

Département des industries créatives et numériques

Automatisation du placement de sources audio
adaptatives environnementales dans Unreal Engine 5
Création d'un plugin à destination des Sound Designers

Mémoire présenté par

MARTINS Miguel

*En vue de l'obtention du diplôme de
Master en Jeu vidéo*

sous la supervision de

CHAPELLE Joakim, superviseur

ADANS Paschal et DERVAUX Thibaut, coachs

Année académique 2024-2025

Table des matières

| | |
|---|----|
| 1. Introduction | 3 |
| 2. Problématique | 4 |
| 3. État de l'art..... | 5 |
| Définitions..... | 5 |
| Techniques de sonorisations | 7 |
| Ambiances phoniques | 7 |
| Sources audio localisées | 8 |
| Automatisation de sources audio | 9 |
| Audio adaptatif | 10 |
| Outils éditeur..... | 12 |
| Récapitulatif..... | 13 |
| 4. Devis méthodologique..... | 14 |
| Méthodes | 14 |
| Outil développé | 14 |
| 5. Présentation du prototype | 16 |
| Structure | 16 |
| Accessibilité | 17 |
| Challenges | 18 |
| 6. Analyse réflexive du prototype | 19 |
| Le prototype Explorer | 19 |
| Le jeu Friends Don't Ghost | 22 |
| Le prototype Parkware | 24 |
| 7. Conclusion | 26 |
| 8. Recommandations..... | 28 |
| 9. Bibliographie..... | 29 |
| 10. Résumé..... | 31 |
| 11. Mots-clés | 31 |

1. Introduction

Ce mémoire explore les pistes de créations d'outils d'automatisation de sonorisation dans le cadre du jeu vidéo. Cette étude décomposera les techniques les plus courantes de placement audio pour les automatiser par la production d'outils dans le moteur Unreal Engine 5. Ces outils seront ensuite mis à l'épreuve dans différents types de jeux vidéo.

L'objectif premier de développement est d'obtenir un impact sur le temps d'implémentation audio tout en permettant une flexibilité d'itérations.

C'est souvent en fin de production que le son peut être intégré car il est nécessaire d'implémenter les appels à fonctions permettant sa lecture. Pour travailler en parallèle entre sound designer et programmeur, il existe des middlewares permettant d'émuler le comportement du jeu et ainsi être autonome vis-à-vis des autres pôles dans leurs pipelines de production. L'un d'eux est Wwise (Audiokinetic, s. d.) qui est cité à plusieurs reprises dans ce mémoire.

Le prototype a pour but de diminuer l'effort à fournir pour lier les deux pôles et de permettre des réglages de manière granulaire lors des tâches les plus génériques. De cette façon, le son est testable en jeu dès l'outil intégré dans le moteur.

Concernant le placement des sources audio environnementales spatialisées, il est envisageable d'automatiser leur génération et leurs variations grâce à des règles adaptatives au décor.

En intégrant divers flux audios, il est possible d'affiner le contrôle du mixage de manière dynamique et automatique. A l'aide de l'outil, les réglages par défaut du moteur de jeu peuvent-être homogénéisés et automatisés en fonction de la granularité des paramètres.

Néanmoins, il existe une multitude de styles et genres de jeux vidéo. On peut par exemple trouver des jeux en première personne ou en troisième, en 2 dimensions ou en 3, ainsi que des jeux au cycle jour nuit ou des jeux intemporels ou figés dans le temps.

Ce mémoire analyse et exploite les projets suivants :

- Electric Dreams, un projet mettant en scène une voiture électrique dans une jungle.
- Rage Audio, un logiciel développé à partir de 2005 par le studio Rockstar et analysé pour son utilisation dans leur jeu : Grand Theft Auto-V
- Soundscape, un plugin d'instanciation audio développé par Epic Games
- Solar Ash Kingdom, un jeu d'action, aventure et de plateforme développé par Heart Machine.
- Outcast 2 et « Static Mesh », à travers les conférences du studio Demute au Game Camp
- Metasounds, un plugin DSP développé par Epic Games
- Unreal Engine 5, un moteur de jeu exploité dans ce mémoire et développé par Epic Games

Du point de vue épistémologique, la première étape consiste à déterminer les différents types d'implémentations et leurs paramètres, en analysant des productions de jeu AA et AAA ayant exposés leurs méthodes de travail ainsi que des projets réalisés lors de compétitions Game Jam. Dans la seconde étape, un outil est développé pour appliquer diverses automatisations.

Finalement, cet outil est testé dans des projets non entièrement finis dans le but de les améliorer.

2. Problématique

Comment faciliter le travail des sound designers, en automatisant le placement en temps réel de sources audio à proximité du joueur dans un environnement à l'aide d'un outil ?

Le but de l'outil serait de faciliter le travail des sound designers par le biais de placements de sources audio.

Il permettrait de construire une sonorisation cohérente avec les itérations de niveaux dès la pré-production avec une réduction probable des coûts d'intégration, associée à une plus grande rapidité de prototypage.

Le résultat visé pour ce mémoire est un prototype sous forme de plugin exploitable par des projets utilisant Unreal Engine 5.

Dans le cadre de création intensive de jeux ou des Game Jams, avec des limites de temps développement, il est important de disposer d'outils réutilisables entre les projets.

Ainsi, le prototype prendra la forme d'un plugin conçu pour les tâches d'implémentation sonore, tout en offrant une extensibilité adaptable aux besoins spécifiques de chaque jeu.

Par son aspect générique, son utilisation sera testée et affutée pour être compatible avec différents styles de jeu, notamment dans un jeu de parkour à la première personne, auquel j'ai participé. Il sera également utilisé dans un jeu à la troisième personne. La possibilité d'exploiter ces projets est une raison supplémentaire pour l'utilisation du moteur Unreal Engine 5.

Les différents points de vue des projets permettront de créer un outil se concentrant sur les tâches les plus communes et répétitives tout en testant leur pertinence. Pour répondre à leurs besoins, le prototype doit être suffisamment modulaire et flexible pour automatiser l'implémentation des sources audio et de leurs paramètres via les outils de l'éditeur. Grâce à son fonctionnement en temps réel, le plugin influencera le mixage audio du jeu ainsi que l'instantiation des sources sonores.

Grâce à l'automatisation de la sonorisation environnementale, il sera possible d'intégrer des sons aux éléments affichés à l'écran sans avoir à disposer manuellement une multitude d'instances audio. Ce processus reposera sur des règles procédurales garantissant un positionnement précis et efficace.

Également, le placement des sources s'accompagnera d'effets pouvant être combinés pour exploiter des changements environnementaux ou temporels. Un exemple est la sonorisation d'un buisson balayé par le vent en journée, dont le son changerait en soirée pour être amplifié et combiné avec le chant des cigales.

3. État de l'art

Définitions

Pour aborder les méthodes de sonorisation environnementale, voici leurs concepts clés :

DSP : Pour Digital Signal Processing, c'est un système de traitement de signaux numériques.

Dans le cadre de Metasounds le système se concrétise sous forme de graphiques permettant aux sound designers de créer des systèmes audios procéduraux. (Epic Games, s. d.)

Audio monophonique : Il s'agit du traitement audio de manière monaurale comparable à un son en une dimension (Ciesla, 2022, p. 231, 232)

Audio stéréophonique : Il s'agit d'un traitement audio en 2 dimensions. Il comporte deux canaux : un droit et un gauche. (Ciesla, 2022, p. 231, 232)

Audio 2D : Il s'agit de sons stéréo non-spatialisés.

Ils sont utiles en tant que musique, feedback d'interface et lorsque l'auditeur reste statique. (Epic Games, s. d., p. Audio Engine Overview).

Il existe également des sons 3D. Ces sons sont représentés sur un système 3D de coordonnées cartésiennes permettant ensuite de les spatialiser. (Ciesla, 2022, p. 231, 232)

Les sons 2D et 3D transmettent des informations au joueur.

L'ouvrage « Sound and Music for Games » mentionne que la musique peut être **diégétique** si la source audio provient d'une radio dans le monde. La musique diégétique peut également être appelée musique source.

A contrario, si seul le joueur entend la source et que la musique n'affecte pas les personnages dans le monde du jeu, la musique est **non-diégétique**, souvent appelée musique de scène. (Ciesla, 2022, p. 260)

Spatialisation : Il s'agit de la simulation de phénomènes sonores basée sur la localisation tels que l'orientation, l'atténuation, la propagation, l'obstruction et la réverbération (Epic Games, s. d., p. Audio Glossary). Additionnellement, on peut aussi compter l'effet Doppler qui modifie la tonalité de l'audio dans les effets de spatialisation. (Verfaille et al., 2006, p. 2). Comme l'ajoute Robert Ciesla dans son ouvrage, l'effet Doppler apporte du réalisme et de l'immersion. Il est même indispensable pour les jeux de simulation de courses et de vol. (Ciesla, 2022, p. 241)

Le **panoramique** ou **réglage pan** consiste à donner aux auditeurs, l'information de direction d'émission du son. Les 2 méthodes les plus utilisées pour donner l'impression d'émission d'une direction précise sont :

- **L'IID (Interaural Intensity Difference)** applique une différence de volume avec une intensité supérieure du côté de la source sonore. Si un son provient de la gauche, il sera plus intense dans l'oreille gauche que dans la droite (Epic Games, s. d., p. Spatialization Overview).

Il est possible de calculer l'IID via la rotation entre l'auditeur et la source audio. (Epic Games, s. d., p. Spatialization Overview)

Cette approche peut également fonctionner sur les audio non-spatialisés. Dans le cas du projet

Electric Dream il s'agit de sonorisations quadriphoniques. (4 canaux audio).

Dans l'exemple montré par Dan Reynolds, les canaux audios subissent une rotation par rapport à la caméra pour suivre une orientation ancrée dans l'environnement. (Unreal Engine, 2023)

L'exemple oriente les canaux audios en temps réel sur base de l'axe vertical de la caméra (dans Unreal Engine l'axe Z).

