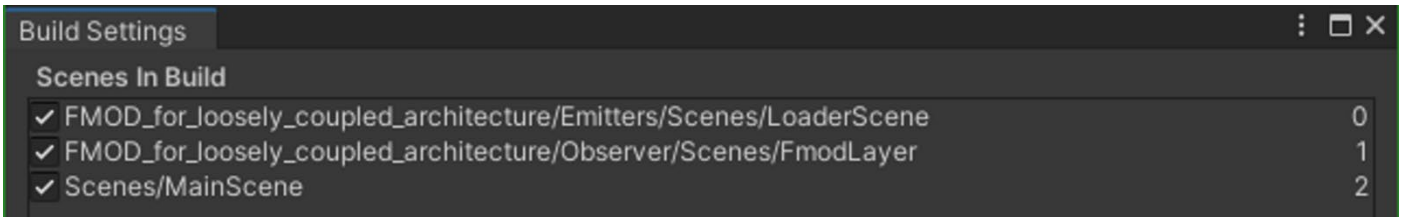




LoaderScene: Cette scène va permettre le lancement et la gestion des autres scènes en plus de la scène pour les audios. (**FMODLayer**).

FMODLayer: On crée la base de donnée (*lié à la banque FMOD Studio, la liste des évènements, des Bus et des VCA)*. Son rôle est d'exécuter des commandes comme les fameux "**PlayOneShot**" de FMOD via des scripts, on n'utilisera pas les **FMOD Event Emitter.** Donc, on y renseignera une liste d'objet scriptables qu'on aura créée (La *DataBase*).

Ajoutez **LoaderScene** et **FMODLayer** dans les options **Build Setting**, dans cet ordre :

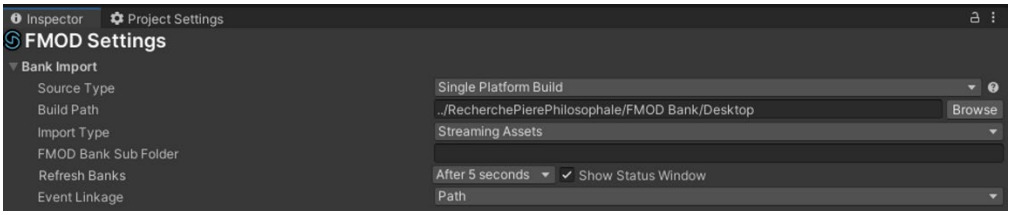


S'assurer que **LoaderScene** soit en haut de la liste, c'est lui qui chargera les autres scènes du projet.

2. Configuration de FMOD Plugin :

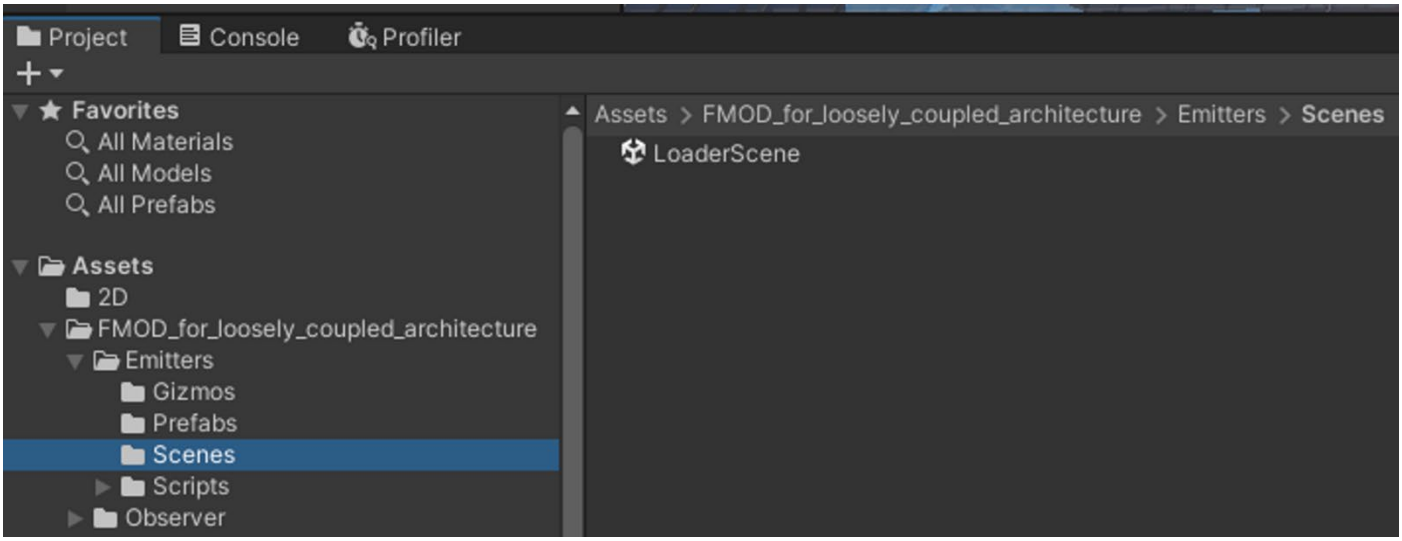
Le plugin FMOD dans votre projet doit être configuré sur l'option "**Single Plateform Build**" ou "**Multiple Plateforme Build**". On se sert du cache du plugin pour créer notre base de données. En mode Streaming, le cache ne sera pas réalisé. Dans le cadre d'un travail en équipe cette solution est la plus recommandé.

(*Pour plus d'informations sur l'utilisation de ces paramètres merci de vous rendre sur la documentation de FMOD disponible à cette adresse. ⇒* https://www.fmod.com/docs/2.00/unity/user-guide.html#accessing-your-fmod-studio-content)

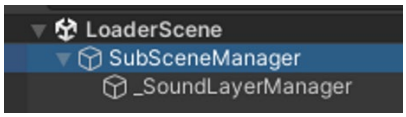


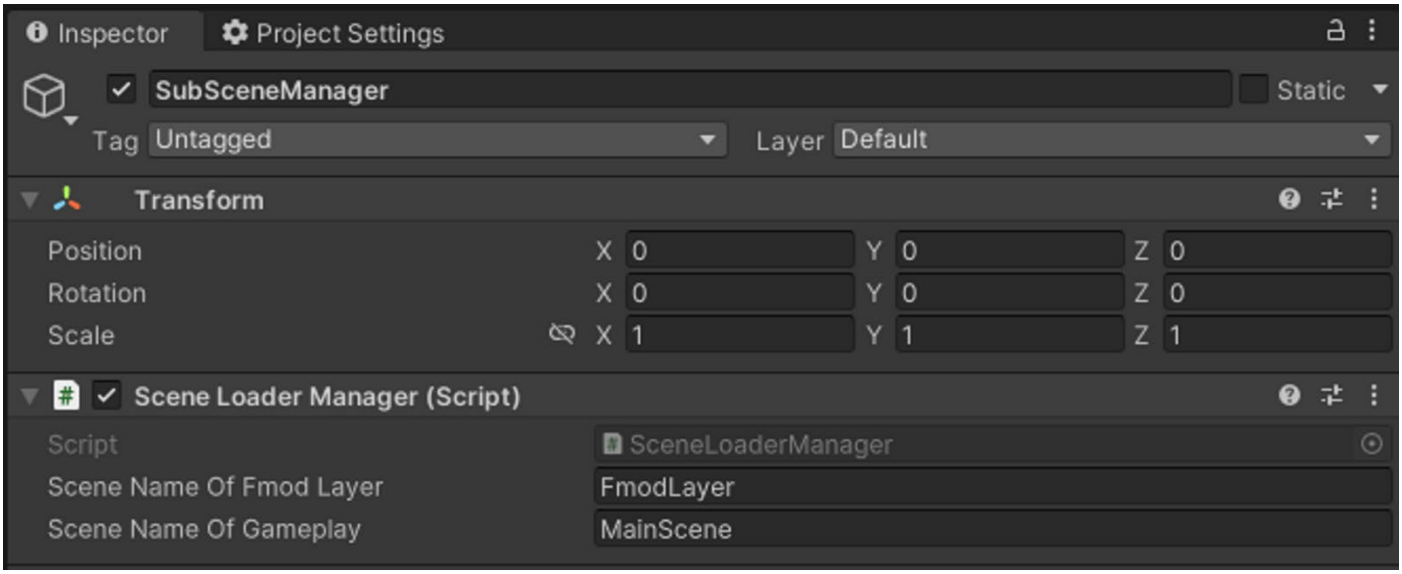
3. <u>Configuration des scènes</u> :

On charge la scène **LoaderScene**:



Une fois chargé, sélectionnez, dans la partie **Hierarchy**, le **GameComponent SubSceneManager:**



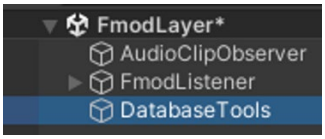


Dans la partie **inspector:**

- A la ligne **Scene Name Of FMOD Layer** il doit y avoi d'indiqué le nom de la **scène FMODLayer,** qui correspond à la scène possèdant les **AudioClipEmitter** et qui vous permettra d'entendre l'audio.
- La ligne **Scene Name Of Gameplay** devra être renseignée afin d'y inscrire le nom de la **scène principale du projet**. (*Partie Gameplay*)

Chargez ensuite la scène **FMODLayer:**

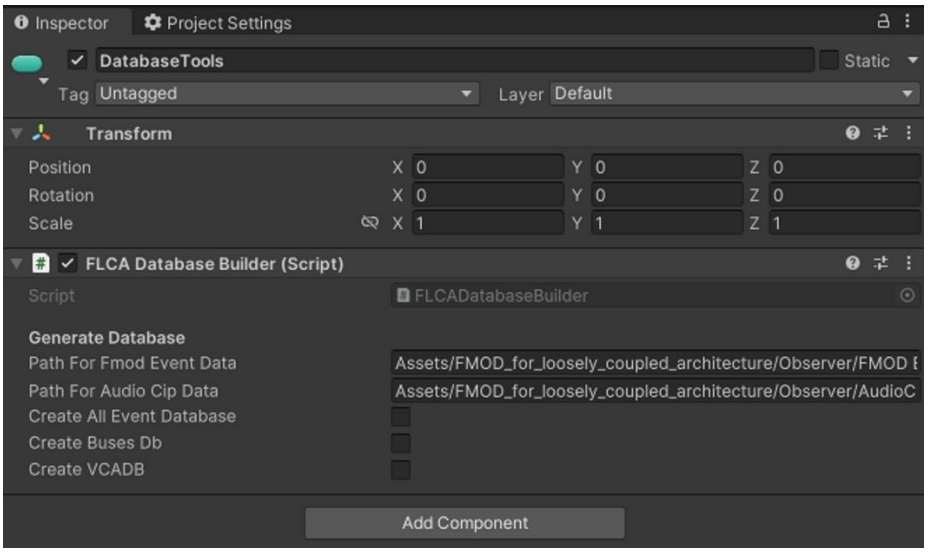
Sélectionnez, dans la scène, **DatabaseTools:**



