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

Prése	entation de l'extension	2
1.	Schéma Synthétique du fonctionnement :	2
2.	Quelques explications :	2
Confi	gurer son projet pour intégrer l'extension :	3
1.	Intégrer les 2 scènes à votre projet	4
2.	Configuration de FMOD Plugin :	4
3.	Configuration des scènes :	5
Créer	r la base de données sonores du projet :	
1.	Créer la base de données :	8
2.	Paramétrer les scènes :	10
3.	Informations complémentaires !	12
Utilisa	ation concrète de l'extension :	12
1.	Comment utiliser l'AudioClipEmitterManager	12
2.	La Sphère d'atténuation :	19
3.	Ne pas oublier l'AudioListenerEmitter :	20
Tout	est paramétré, place à l'écoute !	21
Quelo	ques cas pratiques	22
1.	Faire varier un paramètre de type "Built-in" de FMOD Studio :	22
2.	Jouer des sons en fonction de la collision :	

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 :

FMOD_for_loosely_coupled_architecture

Emitters

🕨 🖿 Observer

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 :

Cooperative Duild	
Scenes in Build	
FMOD_for_loosely_coupled_architecture/Emitters/Scenes/LoaderScene	0
FMOD_for_loosely_coupled_architecture/Observer/Scenes/FmodLayer	1
Scenes/MainScene	2

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)

Inspector Project Settings		
S FMOD Settings		
Bank Import		
Source Type	Single Platform Build	
Build Path	/RecherchePierePhilosophale/FMOD Bank/Desktop	Brows
Import Type	Streaming Assets	
FMOD Bank Sub Folder		
Refresh Banks	After 5 seconds 🔻 🗹 Show Status Window	
Event Linkage	Path	

3. Configuration des scènes :

On charge la scène LoaderScene:



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

🔻 🛠 LoaderScene
🔻 😭 SubSceneManager
🕤 _SoundLayerManager

🖲 Inspector 🔹 Project Settings 🗧 🗄										
SubSceneManager Statio							atic	•		
Tag Untagged			▼ Layer 🛙)efa	ault					•
🔻 🙏 🛛 Transform								0		:
Position		х	0	Y	0	z	0			
Rotation		х	0	Y	0	Z	0			
Scale	8	х	1	Y	1	z	1			
🔻 井 🗹 Scene Loader Manager (Script)								Ø	÷	1
Script SceneLoaderManager 0										
Scene Name Of Fmod Layer	ne Name Of Fmod Layer FmodLayer									
Scene Name Of Gameplay MainScene										

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*).
 <u>Dossier par défaut :</u>
 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*) <u>Dossier par défaut :</u> Assets/FMOD_for_loosely_coupled_architecture/Emitters/AudioClipData

Inspector Project Settings									а	:
🔵 🖌 DatabaseTools								Sta	atic	•
Tag Untagged				Layer D	efa	ault				•
🔻 🙏 Transform								0		:
Position			0			0	0			
Rotation			0			0	0			
Scale	8		1							
🔻 ≢ 🖌 FLCA Database Builder (Scr	ipt)							0		:
Script			FLCAData							۲
Generate Database										
Path For Fmod Event Data		A	ssets/FMO	D_for_loos	sel	y_coupled_architect	e/Observ	/er/F	мо	DE
Path For Audio Cip Data		A	ssets/FMO	D_for_loos	sel	y_coupled_architect	e/Observ	/er/A	udi	oC
Create All Event Database										
Create Buses Db										
Create VCADB										
		A	Add Compo	onent						

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 :

🔻 # 🗹 FLCA Database Builder (Script)		07≓ ÷
Script	FLCADatabaseBuilder	
Generate Database		
Path For Fmod Event Data	Assets/FMOD_for_loosely_coupled_architecture/Obser	ver/FMODEventsDatas
Path For Audio Cip Data	Assets/FMOD_for_loosely_coupled_architecture/Emitte	ers/AudioClipDatas
Create All Event Database		
Create Buses Db		
Create VCADB		
	Add Component	

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 :

- 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*)
- 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**.

 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.





Group of Audio Source data List Of Audio Clip Data	Missing (Group Of Audio Clip Emitter Data)	0
Test clip if FmodLayer Scene is loaded Index Of Audio Source Data To Play	Select GroupOfAudioClipEmitterData 🛛 🗶	
Play Stop Isplaying	Assets Scene	



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



🔻 # 🔽 Audio Clip Observer (Scrip	t)	0	<u>1</u> 4	:
Script	AudioClipObserver			
Fmod Audio Listener	FmodAudioListener			\odot
Links to databases				
List Of Fmod Events	କ୍ତListFmodEvent (Group Of Fmod Event Da	ta)		\odot
List Of Buses	ଦ୍ଧListBuses (Buses Data)			\odot
List Of VCA	କ୍ତListVCAs (VCA Data)			\odot
Set all volumes on update Set Volumes	~			
Get all volumes from banks on sta Get Volumes On Start	nrt			
Testing				
Index Of Event To Play	0			
Master Volume Info	0			
	Play / Stop			