- **L'ITD (Interaural Time Difference)** consiste à appliquer un délai lors de la reproduction du son par les baffles. Etant donné que la propagation du son n'est pas instantanée, il en résultera une latence entre la perception du son d'une oreille à l'autre (Epic Games, s. d., p. Spatialization Overview). » (Martins, 2025, p. 7, 8)

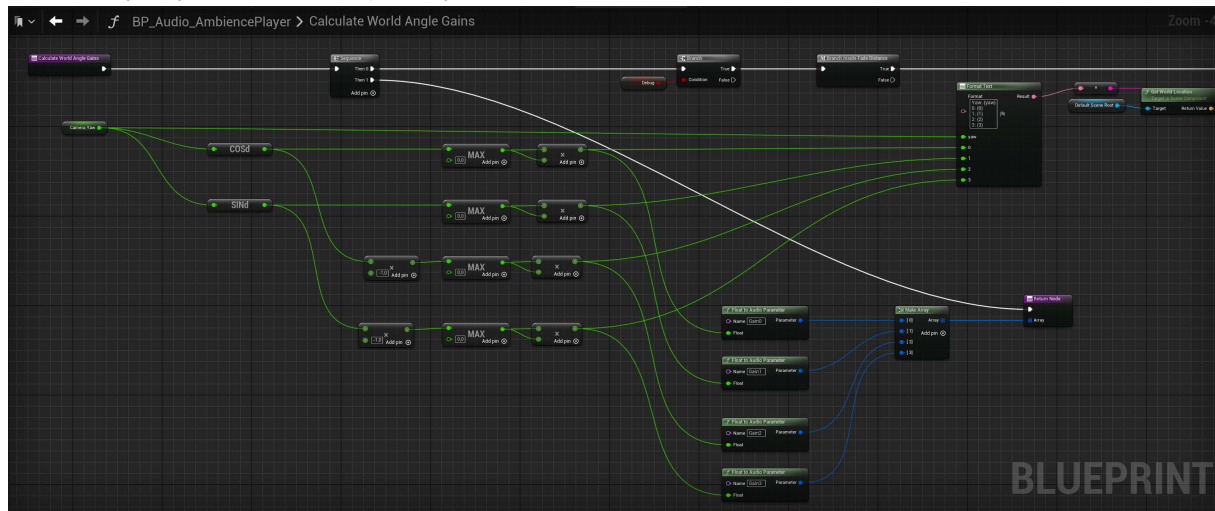


Figure 1: Blueprint BP_Audio_AmbiencePlayer du projet Electric Dream.

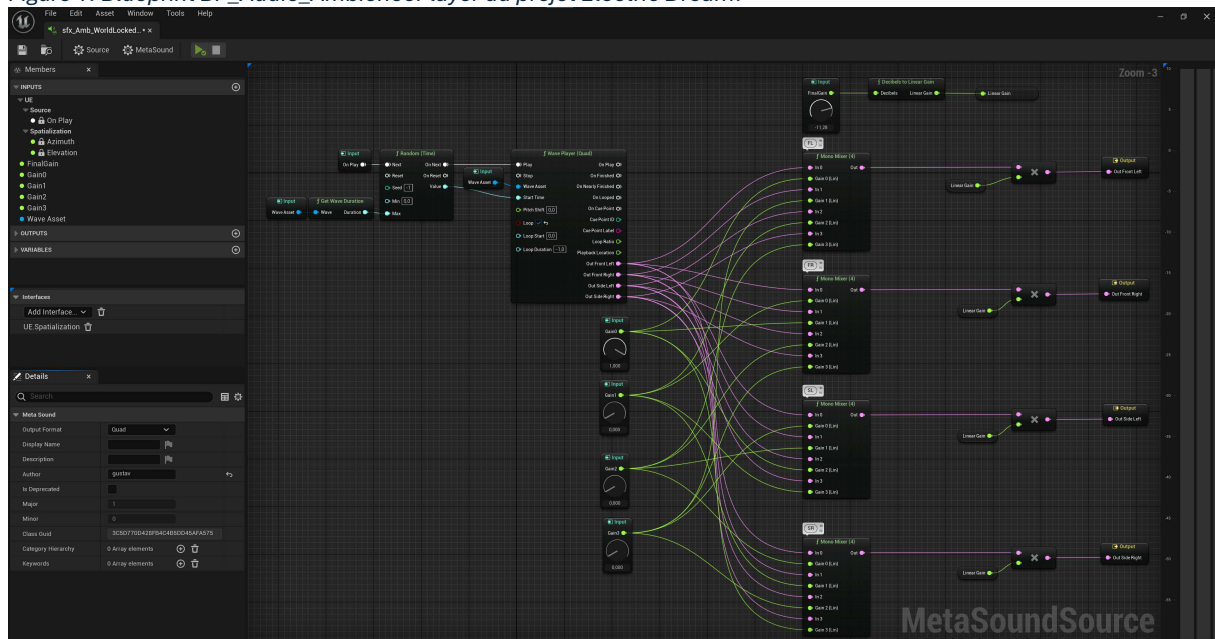


Figure 2 : MetaSound de panning pour audio quadriphonique provenant du projet Electric Dream.

Pour la prise en compte du plan vertical, on peut utiliser des **fonctions HRTFs (Head-Related-Transfer-Function)**. Cette fonction filtre les sons pour simuler sa propagation dans les épaules et la tête de l'auditeur. Les rotations utilisées dans les plans de l'écoute spatialisé sont appelées l'élévation et l'azimut. (Verfaillie et al., 2006, p. 2)

La documentation d'Unreal Engine explique cependant que la méthode ne fonctionne qu'avec des casques. Le résultat peut être moins bon que la technique de panning avec des enceintes stéréo. (Epic Games, s. d., p. Spatialization Overview).

Techniques de sonorisations

Unreal Engine propose 4 méthodes pour générer une source audio en jeu :

- 1) Via les fonctions PlaySoundAtLocation, PlaySound2D, PlayDialogue2D et autres.
- 2) En créant une instance sonore via ces fonctions SpawnSoundAtLocation, SpawnSound2D, and SpawnSoundAttached.
- 3) En utilisant l'API PlaySound qui joue un son, mais ne permet pas d'en modifier dynamiquement ses propriétés.
- 4) Par l'utilisation d'audio component, permettant d'être attaché à un acteur et d'être modifié dynamiquement.

(Epic Games, s. d., p. Audio Engine Overview)

Additionnellement, pour la sonorisation de jeux, il est possible d'utiliser un logiciel tiers tel que Wwise qui possède un plugin l'intégrant dans Unreal Engine (Audiokinetic, s. d.).

Ambiances phoniques

Audio Shape

Pour changer d'ambiance sonore, il est possible d'utiliser les « Ambient Zones », une géométrie placée dans la scène et permettant la détection de la présence joueur. Elle modifie le comportement des sources à l'intérieur et à l'extérieur de sa géométrie. (Epic Games, s. d., p. Ambient Zones)

Les volumes audio servent quant à eux à appliquer des effets et définir le volume du son (Epic Games, s. d., p. Audio Volumes).

Pour détecter l'environnement dans Outcast 2, le studio Demute a utilisé deux techniques :

- Déclencher un événement quand survient un changement de biome.
- Utiliser des audio volumes pour les micro-biomes et sonoriser des villages. Lors de la conférences, Anthony Deneyer précise que leurs placements ont été chronophage, car ils ont été réalisés manuellement.

Les audio shapes du moteur Unreal Engine ont d'abord été utilisés pour segmenter les ambiances sonores du jeu « Solar Ash Kingdom ». En raison de problèmes de performances, les conférenciers ont migré de cette logique vers des valeurs d'atténuation directement disponibles sur les composants audio. Lors de la présentation, ils expliquent également utiliser un système de spline permettant de déterminer la position du joueur le long d'un niveau pour appliquer des effets et du layering sur la musique. (Unreal Engine, 2022)

Cette approche est aussi utilisée dans les jeux « GTA IV et GTA V » sous le nom de zones audio. Elles sont placées manuellement sur la carte. Lorsque le joueur les traverse, des sons se déclenchent aléatoirement dans le volume. (MacGregor, 2014).

Sonorisation environnementale par état

Pour la sonorisation d'ambiance, il est possible d'utiliser des états. (Audiokinetic, s. d.)

Comme l'écrit Robert Ciesla, les états permettent des changements globaux de propriétés dans le mix. (Ciesla, 2022, p. 249)

Unreal Engine propose également un système de tag gameplay permettant de changer d'état par script ou par « Trigger Volume ». (Epic Games, s. d., p. Soundscape)

C'est ce qu'utilise le plugin Soundscape. Ce système est très similaire à ce que décrit MacGregor avec la génération de sources dans un volume.

Soundscape est exploité dans le projet Electric Dreams.

Les ambiances sonores sont diffusées en quadriphonie pour laisser paraître l'environnement plus spacieux. Les sources sont instanciées procéduralement autour du joueur par sélection de groupe. (Epic Games, 2023)

Soudscape utilise des fichiers « Couleurs » et « palettes » pour grouper les fichiers audios et les instancier (Epic Games, s. d., p. Soundscape)

Electric Dreams possède des palettes Soundscape comme les oiseaux, insectes, grenouilles et feuillages permettant de jouer des sons plus proches ou éloignés en entrant dans des zones plus ouvertes. Comme l'explique Dan Reynolds, le système Soundscape sélectionne les ambiances sonores à l'aide de tags gameplay permettant de choisir l'état qui le déclenchera ou l'exclura. (Epic Games, 2023)

Sources audio localisées

Sources ponctuelles

Il est possible de placer manuellement des « Ambient Sound Actor » et des sources audios dans la scène (Epic Games, s. d.).

Cette approche manuelle peut rester nécessaire même s'il y a une automatisation du placement.

Lors de sa présentation, Anthony Deneyer explique que l'environnement peut changer, pouvant générer une désynchronisation des sources sonores et de leurs visuels (Game Camp, 2023).