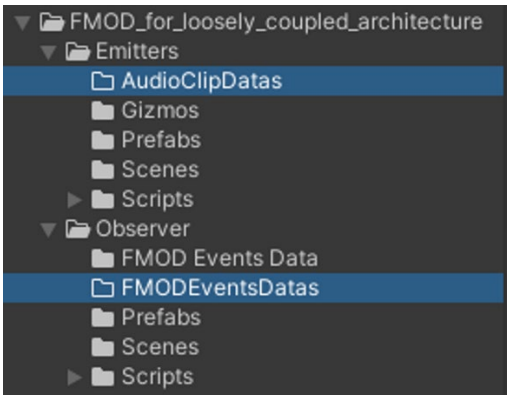
Renseignez, dans la partie **inspector** à la ligne :

- **Path For FMOD Event Data** ⇒ Le dossier dans lequel se retrouvera la base de données liées aux appels vers FMOD (*côtés Sound Designer*).
  Dossier par défaut :
  Assets/FMOD_for_loosely_coupled_architecture/Observer/FMODEventsData

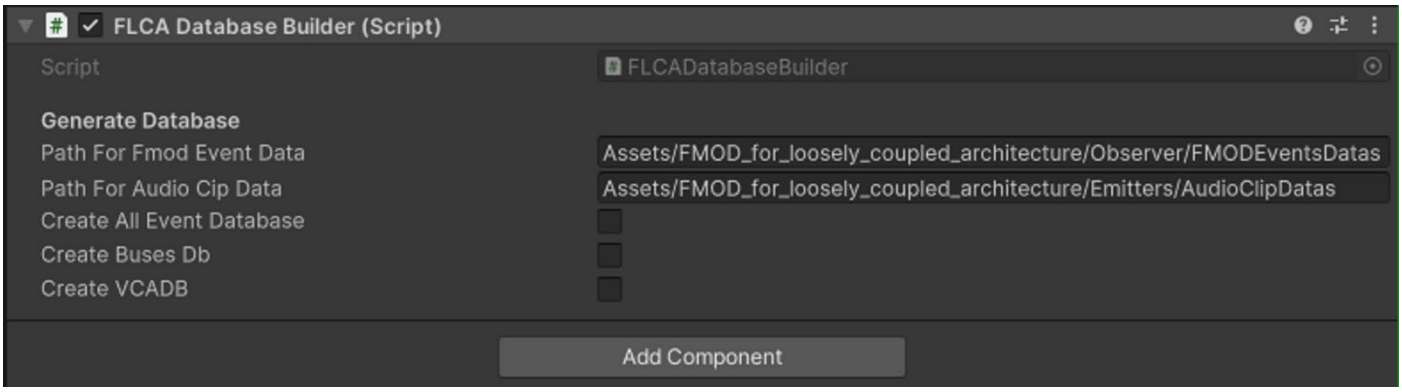
- **Path For Audio Clip Data** ⇒ Le dossier dans lequel se retrouvera la base de données liée au **AudioClipEmitterData** qui sont les éléments posés dans la scène. (*Côtés développpeur / Programmer Audio*)
  Dossier par défaut :
  Assets/FMOD_for_loosely_coupled_architecture/Emitters/AudioClipData



Dans notre exemple nous allons créer 2 dossiers. Un dossier pour le côté développeur (***AudioClipData***) et un autre dossier pour le côté Sound Designer (***FMODEventsData***).



Donc, nous allons renseigner le chemin d'accès à ces 2 dossiers au bon endroit :



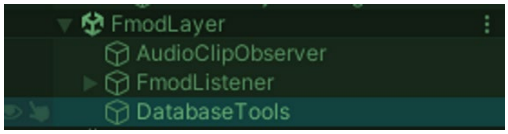
Une fois que ces deux chemins sont renseignés, sauvegardez vos changements.

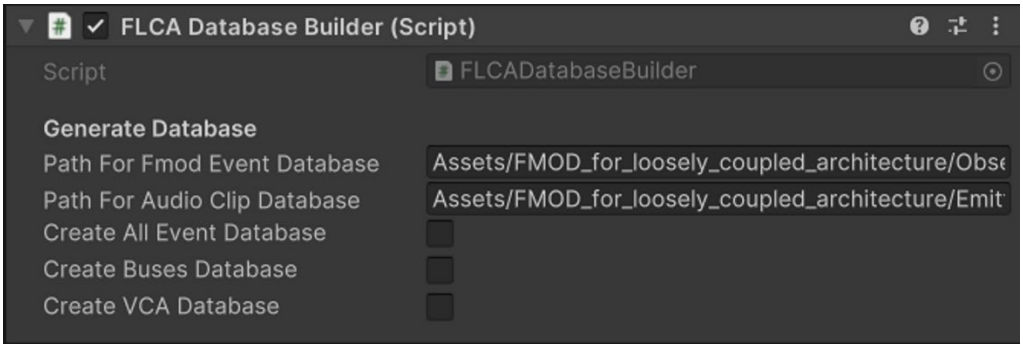
Créer la base de données sonores du projet :

1. Créer la base de données :

Chargez la scène **LoaderScene**, puis lancez un “**PlayRuntime**” dans l’éditeur d’Unity. Vous devriez obtenir une scène comportant plusieurs sous-scènes :

Sélectionnez dans la sous-scène **FMODLayer**, le GameComponent **DatabaseTools** vous devriez avoir ce menu à droite dans l'**inspector** :





Maintenant, cochez les cases correspondantes à la **Base de données** que vous souhaitez créer :

1. **Create All Event Database:** va créer les objets scriptables de l'ensemble des éléments liés aux **events** de **FMOD**, ainsi que les **snapshots***. (*Je vous renvoie à la documentation autour des Snapshot de FMOD Studio 2.02 ⇒ https://www.fmod.com/docs/2.02/studio/mixing.html#snapshots-and-the-tracks-view*)

2. **Create Buses Database** : va créer la liste des bus que nous avons créées dans **FMOD Studio**. Si vous souhaitez agir directement sur un bus plutôt qu'un VCA pour votre projet, vous le pourrez.

3. **Create VCA Database** : va créer la liste des VCA, souvent utilisé pour le menu des Volumes dans les options du projet.

Lorsque vous cocherez la case **Create All Events Database**, Unity se figera un instant, le temps de créer de tous les éléments. Suivant la quantité d'événements réalisés dans FMOD Studio et la puissance de votre ordinateur, cela peut prendre un petit instant. (*Comptez en moyenne 30sec à 2min*)

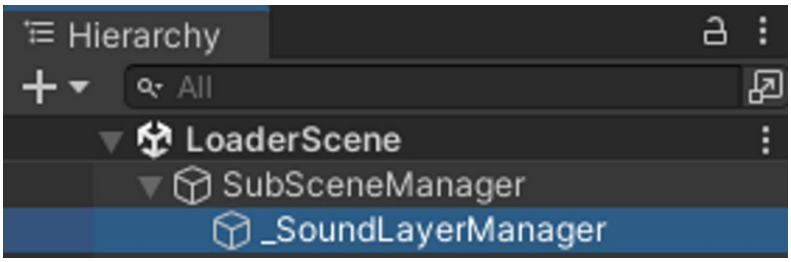
La case restera décochée, ce qui est normal !

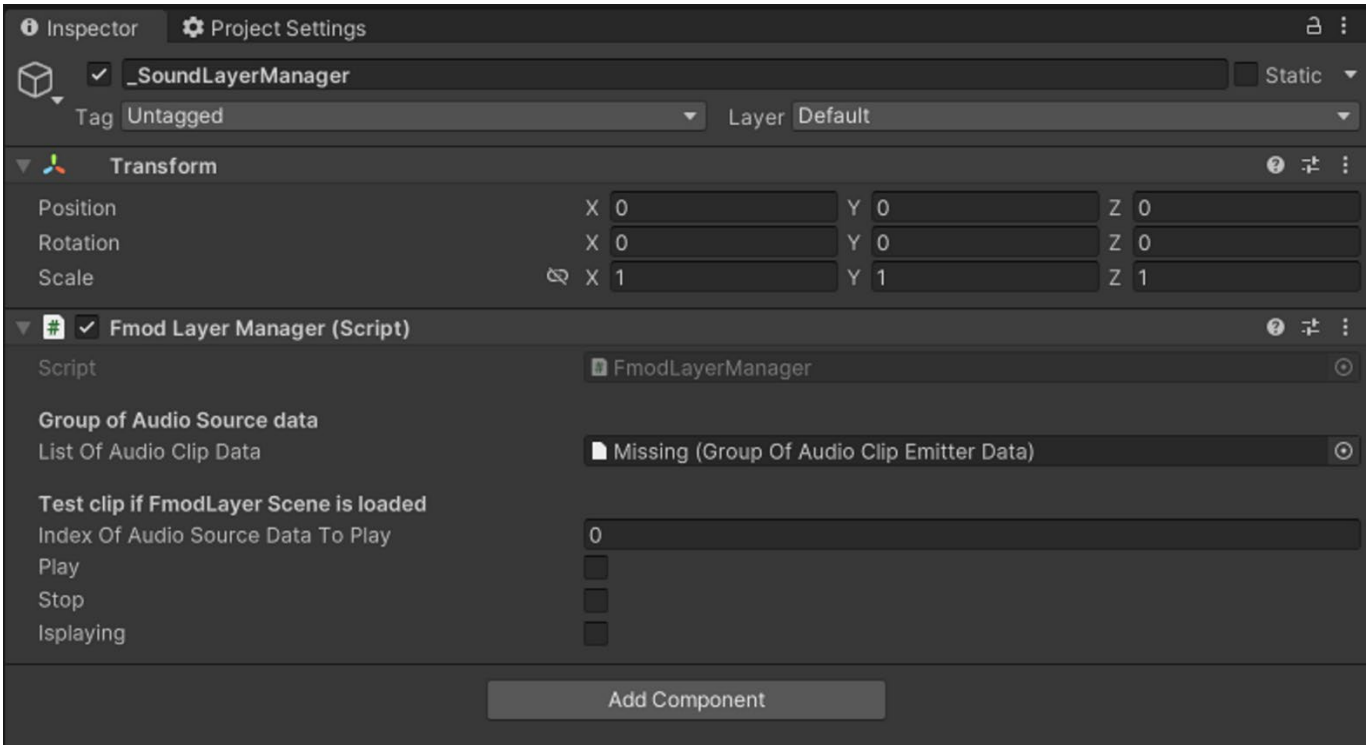
Une fois la base de données créées on peut stopper le **mode Play Runtime** de l'éditeur.