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** :

🗹 📢 🔽 Audio Clip Emit	🕐 🔨 Audio Clip Emitter Manager (Script) 🛛 🛛 🥹 🕫 🖓							
	None (Audio Clip Emitter Data)			\odot				
Configure Fmod play/	stop event							
Play Event	None			•				
Stop Event	None			•				
Preload Sample Data								
Allow Fadeout When S	topping							
Trigger Once								
Set Parameter on eve	nt							
' Parameters								
List is Empty								
		+						
For Is3D and not OneS	Shoot, ask set position, velocity on update							
Set Position								
Set Velocity								
For is3D, attenuation	sphere							
	Add Component							

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

Nous renseignons alors l'event Torches :

		🔻 📢 🔽 Audio Clip	Emitter Ma	nager ((Script) Ø ∓	
		AudioClipEmitter			AudioClipEmitter_Torches (Audio Clip Emitter D	εO
		Configure Fmod	play/stop ev	/ent		
	Select	AudioClipEmitterDat	a	x	Object Enable	•
	0				Object Disable	•
	4	0	•	10/17		
	Assets	Scene		9917		
		one udioClinEmitter Bello	w		0	
o io	Ch A	udioClipEmitter_Book			position, velocity on update	
ion.	GO A	udioClipEmitter_Caulo	iron Hot			
	60 A	udioClipEmitter_Crista	al Hot			
	M A	udioClipEmitter_Cryst	als Collision			
		udioClipEmitter_Cryst	als Grab tens		Min 1 Max 7	
	Go A	udioClipEmitter_Gate	Open		~	
	Go A	udioClipEmitter_Lens				
	- 10 A	udioClipEmitter_Lens	Beam		Add Component	
\star		udioClipEmitter_Main	Music			
	GA A	udioClipEmitter_Philos	t completed	ne coo		
	Go A	udioClipEmitter_Ramp)S			
	6 A	udioClipEmitter_Read	ing			
		udioClipEmitter_Reve	rb			
		udioClipEmitter_Torch	r Drop			
	Q0 A		ыор			

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

🔻 📣 🗹 Audio Clip Emitt	er Manager (Script) 🛛 🥹
AudioClipEmitter	AudioClipEmitter_Torches (Au
Configure Fmod play/s	top event
Play Event	Object Enable
Stop Event	None
Preload Sample Data	Object Start
Allow Fadeout When Sto	Object Destroy
Trigger Once	Trigger Enter
Set Parameter on even	Trigger Exit
Parameters	Trigger Enter 2D
	Trigger Exit 2D
	Collision Enter
For Is3D and not OneSI	Collision Exit
Set Position	Collision Enter 2D
Set velocity	Collision Exit 2D
Override Attenuation s	Object Enable
StonEventsOutsideMax	Object Disable
otope ventaoutaidemaxi	Mouse Enter
	Mouse Exit
	Mouse Down
	Mouse Up

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*).

📢 🔽 Audio Clip Emitter Manager (Sc	ript)				0	칶	
AudioClipEmitter Configure Fmod play/stop event	✿AudioClip	pEmitt	er_Torches (Au	dio Clip Emitte	r Da	ta)	•
Play Event	Object Ena	ble					•
Stop Event	Object Dis	able					•
Preload Sample Data							
Allow Fadeout When Stopping							
Trigger Once							
Set Parameter on event							
Parameters					0		
List is Empty							
					+		
For Is3D and not OneShoot, ask set po	sition, veloo	city on	update				
Set Position							
Set Velocity							
For is3D, attenuation sphere							
Override Attenuation		Min		Max 7			
StopEventsOutsideMaxDistance							

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	
Allow Fadeout When Stc	
Trigger Once	1
Set Parameter on event	
Parameters	0
List is Empty	
	+ -
For Is3D and not OneShoot, ask set position, velocity	on updat
Set Position	
Set Velocity	
For is3D, attenuation sphere	
Override Attenuation Min 1 Max 7	7
StopEventsOutsideMax	

- 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).
 <u>Cas de figure</u> : 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.

Inspector Project Settings							а:
Y Player						Stat	tic 🔻
Tag Player		 Layer Player 					-
							~
Pretab Player		Coloct			0.000		o
Overndes	1	Select		<u>.</u>	Open		
Transform						6	* :
Position		X -6.08	Y 0	Z	-6.41		_
Rotation	65	X 0	Y -294.575	Z	0		_
Scale		XI	Y I		1		
🔻 🔶 Rigidbody						9	21
Mass		80					
Drag		5					
Angular Drag		0.05					
Automatic Center Of Mass		~					
Automatic Tensor		~					
Use Gravity		~					
Is Kinematic							
Interpolate		None					•
Collision Detection		Discrete					•
▼ Constraints							
Freeze Position		🗌 X 🗌 Y 🗌 Z					
Freeze Rotation		🗹 X 🗹 Y 🗹 Z					
▼ Layer Overrides							
Include Layers		Nothing					•
Exclude Layers		Nothing					•
🔻 # 🗹 Controller (Script)						0	# i
Script		Controller					\odot
Move Speed		5					
Cam Sensibility		75					
Cam		🗊 Main Camera					\odot
		Add Component					
L. L		Add Component					