Sources dynamiques

Les sources sonores peuvent avoir des modulations appliquées en temps réel.

Le projet Electric Dream utilise le plugin Metasound. Comme le montre John Tennant, il est possible d'y définir des variables et de les contrôler en Blueprint.

Dans son cas, il s'agit de différents paramètres d'intensités et du type de matériel détecté pour la sonorisation de pneus de voiture sur le véhicule du projet Electric Dreams.

Dans sa présentation, il montre l'affichage de textes changeant aux variations audios lui ayant permis d'identifier des problèmes et de déboguer son système.

Dans l'exemple d'ambiance quadriphonique de Dan Reynolds, les canaux audios subissent une rotation inverse à celle de la caméra pour suivre une rotation ancrée dans l'environnement. (Epic Games, 2023).

La nature du son n'est pas la seule à pouvoir être dynamique. Si elle est spatialisée, la source audio peut aussi se déplacer.

Un autre outil développé par Demute est celui de sonorisation d'une rivière utilisant le système de « Spline » dans Unreal Engine. Cette courbe est éditable par point. Ce système utilise aussi des RTPC (Real Time Parameter Controls) pour ajuster des paramètres du son par rapport à la largeur de la rivière.

Lors de l'implémentation de la technique, situer la source sonore la plus proches du personnage n'a pas été suffisant. Pour parvenir à la sonorisation de rivière, des points de contacts sont définis sur la courbe permettant de déterminer la dispersion du son. La rotation de la caméra devient importante car les points de contacts sont déterminés par la distance d'une position projetée dans le sens de la caméra. Par ce décalage, la zone observée est mise en valeur par rapport à celle qui se trouve le plus près du personnage.

Automatisation de sources audio

Automatiser le placement de source sonore permet d'avoir un retour immédiat sur les changements de la production. (Unreal Engine, 2022).

La structuration en amont des projets permet, dans certains cas, de s'affranchir de la répétitivité des tâches. Dans le cas de réalisations plus complexes, il peut être productif d'implémenter des outils d'automatisation.

Pour limiter le nombre d'audio joués en même temps, Unreal Engine propose les « Concurrency Assets ». Ils permettent de rendre muet les audio joués en trop sur base de leurs durées jouées et de leurs distances. (Epic Games, s. d., p. Sound Concurrency Reference Guide).

Dans le cas de Demute dans Outcast 2, les sons sont même détruits quand ils sont trop loin, en plus de limiter le nombre de voix pour des raisons d'optimisations. (Game Camp, 2023)

Placement de sons par objet

Il est possible de déposer une source sonore dans le niveau et de gérer ses paramètres manuellement (Epic Games, s. d.)

Il est également possible d'attacher la source à des objets pour ne pas répéter l'action.

L'attachement peut être réalisé de manière manuelle à des collections appelées Blueprint dans le moteur Unreal Engine.

Dans un exemple du livre (Stevens & Raybould, 2016), un son d'oiseau est attaché à un Blueprint d'arbre. En modifiant le fichier, chaque instance posée dans le niveau possèdera le son.

L'approche de B. Bertrand (Bas Bertrand, 2024) est d'attacher des sound cues aux arbres en temps réel.

Enfin, il est possible d'instancier des sources audios sur les objets dans l'éditeur.

Lors de la conférence concernant Outcast 2, Anthony Deneyer explique le processus d'automatisation de sonorisation des arbres. Un outil développé par le Demute détecte les objets statiques dans l'éditeur et leurs instancie des sources audios sur base de données associées aux data assets.

La méthode est attribuée à la documentation du projet Outer Worlds par le studio Obsidian Entertainment (Inc, 2019)

Placement automatique

L'attachement peut également être automatisé par un système générant le placement des instances sonores autour du joueur.

(Epic Games, s. d.) propose le plugin « Soundscape » utilisant des tags pour sélectionner l'ambiance sonore désirée. Dès lors, il est possible de déterminer la fréquence avec laquelle l'audio est répété en y ajoutant des paramètres aléatoires. Le système place des points aléatoirement autour du joueur en les déposant sur la surface des objets alentours.

Dans l'implémentation de (Rhys Anthony, 2023), une sphère de collision détecte le nombre d'élément de végétation pour déterminer l'ambiance sonore sélectionnée.

La technique de (Rhys Anthony, 2024) utilise des outils pour placer des points de sonorisation dans la scène par rapport au type de décors.

Dans la méthode de (Weaver,D, 2023), des rayons de collisions sont envoyés dans la scène afin de déterminer le déplacement des sources. Il lance des rayons pour détecter le sol en contrebas et y déplacer le son jusqu'à l'impact.

Logique modulaire

Dans la conférence du jeu « Solar Ash Kingdom » (Unreal Engine, 2022), les conférenciers expliquent que leur implémentation audio devait s'adapter aux modifications pendant la production.

Ils mentionnent également qu'ils ont créé leurs outils via des « component » interchangeables permettant de simplifier les données reçues en « alpha » et d'itérer rapidement.

Ils expliquent que diviser la logique leurs a permis de changer aisément les paramètres avec la proximité d'éléments environnementaux.

Audio adaptatif

RTPC

Les RTPC (Real-Time Parameter Controls), sont des paramètres dans le logiciel Wwise. Les RTPC peuvent être mappés sur des objets audios comme des bus, effets et sources dans le but de moduler leurs propriétés. Les RTPC peuvent être spécifiques à une source audio ou globale et affecter toutes les sources l'utilisant (Audiokinetic, s. d.).

Un exemple donné est la modification de la tonalité et du volume d'un moteur de jet mappé pour correspondre à sa vitesse (Ciesla, 2022, p. 249)

Custom RTPC

Solar Ash Kingdom utilise un système similaire au RTPC.

Pour homogénéiser les données du jeu, ils transforment les booléens et enums en float qu'ils nomment « alpha » et qui servent de pourcentage. Une fois ces variables établies, elles peuvent être combinées pour créer des portes.

Afin de créer des transitions dans le système audio, ils exploitent une interpolation sur les « alphas ».

C'est lors de l'utilisation de « l'alpha » que la valeur est remappée pour par exemple être convertie de pourcentage en Hertz sur une gamme de 0 à 22 000Hz. (Unreal Engine, 2022)

Pour manipuler les « alphas », leurs approche passent par des composants permettant d'adapter la logique aux situations non-linéaire de leurs jeux.

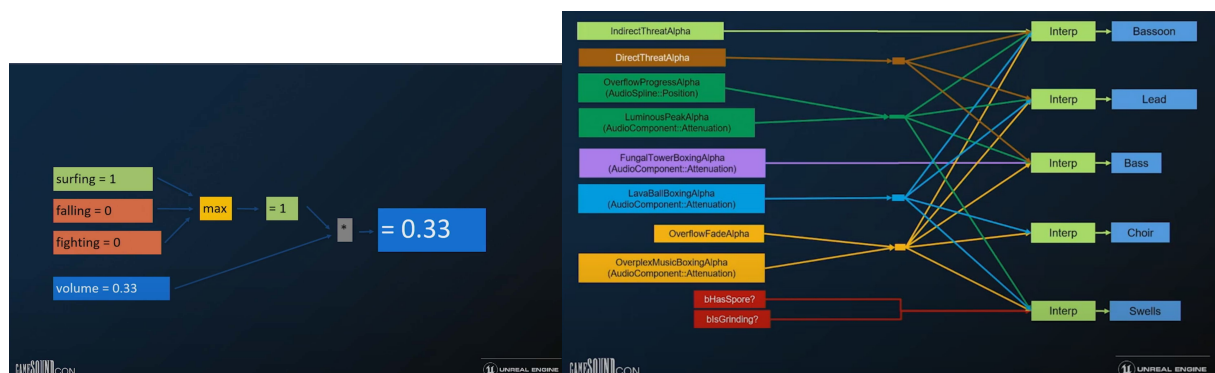


Figure 3 : Diagrammes extraits de la présentation "Modular Audio Systems in 'Solar Ash Kingdom' | GameSoundCon 2022".

Rage Audio procède d'une manière équivalente.

Les concepteurs les appliquent sous forme de formules composées de diverses variables permettant de moduler les sons.

Les sound designers ont accès à une liste de formules et à des variables exposées.

Des paramètres sont exposés directement depuis le moteur de jeu en fonction des classes les

utilisant. Ces paramètres permettent de prendre des éléments dynamiques en compte tel que la vitesse de déplacement d'un objet, permettant de faire varier le son.

Il y a également des variables globales contenant les informations pouvant être utilisées par plusieurs éléments tels la position du joueur et la météo.

Modular Sound Hierarchies



Figure 4 : Image extraite de la présentation "The Sound of Grand Theft Auto-V, 2014".

Lors de la conférence, le workflow y est abordé pour mettre en lumière une amélioration. L'outil appliquant la modulation audio s'est transformé en boîte à outils avec des formules génériques combinables. Le système a permis de la réutilisation tout en offrant des résultats plus rapidement (MacGregor, 2014).

Audio adaptatif

Pour rendre l'environnement cohérent, il est possible de créer des règles de sonorisation. Dans sa vidéo Bas Bertrand, explique que le nombre d'arbres défini l'ouverture dans l'espace autour du joueur. Avec cette technique, il change l'intensité de bruit de vent, d'insectes et d'oiseaux (Epic Games, s. d.).

Lors de la conférence d'Alastair MacGregor, le concept est poussé plus loin encore. Il évoque que l'environnement répond aux sons bruyants dans Grand Theft Auto-V. les objets susceptibles de résonner sont tagués avec leurs propriétés permettant de générer du bruit en conséquence. Il cite notamment l'exemple de bruits de tirs, effrayant des oiseaux, faisant aboyer des chiens ou résonnant dans des objets métalliques autour du joueur. (MacGregor, 2014)

Outils éditeur

Création et modulation d'assets sonores

Rage Audio possède un DSP permettant la création de contenu procédural.