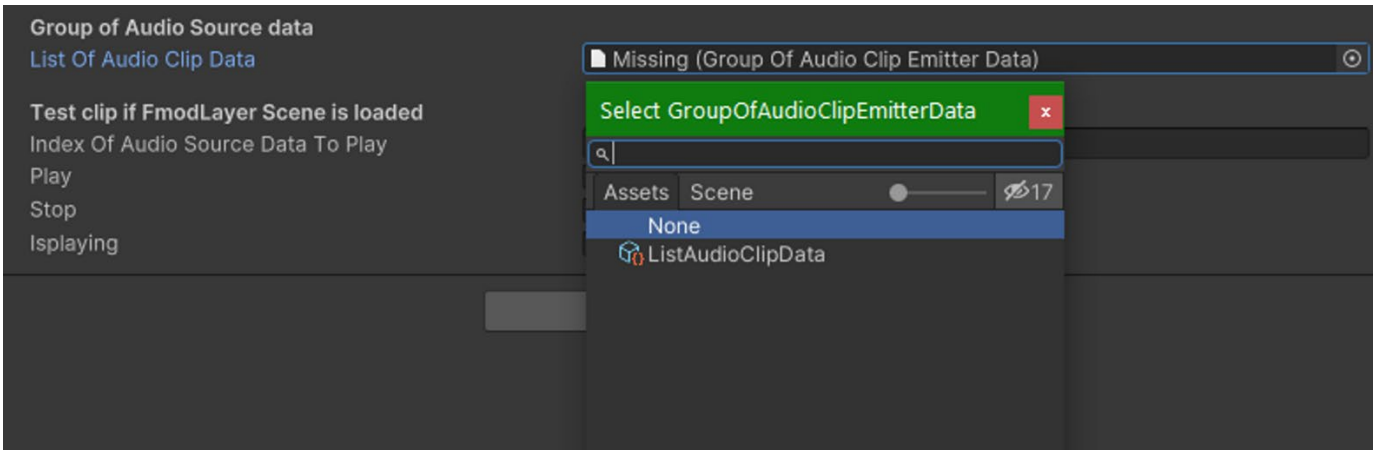
2. Paramétrer les scènes :

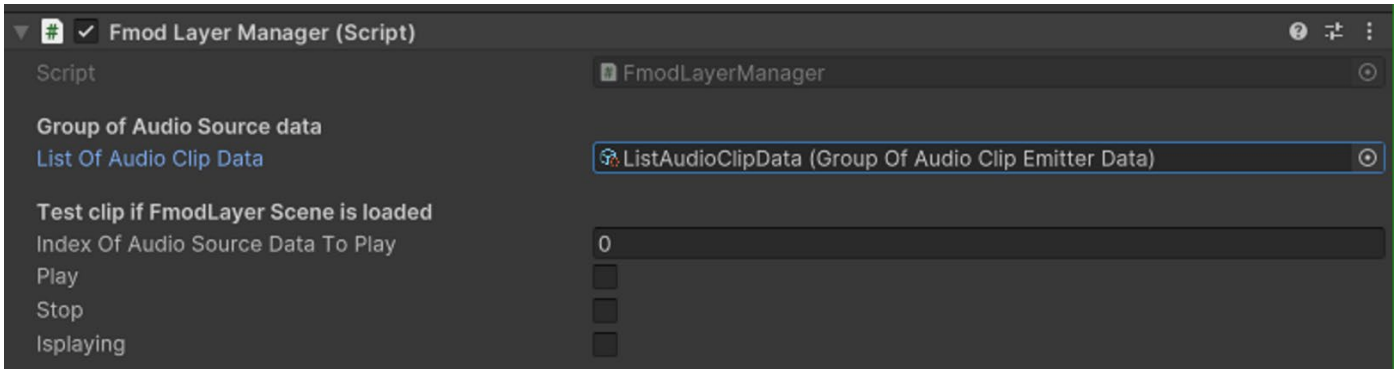
L'étape suivante va consister à renseigner aux bons endroits de nos scène, la base créée, pour créer les liens de nos deux scènes **FMODLayer** et **LoaderScene.**

1. Chargez de nouveau la scène **LoaderScene** (*si ce n'est pas déjà fait*), puis, sélectionnez **_SoundLayerManager** afin d'afficher dans la partie **inspector** les paramètres qui nous intéressent. A la ligne **List of Audio Clip Data**, sélectionnez la database fraichement créée. Aucune possibilité de se tromper.





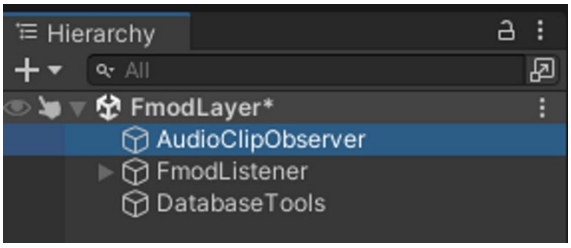


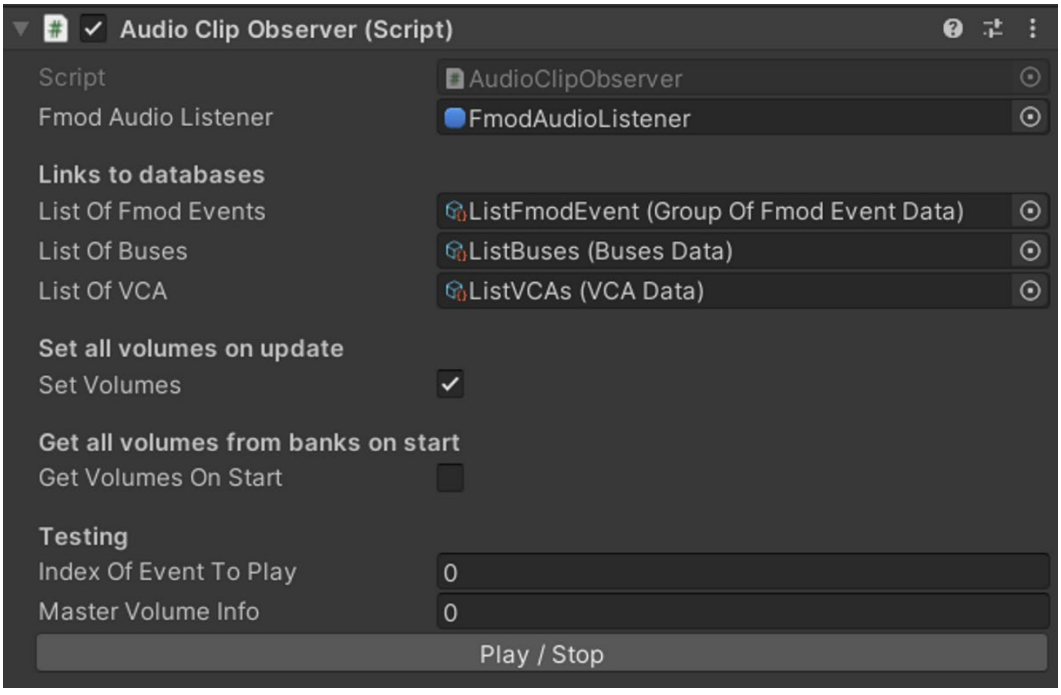


Sauvegardez la scène une fois les changements appliqués.

2. Ensuite, chargez la scène **FMODLayer**, puis sélectionnez **AudioClipObserver**, dans la partie **inspector** vous devrez renseigner 3 éléments :

- **List Of Fmod Events**
- **List Of Buses**
- **List Of VCA**





Pensez à sauvegarder après avoir fait les changements.

Vous avez terminé de configurer les 2 scènes principales vous permettant l'écoute et le contrôle de l'extension pour gérer l'audio de votre projet.

3. Informations complémentaires !

Attention : Veuillez noter que ces manipulations seront à refaire si vous supprimez la base de données des deux dossiers, ou une partie de celle-ci. (*Suite à certains changements, ou un renommage de vos éléments*).

Attention² : **Tous les changements de nom pour les events, les paramètres, les Bus et les VCA, amèneront obligatoirement à une suppression manuelle du contenue des deux dossiers contenant la base de données afin de la récréer proprement. A vous de bien gérer votre architecture dès le début.**

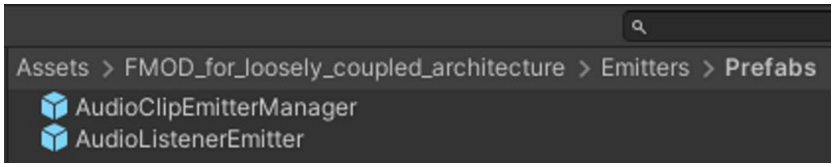
(*Une amélioration de cet aspect afin d'éviter la suppression manuelle est en cours de développement*)

Utilisation concrète de l'extension :

1. Comment utiliser l'AudioClipEmitterManager :

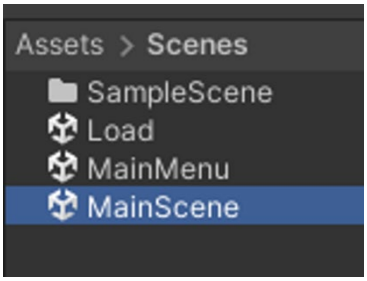
Si vous connaissez déjà le **plugin FMOD**, vous connaissez déjà l'utilisation de celui-ci avec le component **FMOD Event Emitter.** A chaque fois qu'on a besoin de déclencher un son dans la scène, on utilise habituellement **FMOD Event Emitter** sur un **GameObject** de la scène.

Vous allez donc remplacer l'utilisation de **FMOD Event Emitter** par **AudioClipEmitterManager**. (*Soit en tant que GameComponent avec le prefab soit en tant que component tout court, dépendant de la méthode de déclenchement que l'on veut réaliser)* :



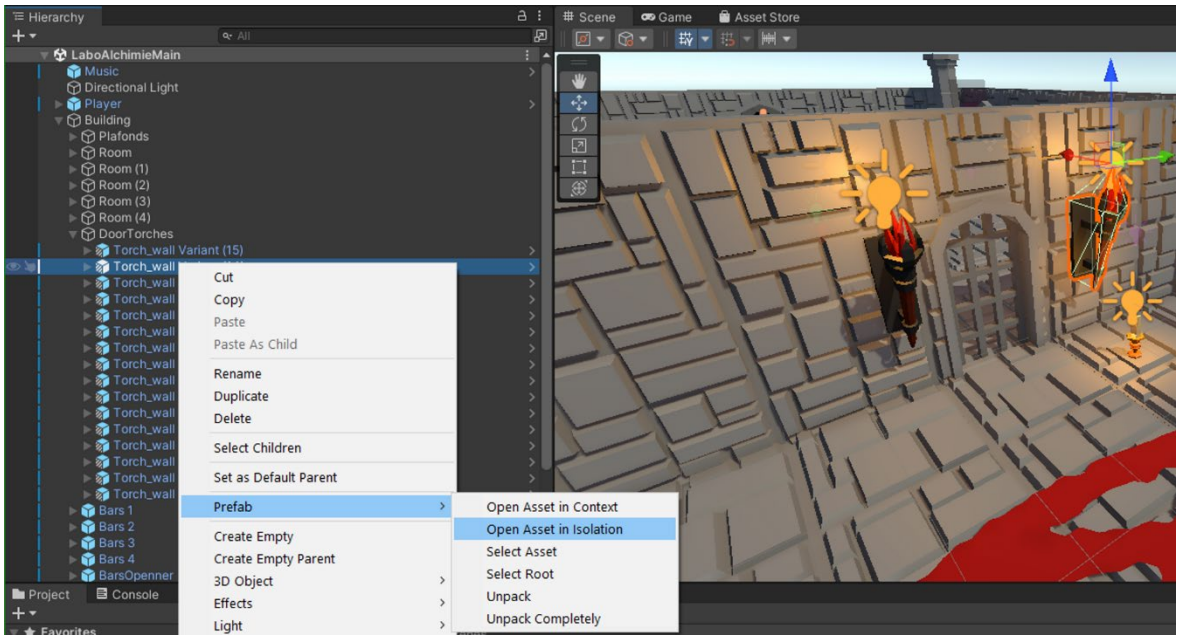
Dans notre exemple, nous allons poser notre **AudioClipEmitterManager** sur les éléments qu'on souhaite sonoriser, ici, les **torches** de cette scène :