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 :

9 LaboratoireAlchimie fsoro - Event Editor			
File Edit Create View Window Scripts	FMOD.io Help		
Events Banks Assets	Footsteps +		
Q- ▶ Ambients		TIME Effection SpeedFlayer 00:00.000 Image: SpeedFlayer 0.00	
▶ Interactables	Timeline SpeedPlayer	+	
Music		100-000 0:00-050 0:00-100 0:00-150 0:00-250 0:00-250 0:00-350 0:00-450 1	0:00:500 0:00:550
V Player	Logic Tracks		
Footsteps	Run SOLO	Multi-instrument	
	MUTE O -oo dB		
	Master M Bode		

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.

Timeline	SpeedPlayer										
		6.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5
Logic Tracks											
Run	SOLO MUTE										
•	-00 dB										
Audio Track Volume	10 dB										0.00 dB
	-oo dB	-oo dB									
Master	M DOG										

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 ⇒ <u>https://www.fmod.com/docs/2.00/studio/parameters-reference.html#built-in-parameters</u>

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**.

Inspector Project Settings		а	:
😭 🗸 Player		Static	•
Tag Player	 Layer Player 		
Prefah 🛱 Plaver			•
Overrides	Select Open		Ĭ
Transform		o. ∹•	
		ο.÷	
Mass	00	•	
Drag	5		
Angular Drag	0.05		
Automatic Center Of Mass	v		
Automatic Tensor			
Use Gravity	7		
Is Kinematic			
Interpolate	None		
Collision Detection	Discrete		•
▼ Constraints			
Freeze Position	XYZ		
Freeze Rotation	✓ X ✓ Y ✓ Z		
Include Layers	Nothing		
Exclude Layers	Nothing		
🕨 # 🖌 Controller (Script)		0 ‡	
🔻 👍 🔽 Audio Clip Emitter Manager	(Script)	9 ≓	
▶ AudioClipEmitter	& AudioClipEmitter_Footsteps (Audio Clip Emitte	r Data	
Configure Fmod play/stop event			
Play Event	Object Start		
Stop Event	None		
Preload Sample Data			
Allow Fadeout When Stopping			
Trigger Once			
Set Parameter on event			
▼ Parameters			
For Is3D and not OneShoot, ask set	position, velocity on update		
Set Position	~		
Set Velocity	×		
For is3D, attenuation sphere			
Override Attenuation			

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

🔻 👍 🔽 Audio Clip Emitter Manager (Script)	Ø ‡	:
▶ AudioClipEmitter	AudioClipEmitter_Footsteps (Audio Clip Emitter	er Data	\odot
Configure Fmod play/stop event			
Play Event	Object Start		•
Stop Event	None		•
Preload Sample Data			
Allow Fadeout When Stopping			
Trigger Once			
Set Parameter on event			
▼ Parameters		0	
List is Empty			
		+ -	
For Is3D and not OneShoot, ask set	position, velocity on update		
Set Position			
Set Velocity			
For is3D, attenuation sphere			
Override Attenuation	Min 0 Max 1		
StopEventsOutsideMaxDistance			

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.

Inspector Project Settings						а	:
Crystal_001 (Prefab Asset)						0 i	1
						🛱 🔻 Open	
Root in Prefab Asset (Open for full editing support)							
Crystal 001						Static 🔻	
		 Laver Defaul 	t				
		Layer Dolladi					
🔻 🙏 🛛 Transform						0 ≓ :	
Position	X 0			0	Ζ	0	
Rotation	X -89.98			0	Ζ	0	
Scale හැ	X 100			100	Z	100	
► 🔶 Rigidbody						07‡ :	
🔻 🗍 🖌 Capsule Collider						0 ≓ :	
Edit Collider							
ls Trigger							
Provides Contacts							
Material	None (Pr	ysic Material)					
Center	X 0			-3.153398e-18	Ζ	0.001884033	
Radius	0.00062	9111					
Height	0.00431	3834					
Direction	Z-Axis						
Tayer Overrides							
Layer Override Priority	0						
Include Layers	Nothing					.	
Exclude Layers	Nothing						

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 :

V	📹) 🔽 Audio Clip Emitter Manager (Script)	0 ‡	:
	AudioClipEmitter Configure Fmod play/stop event	AudioClipEmitter_Crystals Collision (Audio	Clip En	0
	Play Event	Collision Enter		
	Stop Event	None		
	Preload Sample Data Allow Fadeout When Stopping Trigger Once Set Parameter on event			
	Parameters			
	For Is3D and not OneShoot, ask set Set Position Set Velocity For is3D, attenuation sphere	position, velocity on update		
	Override Attenuation StopEventsOutsideMaxDistance	Min 1 Max 8.6		

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 :