Divers paramètres peuvent être modulés en combinant des outils de synthèse de son en temps réel. Le but de l'outil est d'encourager à expérimenter tout en visualisant les connections faites dans l'éditeur nodal. Une fois le son et ses modulations réalisés dans le système nodal, ils effectuent des tests pour identifier le cout en performance de la génération du son et en automatiser le contenu. En mode debug du jeu, les données des pistes de lecture sont traquées pour identifier si elles dépassent un seuil d'intensité et doivent être reporté aux sound designers.

Outils de création de données

L'outil « Static Mesh Emitters » développé par Demute Studio possède une fonctionnalité permettant de créer des data assets pour les assets sélectionnés.

Par la suite, un outil complémentaire a été développé pour le remplissage de ces données en liant la source de type « static meshes » aux audio associés.

Le projet Electric Dream possède aussi un élément de management des données en niveau. L'outil permet de lancer simultanément une fenêtre et de taguer plusieurs acteurs grâce à un Utility Widget et grâce à une table tag. Ensuite, les acteurs du même tag peuvent tous être sélectionnés en même temps via ce même outil.

Placement manuel facilité

L'implémentation de (Rhys Anthony, 2024) permet de visualiser les modifications faites dans l'éditeur. Il automatise le placement de points de sonorisations via un outil détectant les collisions dans l'éditeur.

L'outil « Static Mesh Emitters » permet également de faciliter les modifications manuelles.

Notamment des décalages de positions par rapport aux pivots des objets.

Pour pallier les modifications manuelles dans le niveau, une notification est réalisée lorsqu'une source n'est plus à l'endroit assigné. (Game Camp, 2023).

Récapitulatif

Grâce aux sources, on peut synthétiser les points forts des outils par projet sur le plan de la sonorisation d'environnement de jeu.

| | Rage Audio | MetaSound a montrer dans l'état de l'art Graph Digital Signal Processing | Electric Dream | Solar Ash Kingdom | Static Mesh Emitters Demute Unreal Audio Tools |
|---|--|---|---|---|---|
| Utilités | <ul style="list-style-type: none"> ◦ Gestion de la sonorisation ◦ Sonorisation par zones d'ambiances ◦ Mixage des pistes ◦ Variation audio temps réel ◦ Création d'audio procédurale DSP | <ul style="list-style-type: none"> ◦ Création d'audio procédural ◦ Variation audio temps réel | <ul style="list-style-type: none"> ◦ Génération d'ambiances spatialisés par zone ◦ Variation dynamique de sons | <ul style="list-style-type: none"> ◦ Module audio adaptatif pour les situations de jeu et de gameplay | <ul style="list-style-type: none"> ◦ Automatisation du placement de source statique |
| Méthode de génération des données | <ul style="list-style-type: none"> ◦ DSP temps réel ◦ Synthèse temps réel ◦ Utilisation de paramètre similaires aux "RTPC" ◦ Formules appliquant des effets ◦ Routing audio | <ul style="list-style-type: none"> ◦ DSP temps réel ◦ Synthèse temps réel ◦ Modification en temps réel des inputs metasounds ◦ Combinaison avec les bus audio submix | <ul style="list-style-type: none"> ◦ Soundscape couleurs et palettes ◦ Metasounds patches ◦ Tags gameplay | <ul style="list-style-type: none"> ◦ Composants mixant des valeurs d'alpha par effet ◦ Variable liés à l'environnement et au gameplay | <ul style="list-style-type: none"> ◦ Data Asset liant static meshes et source sonore ◦ Bouton d'automatisation de placement en scène |
| Modularité de l'outil | <ul style="list-style-type: none"> ◦ Déclenchement de source audio séquentiel ◦ Formules appliquant un effet audio | <ul style="list-style-type: none"> ◦ Appel d'évènements ◦ Modification de variables en temps réel | <ul style="list-style-type: none"> ◦ Adaptation au contextes sonores via les tags gameplay | <ul style="list-style-type: none"> ◦ Modularité par acteurs ◦ Formules modulaires par effet audio | <ul style="list-style-type: none"> ◦ Ajout de données dans le data asset pour des offsets ◦ Liaison entre audio et type de mesh configurable |
| Gestion des modifications et vérification des données | <ul style="list-style-type: none"> ◦ Test automatisé des couts en performance ◦ Rapport de dépassement de volume audio des source | <ul style="list-style-type: none"> ◦ Possibilité d'écrire dans les logs depuis Metasound | <ul style="list-style-type: none"> ◦ Debug visuel des sources spatialisés ◦ Liste des sons joués par commandes dans l'éditeur ◦ Edition des paramètres des palettes et couleurs Soundscape | <ul style="list-style-type: none"> ◦ Affichage de text et de primitive de debug | <ul style="list-style-type: none"> ◦ Notifications de désynchronisation entre les visuels et leurs sources audio attirés |
| Flexibilité des outils | <ul style="list-style-type: none"> ◦ Son procédural par source ◦ Formules par type de son appliquant des effets ◦ Formules et effets combinables | <ul style="list-style-type: none"> ◦ Son procédural par source ◦ Possibilité d'ajout de fonctionnalité en C++ ◦ Possibilité d'ajout de variables et d'évènements | <ul style="list-style-type: none"> ◦ Placement automatique des sources audio par Soundscape autour du joueur | <ul style="list-style-type: none"> ◦ Component par système ◦ Liberté de mixage et d'interactions entre variables sur l'audio | <ul style="list-style-type: none"> ◦ Interaction avec les meshes statique dans les scènes ◦ Ajout de fonctionnalité dans l'outil d'instanciation de sources. |
| Expérience utilisateur | <ul style="list-style-type: none"> ◦ Prévisualisation des sons joués dans les nœuds de logique via un raccourcis ◦ Feedback visuels d'ondes sur chaque lien nodal ◦ Hiérarchisation des effets par type de son permettant d'accélérer le workflow | <ul style="list-style-type: none"> ◦ Simulation des événements dans l'éditeur ◦ Feedback des liens activés ◦ Feedback de jauge de volume graduée ◦ Création de fonctions réutilisables dans plusieurs sources | <ul style="list-style-type: none"> ◦ Interface d'édition des palettes Soundscape | <ul style="list-style-type: none"> ◦ Réutilisation des component et paramètres "alpha" ◦ Automatisation de sonorisation de feuillage | <ul style="list-style-type: none"> ◦ Automatisation de sonorisation de feuillage ◦ Outil générant des data assets ◦ Outil complétant les données des data assets |

Figure 5 : Réalisation personnelle d'un tableau comparatif

4. Devis méthodologique

Méthodes

La première étape consiste à analyser les techniques d'implémentations audio présentes dans les projets. Des parallèles existent entre les méthodes en raison de l'utilisation du moteur Unreal Engine 5 pour chacun des prototypes.

Le fait d'avoir participé aux choix créatifs et à la résolution de leurs problèmes, me donne un accès direct au code source qui peut alors être adapté. Ces jeux sont pour la plupart publiés et aucun NDA n'a été réalisé, ce qui laisse une transparence totale dans la documentation des ressources.

Pour toutes ces raisons, la technique d'observation participative est exploitée de manière rétroactive (Soule, 2007).

- L'analyse des problèmes et solutions trouvés en groupe lors de ces projets permettra d'identifier les besoins spécifiques à chaque projet, tant dans l'automatisation par rapport au placement de sources sonores et qu'à leurs mixages.

Cette section permettra alors d'identifier les méthodes, leurs implémentations dans les jeux vidéo analysés, ainsi que de les comparer aux méthodes de l'industrie.

Finalement, une section auto-ethnographique concernant le processus d'implémentation sera réalisée en utilisant l'outil développé. Celle-ci comportera une analyse des rédactions réflexives des sessions de développement (Flamme, 2021).

Le but de cette analyse sera de déterminer l'impact de l'outil sur la production et son efficacité.

Trois jeux sont mis en situation :

- Parkware : Un jeu de parkour en première personne en phase de prototypage. L'outil sera testé pour appliquer des modifications dynamiques sur l'ambiance sonore de manière similaire à Solar Ash Kingdom.
- Explorer : Un prototype réalisé dans le cadre du master de jeu vidéo d'Albert Jacquard. C'est un jeu d'exploration en première personne, l'outil sera testé avec un cycle jour-nuit.
- Friends Don't Ghost : Un jeu inspiré de collecte de personnage à la troisième personne. L'outil enrichira l'ambiance phonique par rapport au score du joueur.

Outil développé

Les fonctionnalités de l'outil permettront de placer des sources audio dans la scène tel que le prototype de Demute l'applique (Game Camp, 2023). En recréant cette structure, il sera possible de la lier à un outil plus large.

Pour offrir plus d'interactivité avec l'outil, un mode éditeur sera développé avec une interface de contrôle de placement audio. Dans ce cas-ci, les outils du mode éditeur seront liés pour tester leurs efficacités.

Tel que le montre John Tennant (Unreal Engine, 2023), le debugging est un élément important dans l'implémentation d'audio. Aussi, des outils de prévisualisation seront implémentés. Pour la sélection d'objet, un outil de tags dans le projet Electric Dream est utilisé (Epic Games, 2023). Il sert au système de génération procédurale mais peut également être utilisé pour automatiser la sélection de type d'acteur dans la scène.

La deuxième fonctionnalité majeure simule le comportement des paramètres « alphas » exposés dans la présentation sur le jeu Solar Ash Kingdom (Unreal Engine, 2022). L'accès à cette fonctionnalité sera accompagné d'un acteur permettant le mixage des valeurs et permettant la création de formules inspirées du moteur Rage Audio (MacGregor, 2014). Ce comportement n'est pas sans rappeler les tables de mixages virtuelles disponible dans les DAW (Digital Audio Workstation). De manière similaire, un système existe dans les matériaux d'Unreal Engine sous le nom de « Material Parameter Collection ».

Une façon d'exploiter ces données est de les centraliser pour les rendre disponibles dans les acteurs avec des « DYNAMIC_MULTICAST_DELEGATE » pour les mettre à jour. Complémentairement, les valeurs « alpha » doivent être accessible dans le système Metasound d'Unreal Engine.