Pour pouvoir placer nos audios dans notre scène de jeu, nous allons charger la scène principale :

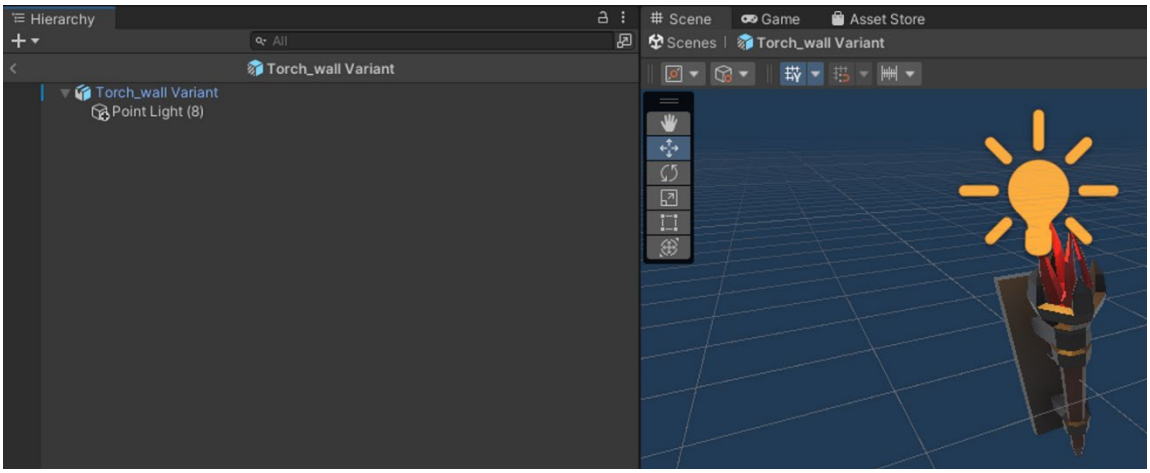


Ici ma scène principale s'appelle **MainScene**, c'est la scène où se trouve l'entièreté du projet.

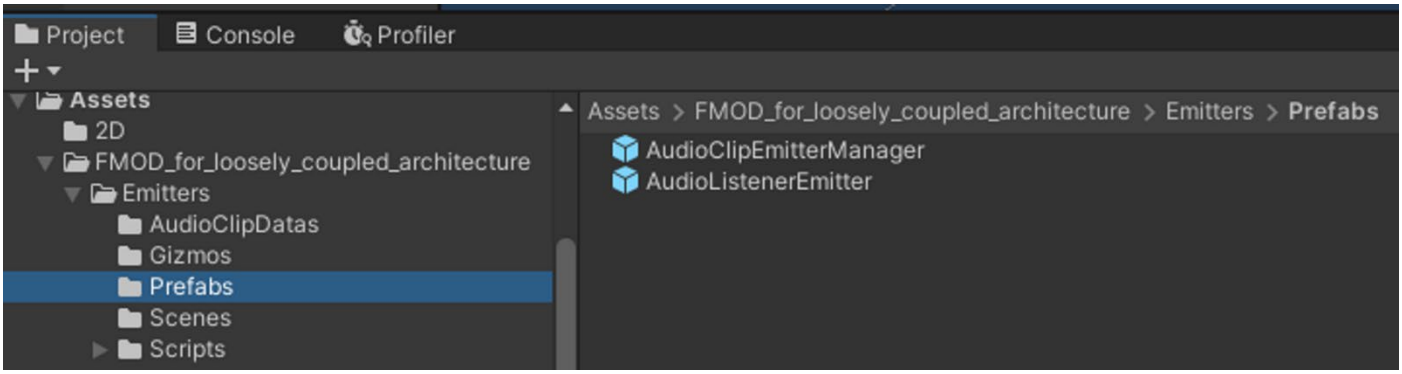
En sélectionnant une des **torches sur le mur**, je choisie d'ouvrir le **prefab parent** qui lui est associé. Afin que mes modifications s'appliquent sur **l'ensemble des torches** de la scène.

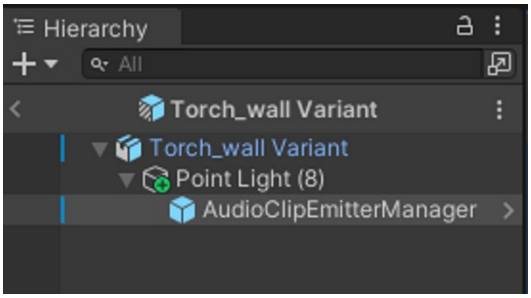


En ouvrant le **prefab**:

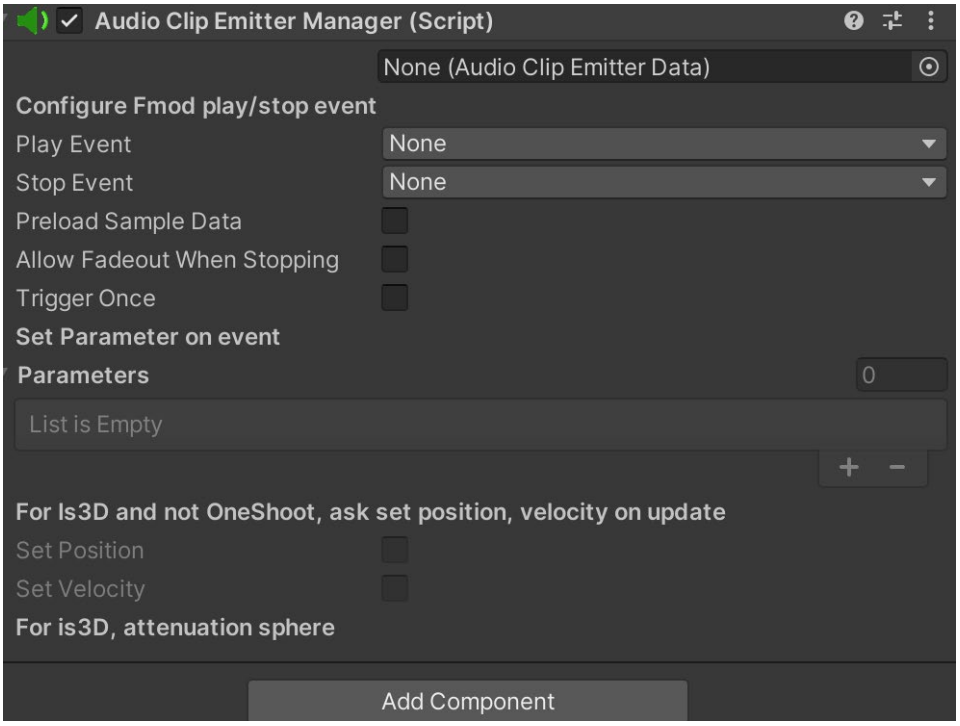


Je dépose mon prefab **AudioClipEmitterManager** :



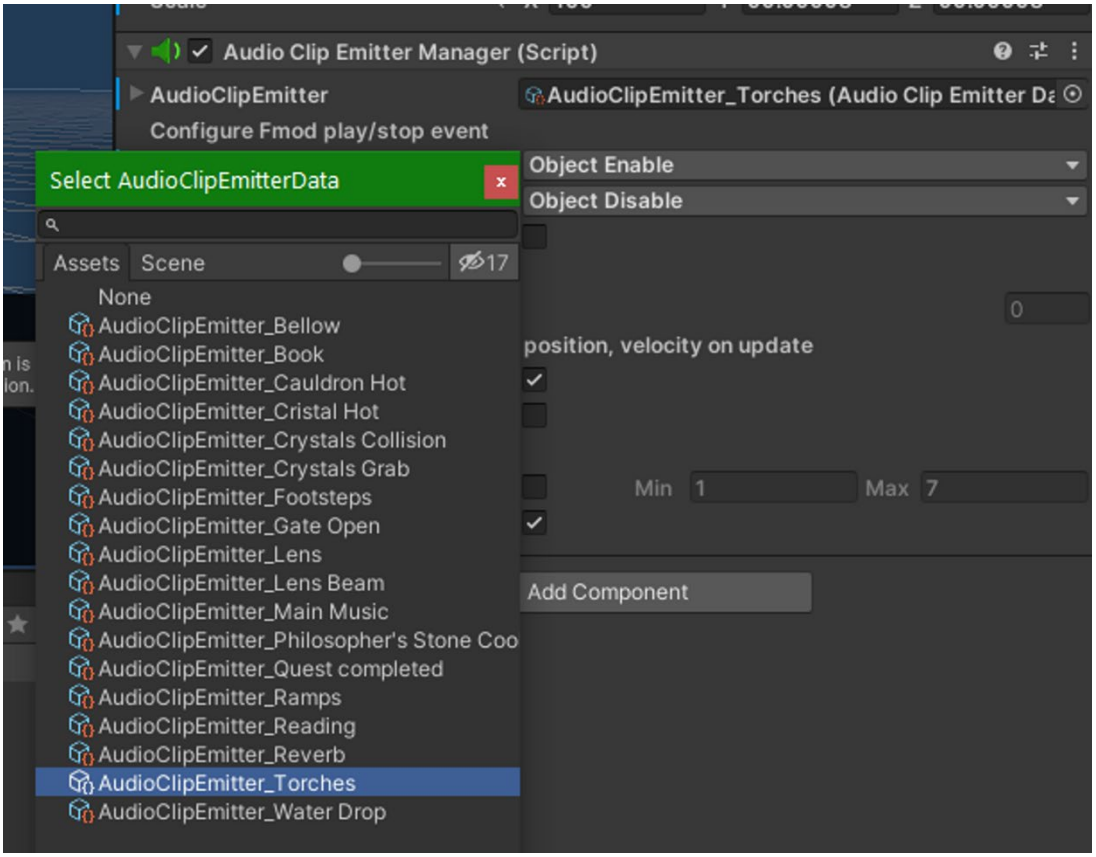


Ensuite, je sélectionne sur le GameComponent **AudioClipEmitterManager** afin d'avoir quelques paramètres à renseigner dans l'**inspector** :

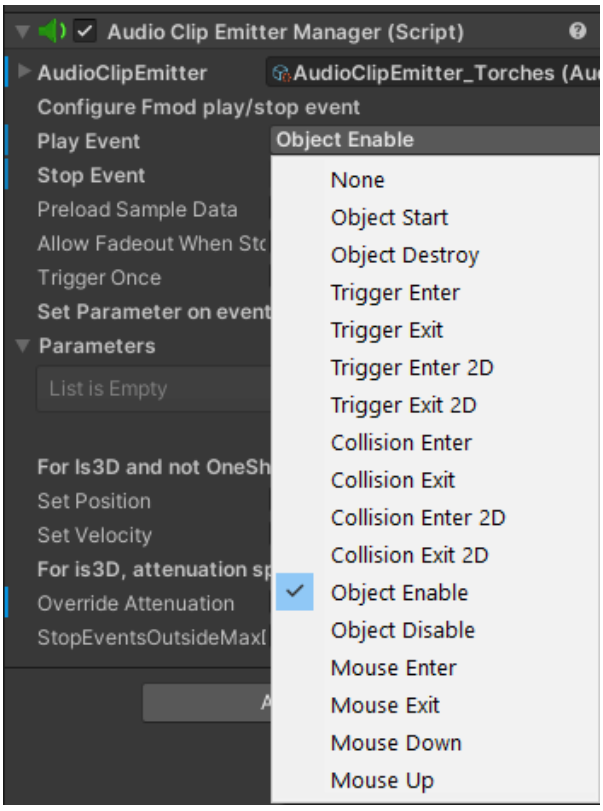


Nous retrouvons un simili d'un **FMOD Event Emitter**, que nous pouvons paramétrer comme d'habitude.

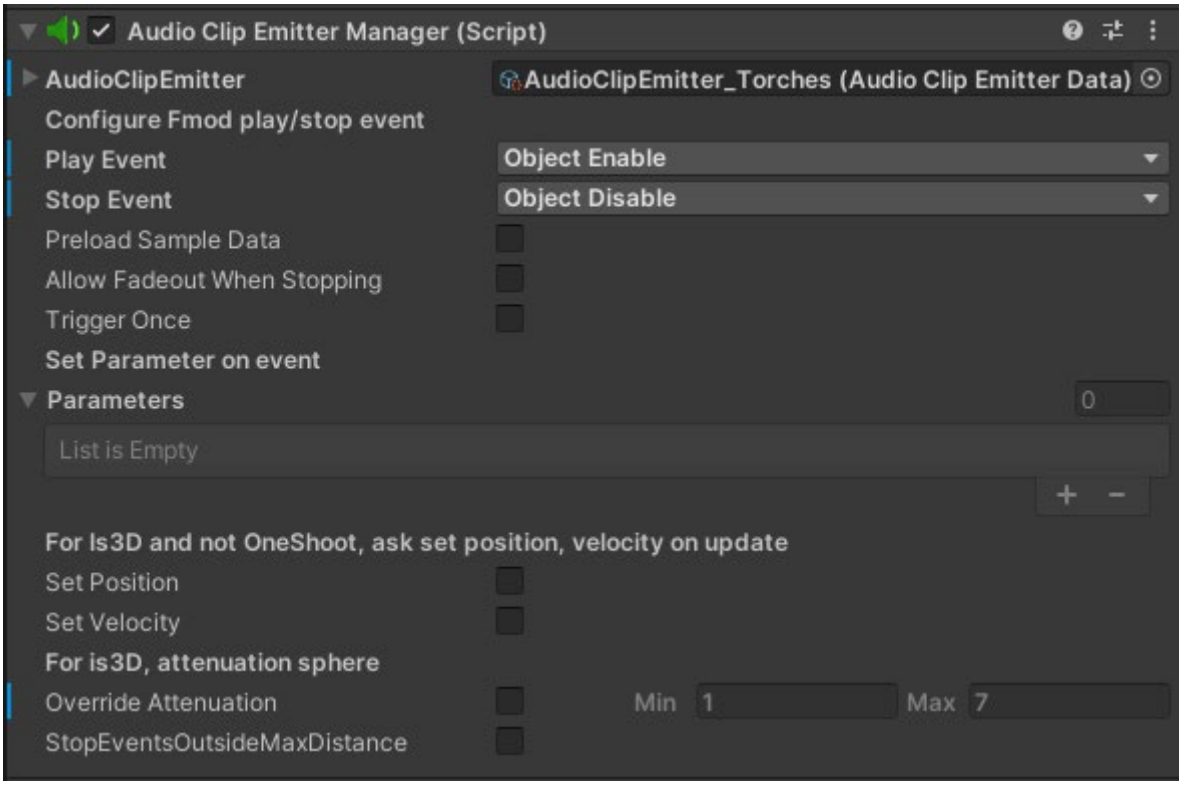
Nous renseignons alors l'**event Torches :**



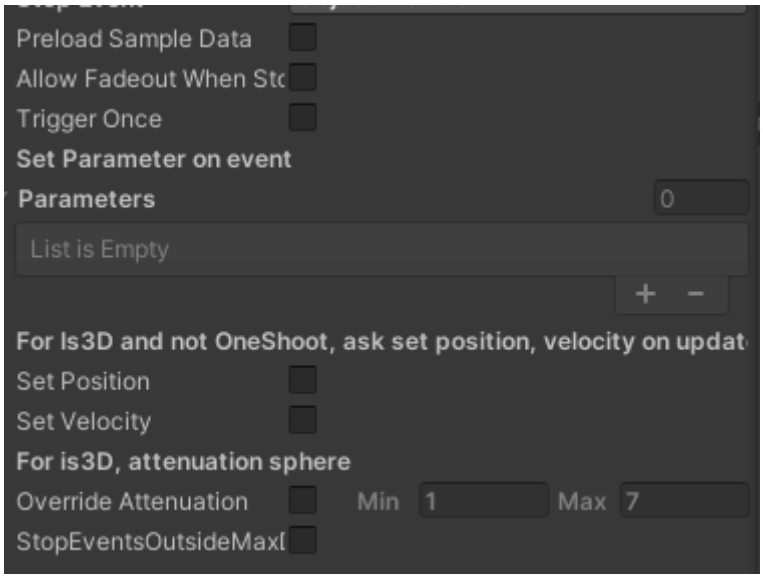
Ensuite, comme pour un **FMOD Event Emitter** classique, nous choisissons les options qui nous intéresse selon la méthode de déclenchement que nous recherchons :



L'**AudioClipEmitterManager** étant relié comme "enfant" du GameComponent **Point Light,** nous allons faire en sorte que l'audio se fasse entendre que si la lumière (*Point light*) est allumée. Le **statut de l'objet** dans l'**inspector** définira si la lumière sera allumée ou éteinte. Donc, je choisie comme paramètre de déclenchement **Object Enable** et en arrêt **Object Disable.** Ce qui aura pour effet de faire jouer le son quand la torche est active (*Object Enable*) et si j'interagie avec celle-ci, je vais l'éteindre, donc, le son se coupera en même temps (*Object Disable*).



Les cases à cocher en dessous, sont liés à différentes options utiles en fonction de la nature de votre son à intégrer :



- **Preload Sample Data** : A la même fonctionnalité que l'option du même nom dans les options du plugin FMOD. Ici, on peut le faire de manière unitaire plutôt que de l'avoir pour l'entièreté des audios.

- **Allow Fadeout When Stopping** : Permet d'activer la possibilité de prendre en compte l'effet ADHSR fait dans FMOD Studio, sur l'event sélectionné. S'il n'est pas présent, n'a aucun effet.

- **Trigger Once** : Permet de lancer l'événement qu'une seule fois par scène.

- **Set Position** : Si cette case est cochée, la position de la source sonore suivra l'objet auquel il est rattaché. (*A ne pas cocher pour les éléments fixes comme les torches*)

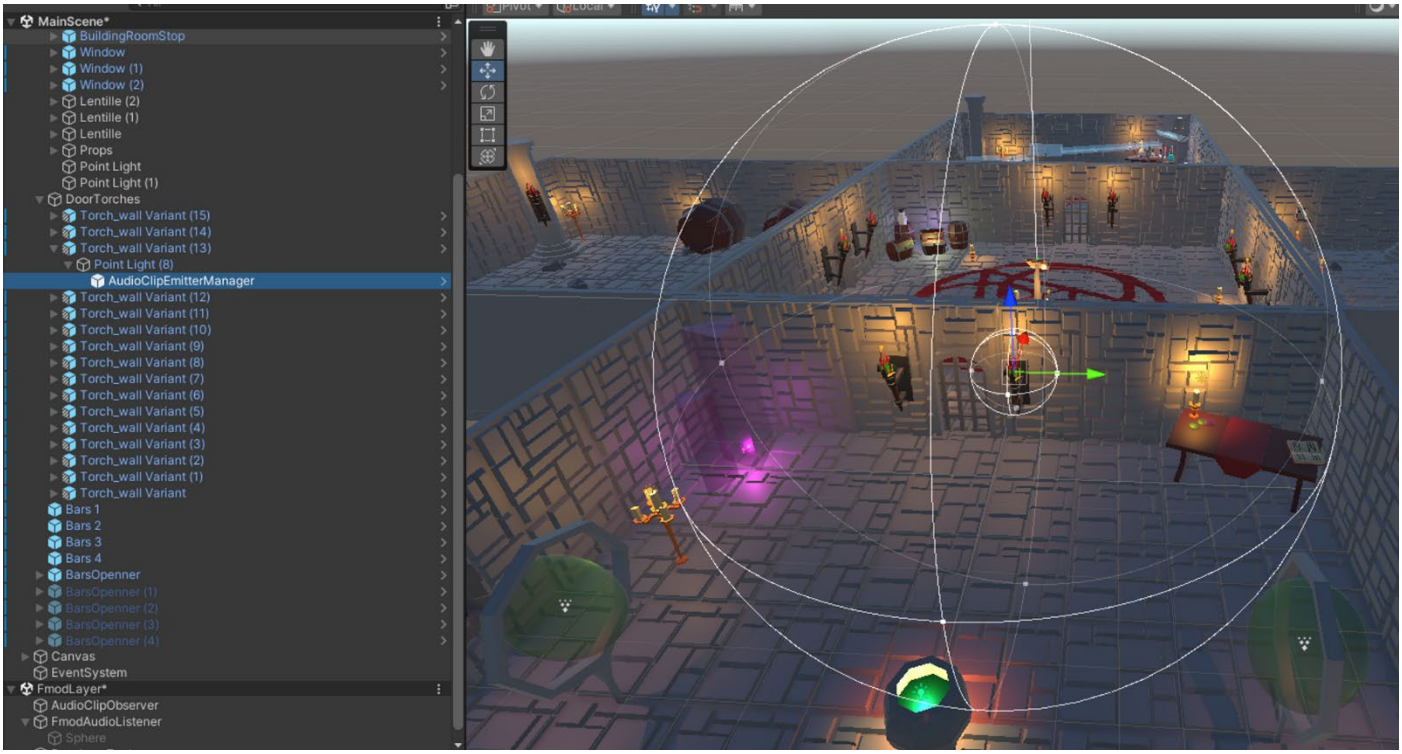
- **Set Velocity** : Si cette case est cochée, cela envoie les informations de la **vitesse de déplacement** de l'objet à l'event FMOD. (*Nécessite un **RigidBody** en "**parent**" pour fonctionner*).
  Cas de figure : Footsteps Player.

- **Override Attenuation** : Permet d'appliquer une sphère d'atténuation avec d'autres valeurs minimales / maximales.