Soundscape sera également exploité car son système est suffisamment modulaire que pour être combiné avec de la logique Blueprint.

L'outil repose sur 2 aspects principaux :

1) L'automatisation de placement doit être extensible à l'ajout de règles et à leurs modifications. C'est notamment ce qui est conseillé dans le système Soundscape en C++ (Epic Games, s. d.) Pour les besoins du projet, le placement se fera par des techniques de détection de l'environnement (Bas Bertrand, 2024). Si des sons doivent être dynamiques, c'est la technique déplaçant les sons par calcul de leur trajectoire qui sera utilisée (Weaver,D, 2023). Le mixage par atténuations fera également partie du prototype (Stevens & Raybould, 2016).

2) Les outils doivent être accessibles et rapides à itérer/tester.

Ils seront réglables par des interfaces basées sur (Epic Games, 2023) et des outils dans l'éditeur de niveau permettant de modifier les scènes, comme le montre (Rhys Anthony, 2024) et comme l'explique (MacGregor, 2014).

5. Présentation du prototype

Structure

Le prototype se compose d'un plugin développé pour Unreal Engine 5.4.

Unreal Engine possède une structure par module permettant de séparer les dépendances entre les scripts. Ce plugin a été nommé « SoundHolder » en référence aux nombreux projets auxquels j'ai participé et dont les sons implémentés étaient en « Place Holder » en attente de fichiers audio plus aboutis et en attente de la finalisation de l'implémentation.

SoundHolder est composé de 3 modules distincts :

- 1) SoundHolder, dont le but est de contenir les classes acteurs et composants utilisés pour la sonorisation.
- 2) SoundHolderEditor, qui contient des commandes pour un mode éditeur permettant de sélectionner des objets et de créer des interactions entre scène et interface outils. C'est ce module qui gère les outils éditeur qui ne seront pas inclus lors de la compilation finale du jeu.
- 3) « SoundHolderMixersubsystem », la plus importante des classes qui est responsable de lier toutes les variables d'entrée et de stocker les « alphas ».

Un subsystem est un objet statique. Il s'agit d'un enfant de « UGameInstanceSubsystem ».

Son temps de vie est équivalent à la durée du jeu et il n'est pas exécuté dans l'éditeur.

Il dispose également d'une classe permettant le mixage des variables via une interface Blueprint.

Le défi du système est de simuler les RTPC et de capturer plusieurs types de données. Le subsystem se compose de « FInstancedStruct » disponible depuis le plugin « StructUtils ». Une dizaine de Structs ont été implémenté en C++ pour permettre de passer des instances Struct dans les fonctions et de les encoder et décoder en Blueprint.

Avec ce système, il est possible de prendre comme paramètres d'énumérations, des booléens, des nombres entiers, des valeurs flottantes, des vecteurs, des rotations et leurs tableaux de valeurs.

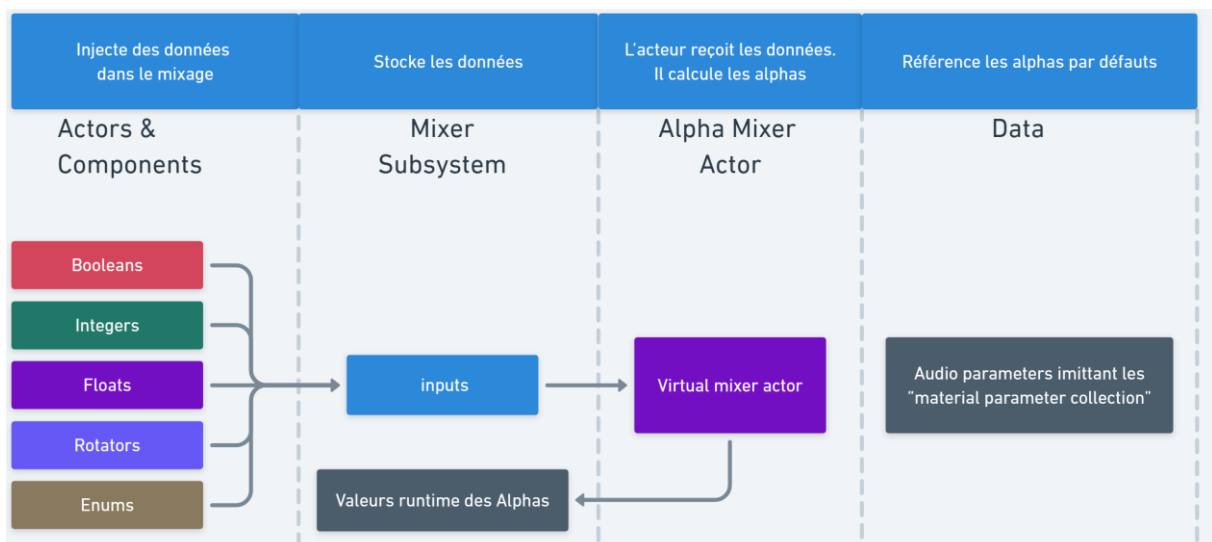


Figure 6 : Réalisation personnelle d'un schéma du système de mixage des variables et paramètres « alphas ».

Accessibilité

Le plugin possède également un dossier « Content » contenant les outils Blueprint tels un « Widget Utility Blueprint » et un « ObjectMixerBlueprintObjectFilter ».

Le Widget sert au placement des sources sonores dans les niveaux. Une fonctionnalité tirée du mode éditeur permet de récupérer les objets sélectionnés via un rayon tiré vers le curseur. Cette sélection permet de récupérer des composants ou acteurs. Le widget permet ensuite d'appliquer des effets à toutes les instances de ces objets.

L'ObjectMixerBlueprintObjectFilter réalise la visualisation des sources et de leurs propriétés, permettant avantageusement d'éditer les paramètres affichés.

Le plugin comporte également un dossier Blueprint contenant des acteurs génériques hérités des classes en C++. C'est par exemple le cas de l'acteur « BP_AlphaMixer » qui hérite d'une classe C++ et qui peut être instancié par le subsystem « SoundHolderMixer ».

Pour permettre de la flexibilité, les paramètres du mixer peuvent être remplacés en jeu par des appel de fonctions. Pour leur permettre d'être initialisés dès le lancement du jeu, il est possible de définir des variables par défaut et des références dans le menu de paramètre du projet.

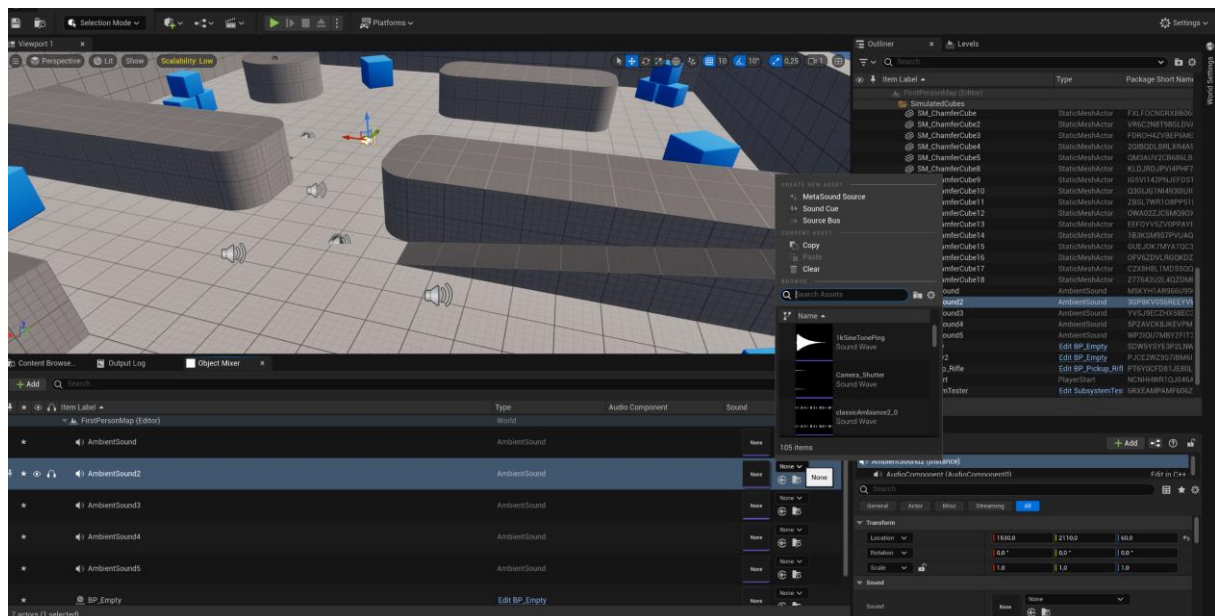


Figure 7 : Réalisation personnelle d'un affichage d'un l'ObjectMixerBlueprint développé pour la visualisation de sources audio.