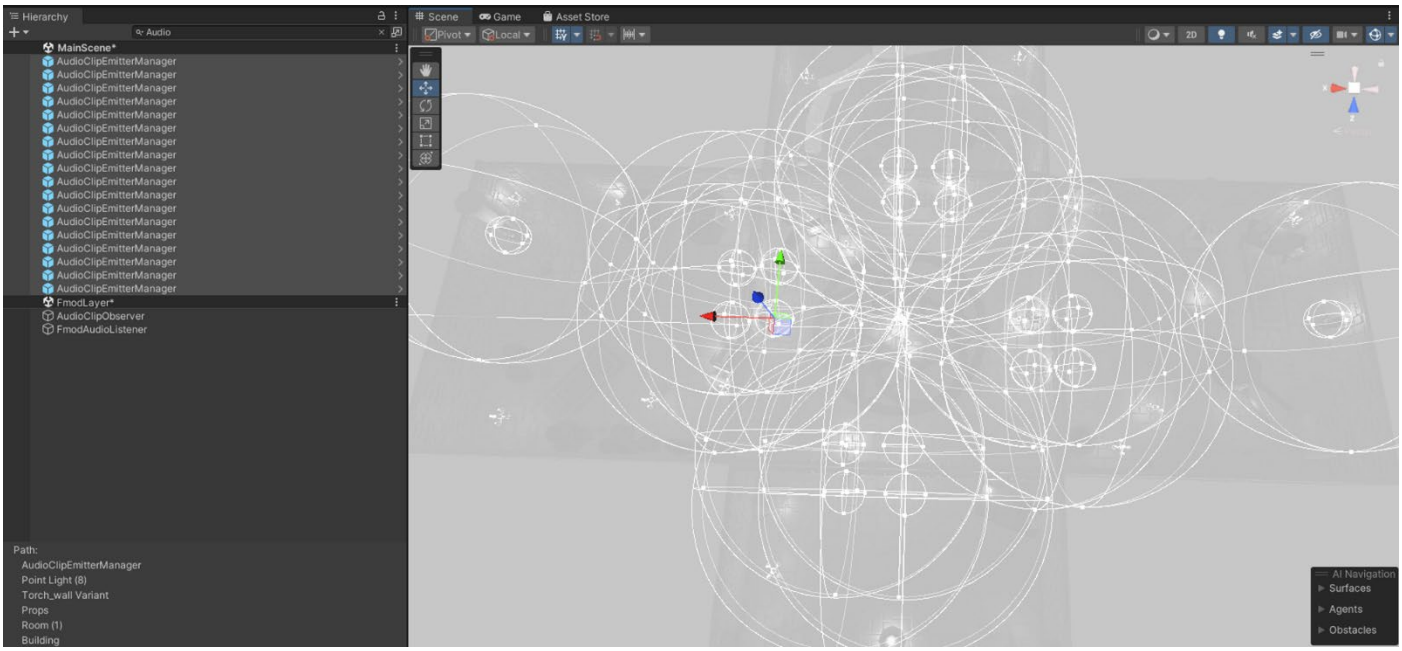
- **StopEventsOutdieMaxDistance**: Permet d'arrêter la lecture du son lorsque l'on sort de la sphère d'atténuation (*valeur maximale*).

2. La Sphère d’atténuation :

Pour voir la **sphère d’atténuation** de vos éléments dans la scène, vous devez avoir en sélection l’**AudioClipEmitterManager** de sélectionné :

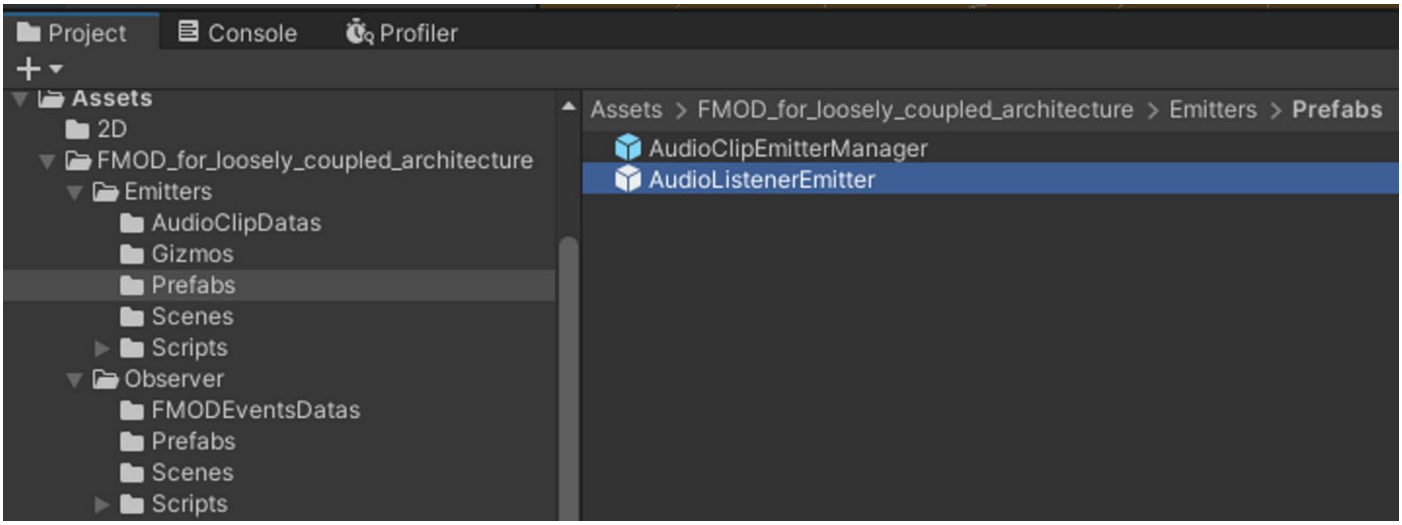


Une petite recherche de mes audios dans la scène, une sélection groupée, et je peux voir l’ensemble de mes **sphères d’atténuations** (*Fonctionne aussi en sélectionnant les **GameObject** possédant un **AudioClipEmitter** en **component***) :

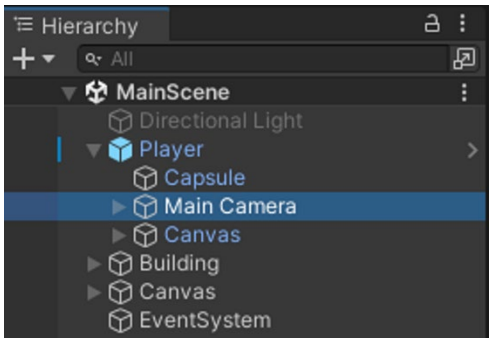


3. Ne pas oublier l'AudioListenerEmitter :

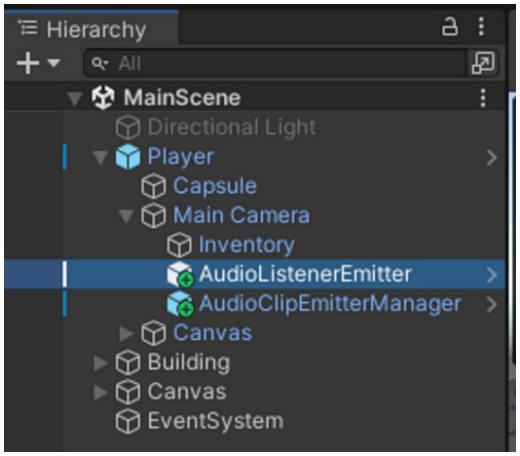
Pour pouvoir écouter vos sons vous aurez besoin d'**oreilles** dans votre scène. C'est là que le **prefab AudioEmitterListener** entre en jeu.



Dans la majorité des projets vous devez mettre le **prefab AudioListenerEmitter** sur le GameObject **MainCamera**. (*Dépendant du style de votre jeu, attention au top down shooter par exemple, qui devra gérer le panoramique différemment.*) Chargez la scène principale et recherchez l'emplacement de la caméra principale :



Glissez et déposez le prefab **AudioListenerEmitter** à l'endroit souhaité :

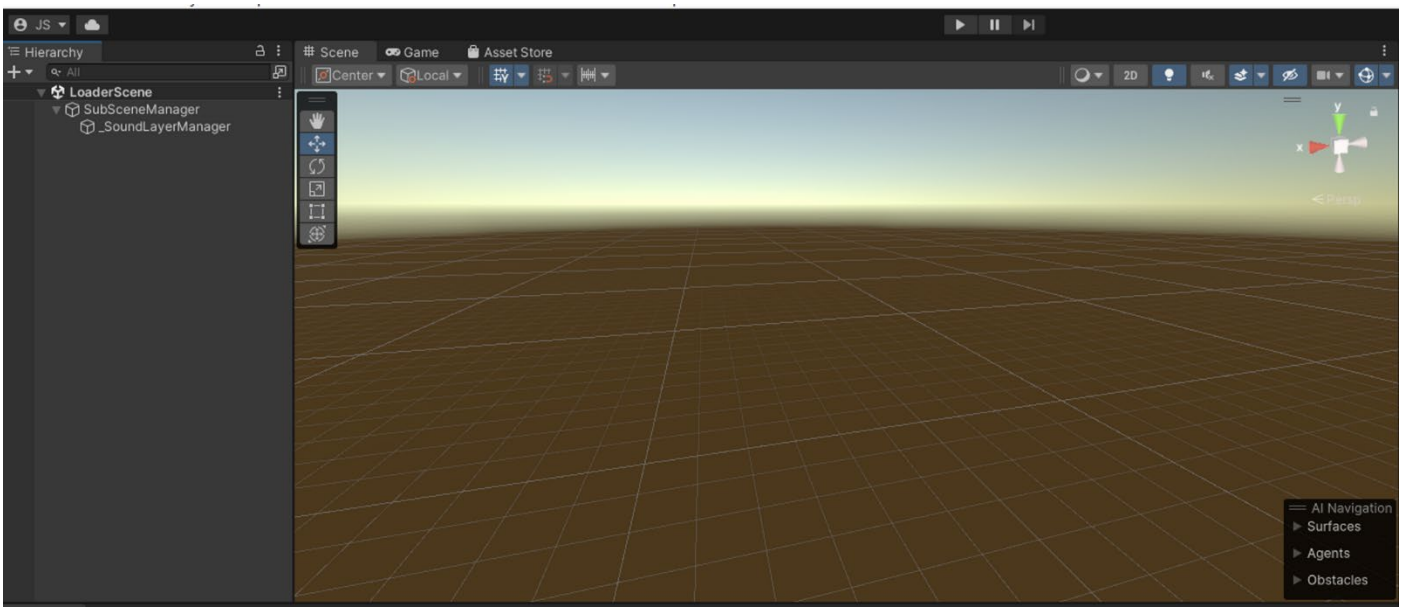


Vous n'avez rien à faire dans la partie **inspector**.

Sauvegardez la scène !

Tout est paramétré, place à l’écoute !

Chargez la scène **LoaderScene** puis lancez le **PlayRuntime**. Vous retrouverez l’ensemble des scènes et sous-scènes avec vos sons :

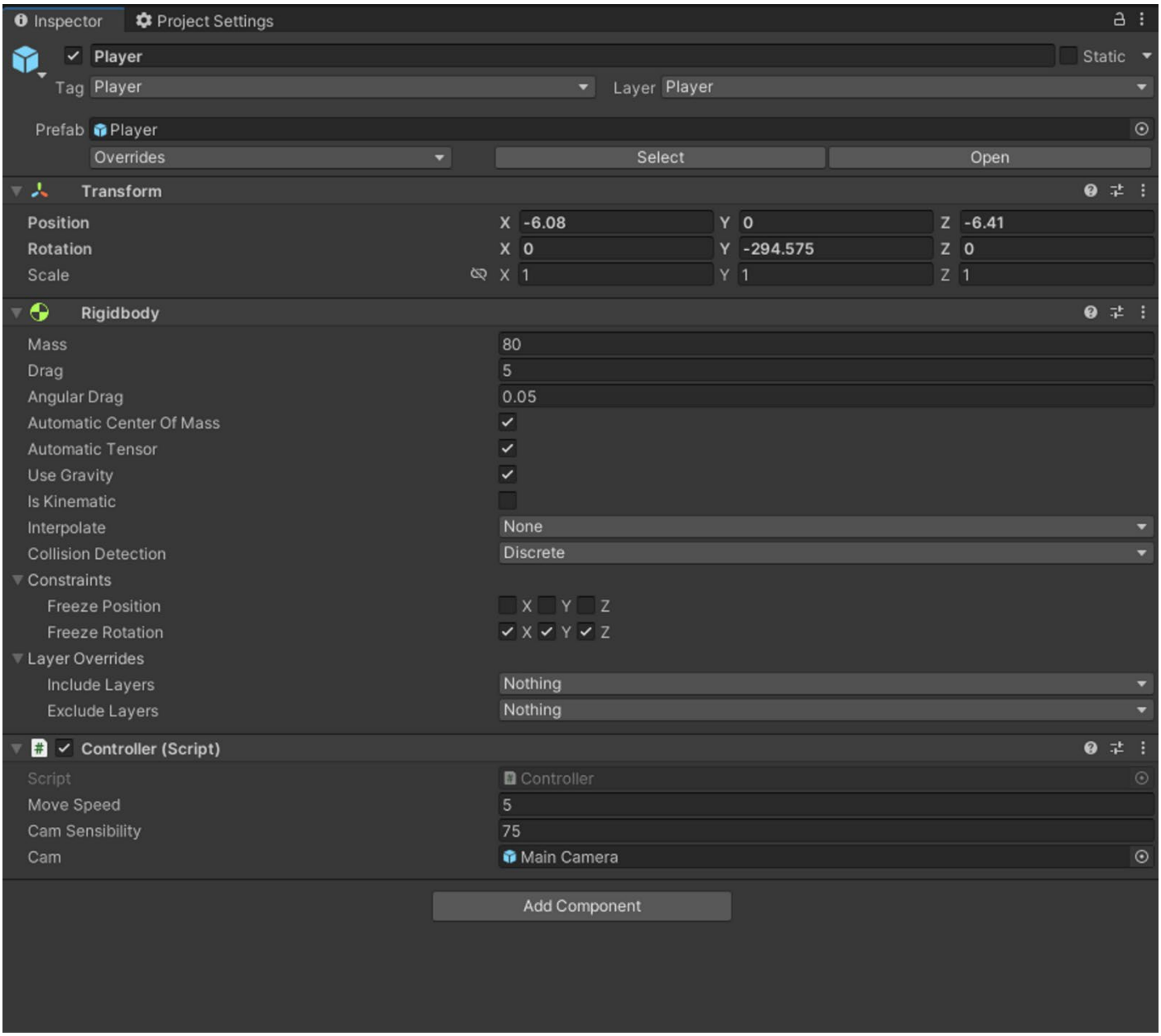


Bien entendu, dans le cadre où vous êtes **Sound Designer** et que vous devez poser vos sons dans la **scène principale**, pour éviter la manipulation de chargement de scène à chaque fois, il vous suffit de simplement glisser et déposer, en sous-scènes, **LoaderScene + FmodLayer** afin de pouvoir travailler rapidement et efficacement.

Quelques cas pratiques

1. Faire varier un paramètre de type “Built-in” de FMOD Studio :

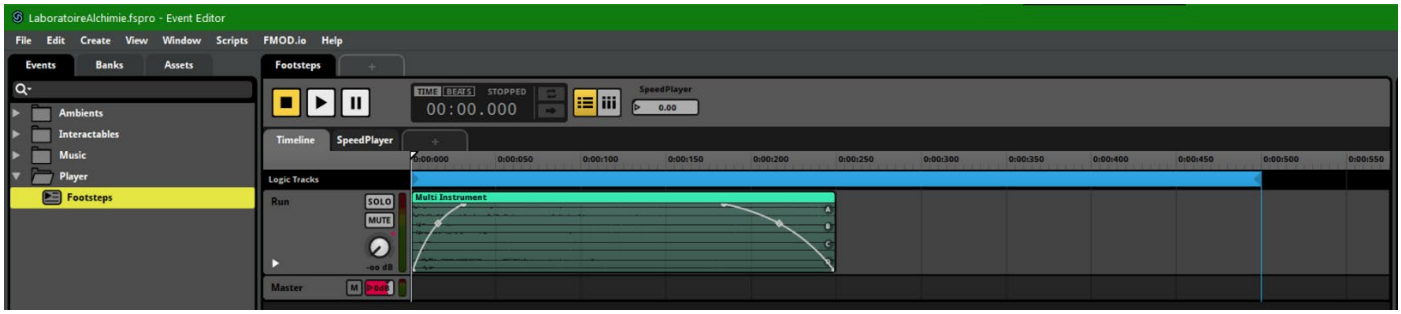
Dans notre exemple, nous possédons un GameComponent **Player** qui possède un **RigidBody**.



La vitesse de déplacement maximal du personnage dans ce projet est égale à une valeur de 5.

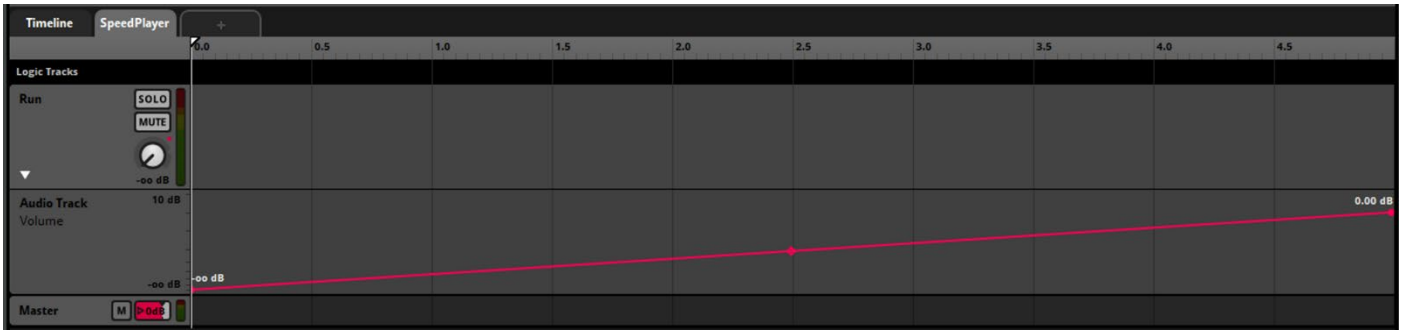
Je vais donc utiliser cette information afin de créer mon paramètre dans **FMOD Studio** :

Un **Event** nommé **Footsteps** a été créée :

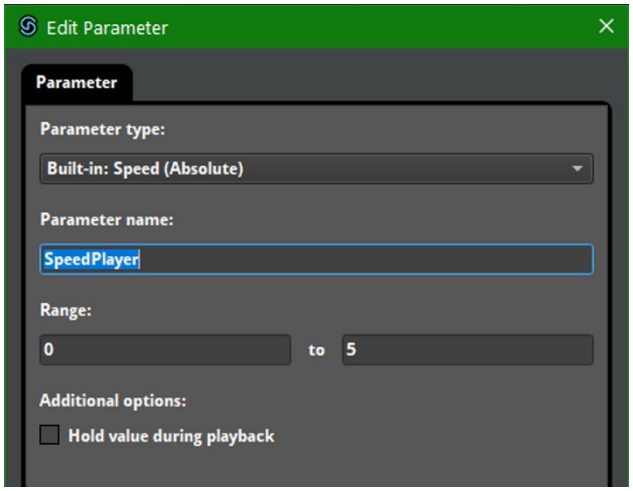


Il y a une boucle qui se jouera dès le chargement du jeu et on va faire varier l'audio en fonction de la vitesse du joueur.

Un paramètre nommé **SpeedPlayer** au format **Built-in** agira sur le volume en temps réel.



Voici comment est configuré ce paramètre **SpeedPlayer** :

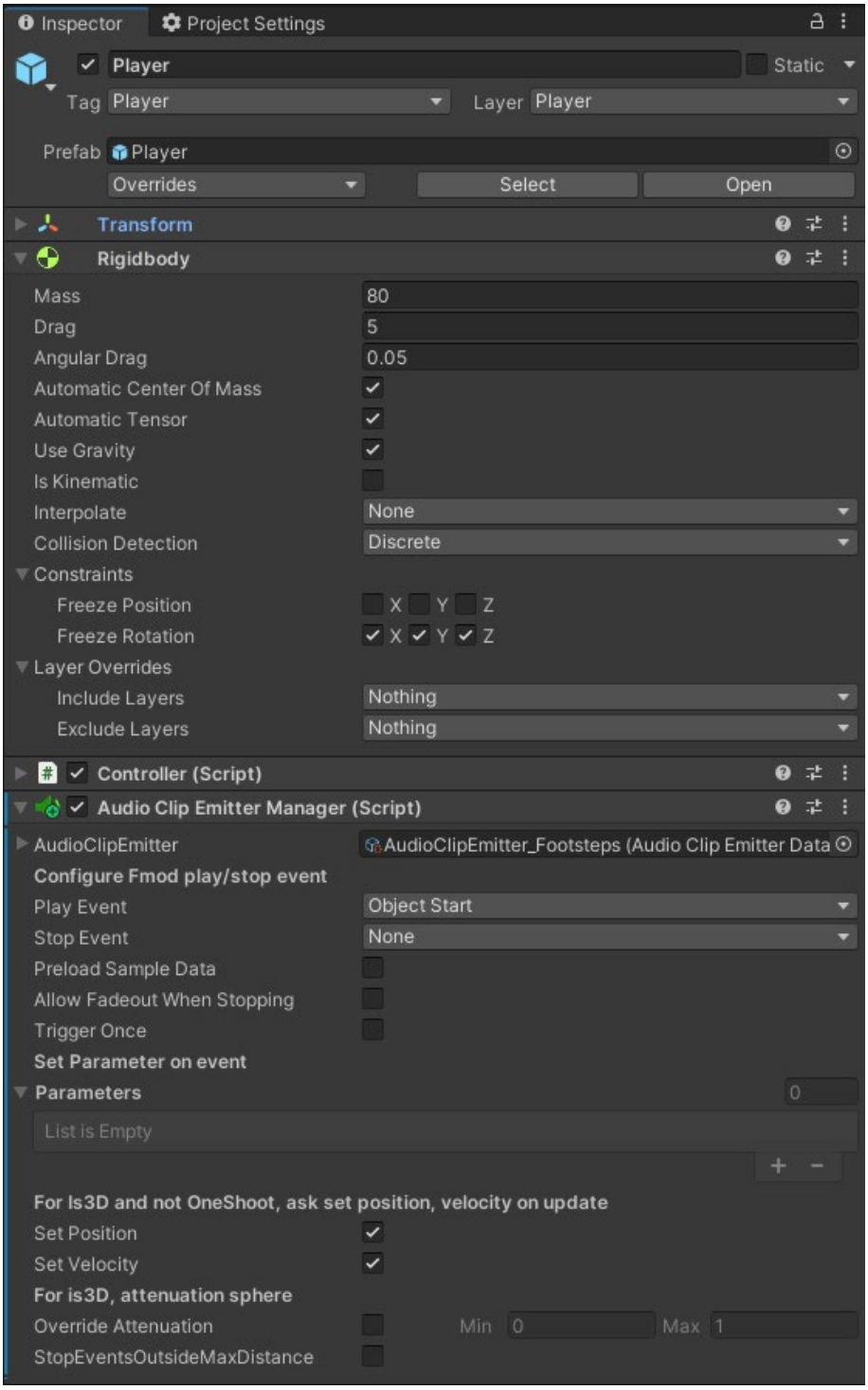


On lui renseigne une valeur minimale et maximale afin d'être cohérent avec la vitesse configurée en jeu.