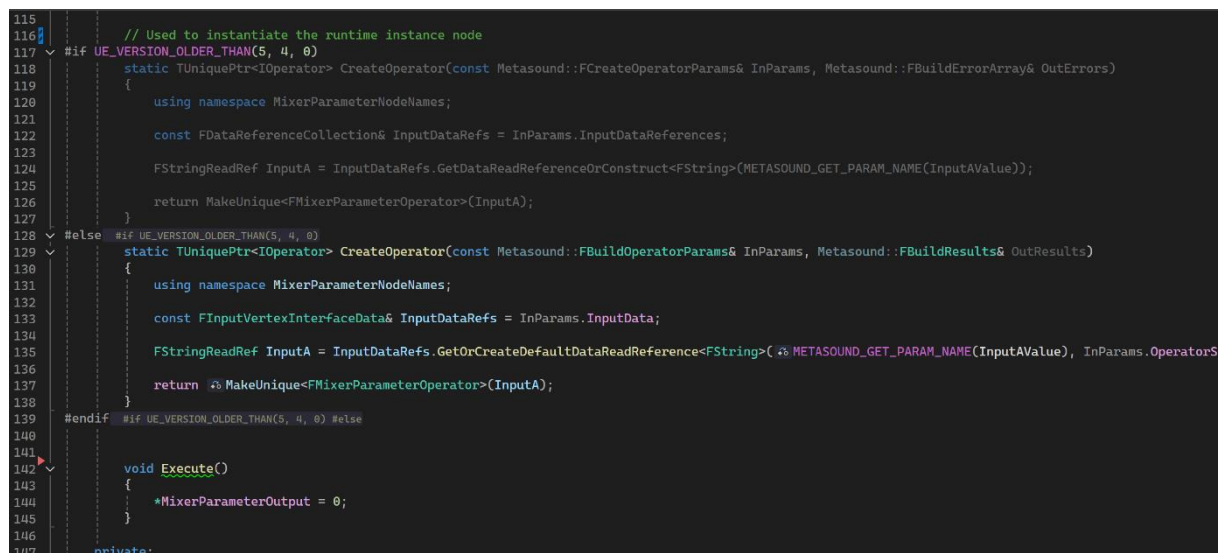
Challenges

Le moteur Unreal Engine 5 possède une documentation conséquente pour décrire les concepts des systèmes. Depuis Visual Studio, il est possible d'explorer le code source des modules tels que Soundscape et Metasound. Le code source du moteur est aussi disponible sur Github moyennant un enregistrement avec un compte Epic Games.

Deux ressources supplémentaires sont :

- Le site web de la communauté avec une section apprentissage et tutoriels.
- Le code template disponible dans la génération de plugins et d'acteurs.

Toutes ces sources ont été exploitées pour la réalisation de l'outil. Elles permettent de croiser les informations nécessaires au développement de fonctionnalités mais également de représenter changement entre les versions d'Unreal Engine. Un exemple est le cas du plugin Metasound qui possède du code qui n'est pas maintenu entre les versions. La consultation du code source devient un atout majeur dans cette situation.



```
115 // Used to instantiate the runtime instance node
116 #if UE_VERSION_OLDER_THAN(5, 4, 0)
117 static TSharedPtr<IOperator> CreateOperator(const Metasound::FCreateOperatorParams& InParams, Metasound::FBuildErrorArray& OutErrors)
118 {
119     using namespace MixerParameterNodeNames;
120     const FDataReferenceCollection& InputDataRefs = InParams.InputDataReferences;
121     FStringReadRef InputA = InputDataRefs.GetDataReadReferenceOrConstruct<FString>(METASOUND_GET_PARAM_NAME(InputAValue));
122     return MakeUnique<FMixerParameterOperator>(InputA);
123 }
124 #else
125 static TSharedPtr<IOperator> CreateOperator(const Metasound::FBuildOperatorParams& InParams, Metasound::FBuildResults& OutResults)
126 {
127     using namespace MixerParameterNodeNames;
128     const FInputVertexInterfaceData& InputDataRefs = InParams.InputData;
129     FStringReadRef InputA = InputDataRefs.GetOrCreateDefaultDataReadReference<FString>(METASOUND_GET_PARAM_NAME(InputAValue), InParams.OperatorS
130     return MakeUnique<FMixerParameterOperator>(InputA);
131 }
132 #endif
133 #if UE_VERSION_OLDER_THAN(5, 4, 0)
134 void Execute()
135 {
136     *MixerParameterOutput = 0;
137 }
138 private:
```

Figure 8 : Réalisation personnelle du code de création de nœud Metasound dans Visual Studio 2019.

Un autre élément challengeant a été de lier l'éditeur mode à un Widget Utility Blueprint. Heureusement, lors de la création de plugin, il existe une possibilité de sélectionner un Template Editor. La mise en œuvre des fonctions est néanmoins complexe à comprendre et à implémenter.

Des commandes s'enregistrent dans une classe boîte à outil (toolkit). Ensuite les fonctionnalités sont référencées dans ces classes pour être identifiées et générées.

Pour mon implémentation, la complexité de mise en œuvre du mode éditeur limite son usage à la sélection des objets et à la connexion à une interface.

Concernant l'extension de ses fonctions, il était plus simple de gérer la logique de génération depuis le widget. L'approche utilisée par le studio Demute est pertinente et permet une grande flexibilité d'itération et d'extension (Game Camp, 2023).

6. Analyse réflexive du prototype

Le prototype Explorer

Explorer est un prototype de jeu d'exploration en première personne développé dans le cadre du master de jeu vidéo de la HEAJ. Ce projet arrivait vers la fin du prototypage quand il a été abandonné en raison de contraintes narratives et de game design.

Pour garantir le succès du jeu, nous comptons sur l'immersion des joueurs dans l'environnement :

- via l'intégration d'un cycle jour-nuit dans les mécaniques du jeu,
- en amplifiant l'importance de l'environnement pour rendre le joueur mal à l'aise, voire générer un sentiment d'effroi.

Pour des raisons de prototypage, le cycle jour nuit n'existait que dans le concept. Le but du jeu est d'observer la faune et la flore. Le joueur y incarne un aventurier utilisant la démarche scientifique qui note ses observations dans un carnet. L'environnement est constitué d'une plage et de grottes.

Comme écrit dans le rapport de production d'Explorer (Microwave Interactive, 2024a), « Il y a très peu de visuels mais l'environnement est assez développé, permettant de laisser paraître plus de diversité de contenu. Le mode de sound design visé est le style designé.

C'est-à-dire de combiner des sonorités réalistes tout en dépayasant le joueur avec des éléments non-réalistes. Pour renforcer le sentiment de progression par la compréhension sonore de l'environnement, nous avons opté pour l'utilisation de passage au sound design "horrifique". En ce sens, la base du son reste réaliste et "designée" pour construire une sonorité de tension exploitant notamment le bruit. »

C'est en partie les playtests qui nous ont encouragés à exploiter cet aspect de tension apporté par le bruit du vent dans la cave. L'environnement est le focus du jeu et le joueur doit se sentir immergé dans un monde plus vaste que sa caméra ne laisse le supposer.

Méthodes d'implémentations

L'implémentation effectuée dans ce prototype est quelque peu simpliste mais néanmoins fonctionnelle. Une dizaine de sources audios spatialisées sont activées en boucle avec des variations. Elles sonorisent la mer, le vent sur la plage et dans les grottes.

Cependant, les objets récupérables bénéficient d'une implémentation plus technique. Ils possèdent des détections de collisions permettant la lecture de son. Lors de leur chute, ils effrayent les créatures et les notifient notre présence.

Le jeu utilise également des bruits de pas liés aux oscillations de la caméra et des bruits de vêtements générés lors du déplacement ou lors d'actions spécifiques.

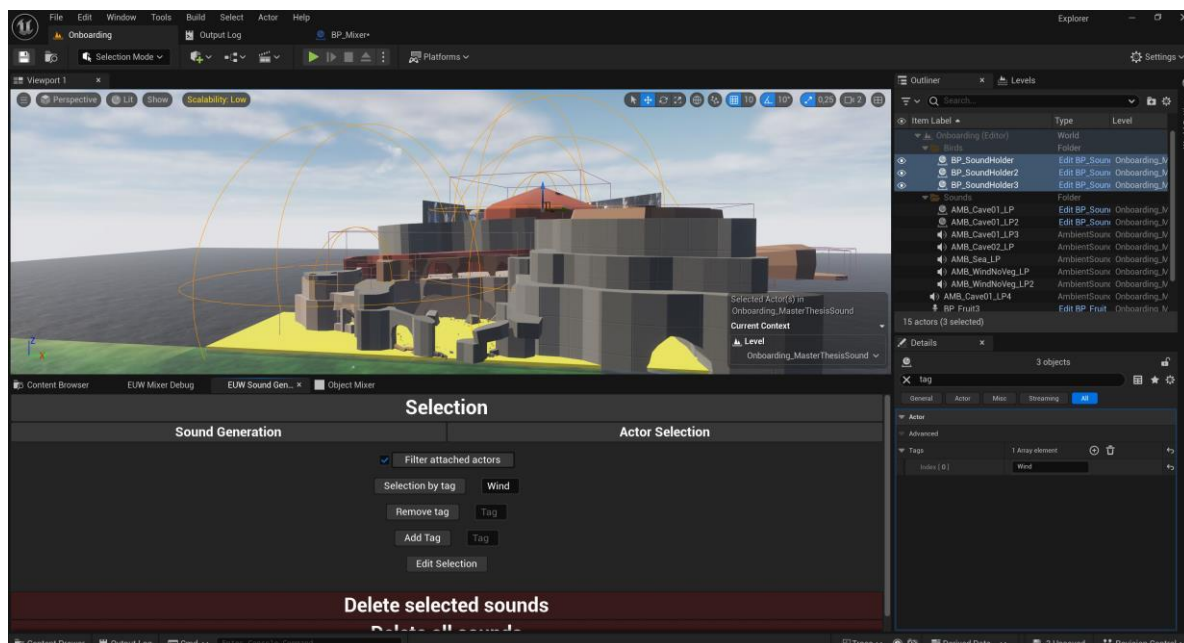


Figure 8 Réalisation personnelle de la sélection et de la génération de sources audio dans Explorer.

Sonorisation avec Sound Holder

Avant d'intégrer le plugin, des fichiers audio ont été ajoutés pour sonoriser des chants d'oiseaux et leurs vols, des amphibiens, des collisions, des bruits d'eau, des cris animaux au loin, ainsi que des bruits d'ambiance de cave.