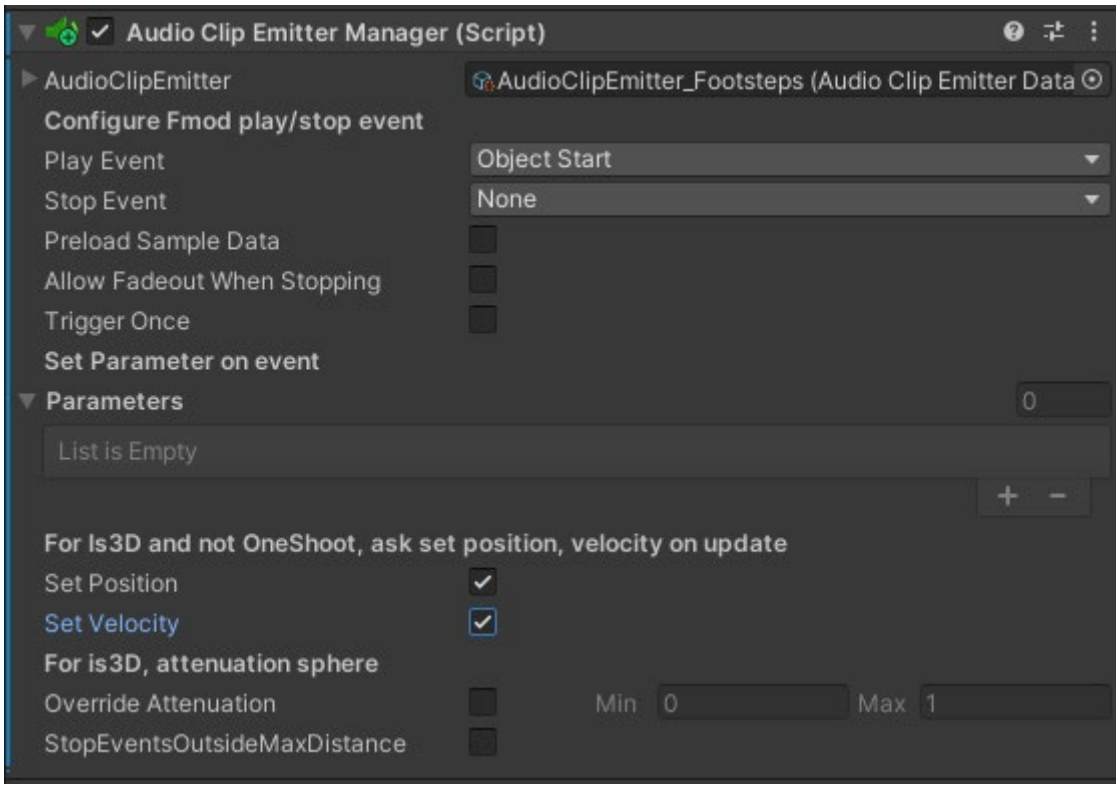
Maintenant que le paramètre et l'événement est créée, je pense à "Builder" mon projet FMOD Studio, afin de mettre à jour les banks qui sont appelés dans le projet et je retourne sur Unity pour relier cet event à mon joueur.

Infos supplémentaires : Concernant le sujet lié aux paramètre Built-in je vous renvoie à la documentation de FMOD à cette adresse ⇒ https://www.fmod.com/docs/2.00/studio/parameters-reference.html#built-in-parameters
Vous aurez un descriptif global du fonctionnement des paramètres Built-In de FMOD Studio.

De retour sur **Unity**, nous allons poser un component sur le **Player**, au même niveau que le **component RigidBody**, nécessaire au bon fonctionnement de notre event qui a besoin de récupérer les informations de celui-ci afin de faire varier notre paramètre **SpeedPlayer**.



Voici comment paramètre notre **AudioClipEmitterManager** component afin que l'ensemble fonctionne :

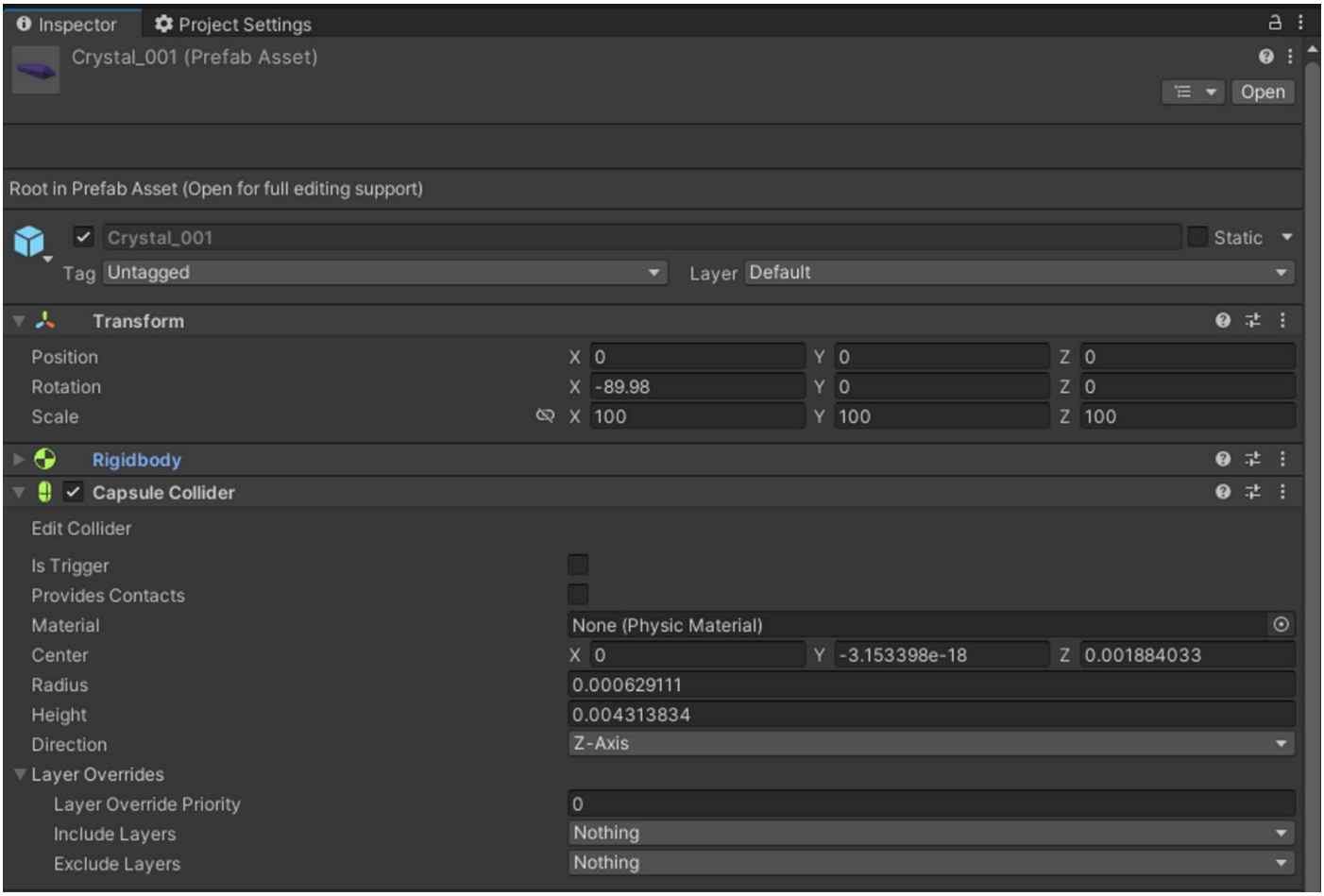


Ayant créée un événement avec une boucle infinie, j'ai choisi de faire démarrer cet événement au démarrage du jeu. Pour que le paramètre reçoive la mise à jour en temps réel de la **vitesse de déplacement** j'ai besoin que la case **Set Velocity** soit cochée.

Info complémentaire : Vous pouvez utiliser ce même procéder sur tout élément en mouvement dans votre scène et possédant un **RigidBody** afin de créer un **effet de doppler** entre le joueur (*qui écoute le son*) et l'objet qui jouera un son et qui sera en mouvement en se rapprochant ou s'éloignant du joueur.

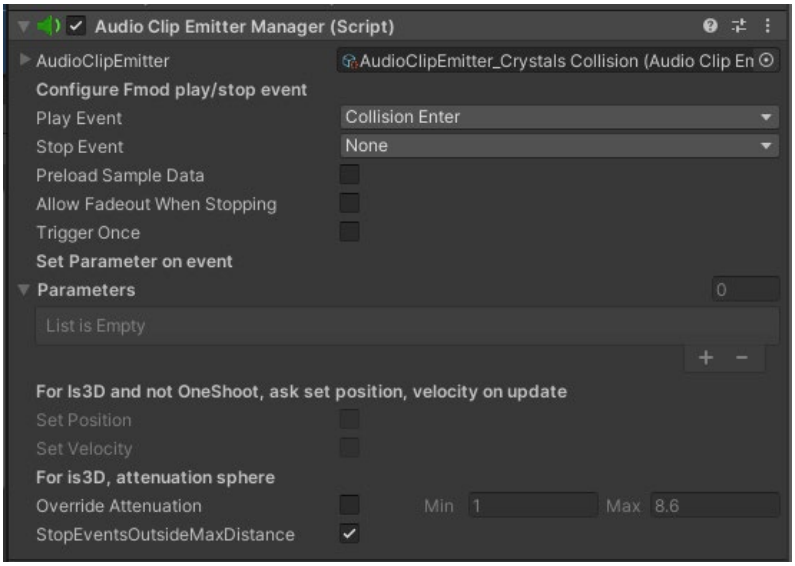
2. Jouer des sons en fonction de la collision :

Dans notre exemple de projet nous allons nous rendre sur les cristaux. Chaque cristal du jeu possède une capsule collider component qui est important pour faire fonctionner ce qui va suivre.



Je souhaite faire en sorte qu'un **son d'impact** un peu cristallin se fasse entendre quand mon **Player** entre en **collision** avec ces différents cristaux (*500 cristaux différents au total*). Pour ce faire, j'ai un évent simple dans **FMOD Studio**, avec mon asset sonore.

Sur le cristal je vais donc renseigner cet **event** dans mon **AudioClipEmitterManager** Component. On doit avoir ceci :



Je prends la décision de renseigner que “**Play Event**” pour définir quand le son se fera entendre. Je ne souhaite pas que le son s’arrête de jouer quand je ne suis plus en collision avec le cristal, donc je ne renseigne rien pour “**Stop Event**”. Le nombre d’élément sonore simultané sera définie directement dans l’événement créée dans **FMOD Studio** avec le “**Max Instance**”, que je réglerai également sur un Bus dédié aux SFX.

Voici à quoi ressemble ces paramètres dans **FMOD Studio** :