Explorer est le premier prototype à avoir intégré le plugin « SoundHolder ». Cette première intégration dans un autre projet a permis de déterminer l'ordre des étapes clés à l'initialisation des outils :

- La première étape était de créer un Data Asset de Mixer Parameter Collection. Depuis celui-ci, sont définies les variables utilisées pour le mixage. Les variables créées consistaient en un pourcentage du temps de la journée en jeu et un paramètre de bruit.
- Ensuite, un acteur hérité de « Alpha Mixer Actor » était généré en utilisant sa fonction. Le cycle jour nuit a été implémenté en utilisant une de mes réalisations antérieures comme source d'inspiration : le prototype de jeu Fadeless (Microwave Interactive, 2024b).
- L'avancement dans le temps est identifiable par le déplacement du Soleil dans le ciel. Il se matérialise aussi par une barre au bas de l'écran. La valeur de temps est renseignée comme pourcentage dans les valeurs « Alphas ». Cette valeur est directement récupérée par des fichiers Metasound gérant la sonorisation.
- Les objets récupérables disposent désormais d'un fichier audio lu lors des collisions. Par la même occasion, ils augmentent une valeur de bruit définie dans les « Alphas ». L'acteur « Mixing Logic », permet de réinitialiser les paramètres de bruit après un délai défini.
- Lors de la modification de la valeur, des acteurs peuvent être notifiés. Au début du niveau, il est possible d'effrayer des oiseaux en faisant du bruit avec l'objet.
- Les sources placées dans le niveau ont été modifiées afin de récupérer l'information du moment de la journée et de créer une transition entre différents fichiers audios. Suivant cette logique, le vent est différent au lever du soleil ou en journée. De même, des amphibiens sont audibles en matinée. Cette interpolation audio est possible grâce à un nœud Metasound personnalisé en C++.

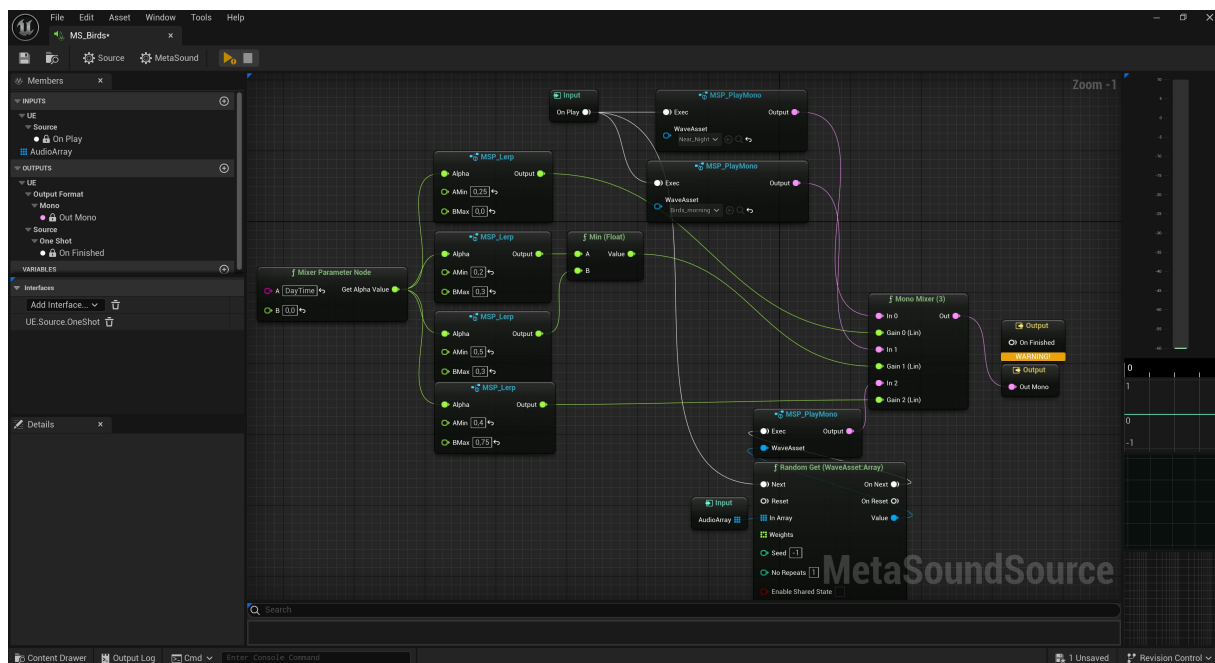


Figure 9 : Réalisation personnelle du graph Metasound des oiseaux en fonction du moment de la journée.

Un élément non-pris en compte est le peu de ressources 3D nécessaires pour le prototype. L'environnement est composé de cubes, de grands meshes pour les grottes et d'instances de brins d'herbe. Pour fonctionner, l'outil de placement automatique nécessite des meshes. Le placement n'a pas pu être réalisé automatiquement. Soundscape parait plus propice à sonoriser des environnements par type d'ambiance sans être contraint par la richesse de celle-ci.

L'outil de placement automatique n'a cependant pas été inutile. Avec la sélection et l'ajout de tags, l'outil a permis de créer des collections de sélections rapides. Il était possible de sélectionner des catégories de sons pour les éditer ou vérifier leurs positions. Des debugs peuvent être affichés pour les BP_SoundHolder sélectionnés.

Le jeu Friends Don't Ghost

Le dernier jeu a été réalisé à la Beginner's Jam Halloween 2024 en collaboration avec Yiri Zylka et Giulia Storino (Martins et al., 2025).

Il s'agit d'un jeu de collecte de personnage à la troisième personne. Le joueur y incarne un fantôme qui se fait « ghoster »¹ le jour de son anniversaire. L'objectif est d'accumuler tous les personnages de la scène pour réaliser la plus belle fête d'anniversaire possible. De ces trois projets, il est le seul avoir été publié.

L'outil enrichira l'ambiance phonique par rapport au score du joueur.

Méthodes d'implémentations

Le jeu a été développé pendant une semaine. Contrairement à ce qu'on pourrait croire, le rythme de développement était faible et constant. Cette occasion de travailler quelques heures par jour a autorisé un temps de recul et de simplification de la structure du projet.

Le prototype venant d'une Game Jam, des raccourcis sont faits dans l'implémentation dans le but de gagner du temps. On peut citer le cas des zombies qui possèdent une source audio continue qui règle elle-même le délai entre la lecture de son dans une liste.

L'implémentation sonore est simple et efficace avec une musique, des sons de feedbacks lorsque le joueur perd du score, ainsi que des gémissements aléatoires des ennemis permettant de signaler leur présence. Ce jeu a été élu troisième meilleur jeu dans la catégorie visuel et sonore parmi la soixantaine de jeux réalisés.

Sonorisation avec Sound Holder

L'utilisation du plugin a été rapide. En quelques heures les ambiances audios ont été intégrées. Les techniques utilisées sont similaires à Explorer. La musique subit des effets par rapport au score, dans ce cas-ci un effet Flanger. De manière globale, le score rétablit des fréquences aigües dans la musique. Lorsqu'il augmente, le score est aussi récupéré par des événements pour déclencher des bruits festifs.

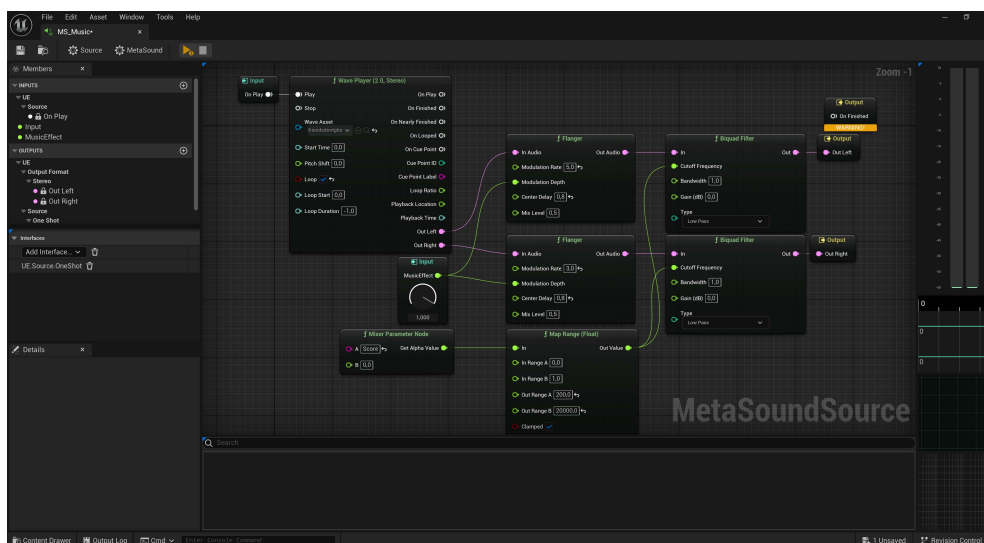


Figure 10 : Réalisation personnelle du graphique Metasound de la lecture de musique dans le jeu Friends Don't Ghost.

¹ Terme en argot signifiant le fait d'être ignoré.

L'ambiance sonore de la salle est aussi agrémentée de bruits de foule. Quand le score est faible, il y a très peu d'ambiance. Quand il augmente, la foule devient plus énergétique. Passé un seuil défini, elle s'atténue pour laisser place à la musique et à d'autres bruits de fêtes.

L'outil de placement automatique a été testé sur les cercueils. Il y en avait une soixantaine dans la scène, une coïncidence par rapport au nombre d'équipes ayant participé à la Game Jam. Le contexte du jeu étant dramatique, ce sont des bruits de coups frappés aux couvercles des cercueils qui ont été ajoutés. Pour rester dans le drame, le son de la fête est trop fort tous ne pourront pas y participer.

Malgré la grande densité d'éléments présents, l'édition des instances s'est effectuée en temps réel. L'outil a fonctionné au-delà de nos attentes.

Le prototype Parkware

Il s'agit d'un jeu de parkour à la première personne. L'emphasis sonore y repose sur l'immersion et le dynamisme. Le jeu utilise un sound design minimaliste mais adaptatif au déplacement du joueur. Le module de mixage d'« alphas » a été mis à l'épreuve avec une musique dynamique,

Parmi les trois jeux présentés dans ce mémoire, ce jeu, qui est encore au stade de préproduction, est pour l'instant le moins abouti.

Méthodes d'implémentations

A l'inverse des 2 autres jeux, Parkware (Fernandes Pereira & Martins, 2024) met le focus sur le personnage. Il ne s'agit plus d'un environnement vivant et crédible autour de celui-ci mais d'une simulation dans laquelle les joueurs se meuvent. Ce prototype ne contient pas encore de fichier audio.

Sonorisation avec Sound Holder

Pour tester le plugin, des paramètres liés au déplacement du joueur ont été utilisés pour reproduire les propositions du jeu « Solar Ash Kingdom ».

La musique est divisée en pistes et évolue selon les actions du joueur.

L'acteur de mixage récupère la vitesse, des états pour appliquer les « Alphas » qui modifient la musique.

L'environnement influence également l'audio, par exemple lorsque le joueur court sur un mur, la spatialisation du son peut-être effectuée en réponse. L'objectif principal a été d'intégrer la musique aux situations du jeu. Utiliser les « alphas » pour ajuster le volume des instruments a aussi été implémenté.

La mise en place du système a consisté à générer des composants permettant de capturer des propriétés autour du joueur, telles par exemple le nombre de sphères autour du joueur ou la taille du volume dans lequel il se déplace. Ainsi, les paramètres sont envoyés dans l'acteur de mixage qui retourne les « alphas ». Ces « alphas » sont alors récupérés dans une source audio Metasound et modifient le volume des pistes.

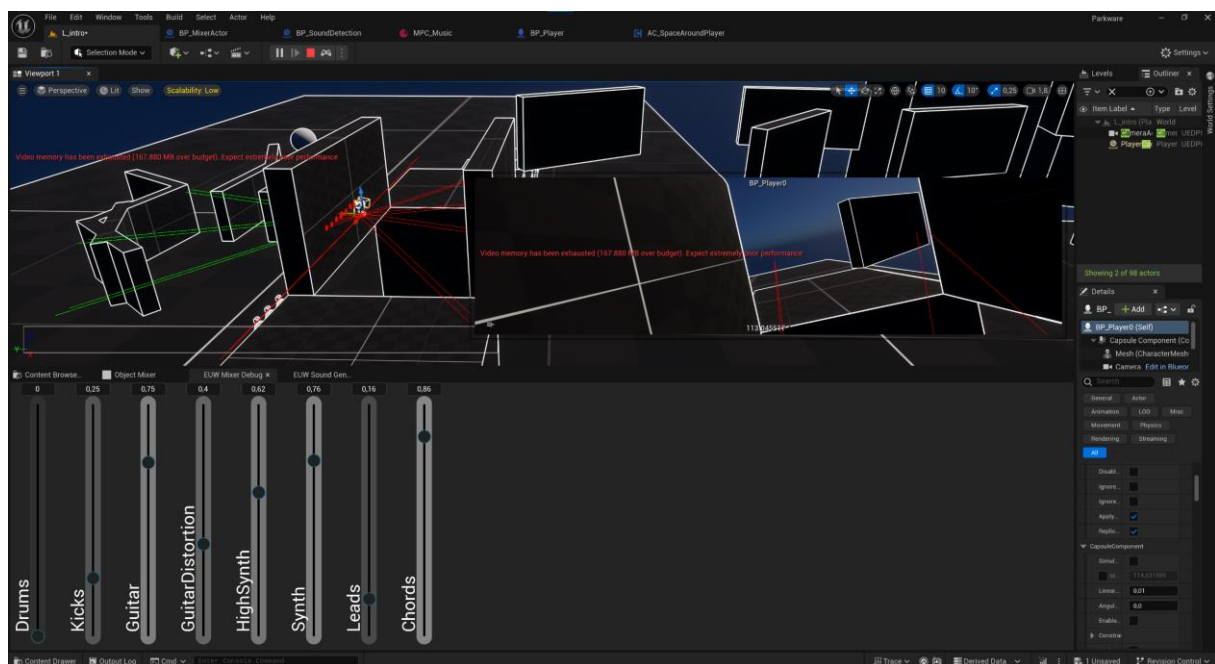


Figure 10 : Réalisation personnelle du mixage des pistes audio de la musique.

Dans cette figure, le joueur court sur un mur et des rayons sont envoyés permettant de déterminer l'espace visible.

Plusieurs limitations sont apparues par la suite. Par exemple, contrairement à Explorer, le choix d'un morceau de musique défini ne peut pas s'effectuer avec une valeur comprise entre zéro et un. Le system Quartz d'Unreal permet d'effectuer des fonctions en rythme. Dans le cadre de ce projet, la méthode a été testée mais n'a pas abouti en raison de la complexité de la tâche et en raison de la gestion difficile de toutes les pistes dans Metasound. Le plugin développé, Sound Holder, n'a pas pu remplir ce besoin.

| | Projet du prototypage | Explorer | Friends Don't Ghost | Parkware |
|--------------------------------|--|--|--|--|
| Gameplay | Shooter | Exploration | Collecte | Parkour |
| Type de vue | Première personne | Première personne | Troisième personne | Première personne |
| Objectif sonore | <ul style="list-style-type: none"> ◦Tester toutes les fonctionnalités simultanément | <ul style="list-style-type: none"> ◦Sources audio adaptatives au cycle jour nuit ◦Réaction au bruis : des oiseaux peuvent être effrayés par des bruits | <ul style="list-style-type: none"> ◦Rendre les sons adaptatifs au score ◦Rendre l'environnement plus riche en audio spatialisé ◦Appliquer des effets sur la musique ◦Appliquer plus de feedbacks audio pendant la prise de score | <ul style="list-style-type: none"> ◦Le déplacement du joueur et l'environnement doivent influencer la musique en jeu |
| Fonctionnalités testées | <ul style="list-style-type: none"> ◦Mixage ◦Audio réactif ◦Placement automatisé ◦Selection par tags ◦Mixage de valeurs "alpha" ◦Filtre d'édition éditeur ◦Interface éditeur de test de mixage | <ul style="list-style-type: none"> ◦Mixage dynamique sur la scène ◦Audio réactif ◦Placement automatisé ◦selection par tags | <ul style="list-style-type: none"> ◦Placement automatique ◦Environnement adaptatif au valeurs"alphas" ◦Selection par Tags | <ul style="list-style-type: none"> ◦Acteur de mixage ◦Valeurs "alphas" ◦Utilisation de composants ◦Détection environnementales |
| Flexibilité de l'outil | NA | L'ajout de fonctionnalité n'a pas été restreint par l'outil | Les variable d'alphas ont réduit le temps de développement, notamment avec l'utilisation du getter lié au subsystem | Mixer les paramètres n'est pas très intuitif quand il y a beaucoup de paramètres, La gestion des pistes a été difficile à définir en raison du nombre de fichier |
| Contrôle du placement | NA | Peu effectif dû aux manques de mesh | Le placement automatique est pratique et efficace | NA |
| Contrôle de selection | NA | Selection efficace par tags | Facilitation de l'édition de paramètres communs aux sources tel par exemple l'atténuation | NA |
| Contrôle de mixage | NA | Les sources se mettent à jour automatiquement via le nœud Metasound | Le mixage rend à la fois les sons réactifs au score mais également évolutif en fonction du score | Les pistes audio sont gérées par les alphas facilement utilisé avec Metasound |
| Réactivité audio | NA | L'audio permet des réactions : des oiseaux peuvent être effrayés au bruit | Les sons de fête sont devenu évolutif au score | Le mixage est réactif à la détection des acteurs autour du joueur |
| Limitation | NA | Le placement automatique ne s'applique qu'aux static mesh | Le placement automatique ne comporte pas de fonction aléatoire. Il y aurait pu avoir plus de contrôle pour filtrer un pourcentage d'acteurs dans la scène | Utiliser MetaSound pour une composition verticale peut prendre beaucoup de temps. L'arrangement horizontal de la musique est limité |

Figure 11 : Réalisation personnelle d'un tableau comparatif de l'utilisation de Sound Holder.

7. Conclusion

L'automatisation vient d'abord des techniques structurant les projets qui peuvent être facilitées par des outils comme celui de Demute (Game Camp, 2023), plaçant les sources audio. Pour garder la maîtrise, il est plus facile de diviser les outils en catégories. En ce sens, le menu de sélection pour les acteurs par tags a été très utile pour sonoriser Explorer.

La visualisation dans l'éditeur est également un élément important. Le prototypage d'interface dans l'éditeur a permis de visualiser l'origine de problèmes avant même d'en avoir entendu leurs conséquences. L'interface des « alphas » sous formes de jauge était particulièrement efficace en ce sens. Vers la fin du développement, l'ajout dans les paramètres du projet a permis de tester des comportements entre acteurs sans avoir à coder de nouvelles fonctions.

Le système de mixage a véritablement passé un cap à partir de la définition des variables par défaut dans les paramètres du projet. En effet, pour améliorer l'efficacité et la rapidité d'itération, il ne suffisait que de changer le Data Asset et l'acteur de mixages dans les paramètres du projet, permettant l'exploitation de fonctionnalités différentes.

Toujours dans le même système, les types de données génériques créés à partir de UStruct ont permis de centraliser les données dans une seule fonction en supportant plusieurs types de variables.

Lors du démarrage du prototypage, un acteur servait à initialiser le subsystem de mixage par des fonctions C++. Après une dizaine de tests utilisant différentes données, des instances oubliées ont généré des conflits. Après les valeurs modifiables par défaut du projet et leur édition à la souris avec l'interface de mixage, ces problèmes ont disparu. Il était alors directement possible de simuler des changements de valeurs « alpha » de manière manuelle.

Au fur et à mesure du développement des fonctionnalités, il a aussi pu être constaté que la découpe entre systèmes permettait de se concentrer sur l'efficacité de petites tâches sans avoir à gérer trop de communication entre acteurs.

Cette logique de découpe a permis de gagner beaucoup de temps dans le développement par réutilisation, notamment par l'utilisation des sub-Widgets. Mais également dans le debugging en isolant les systèmes voulus.

Diviser le code en 3 sub-modules permet aussi de séparer la logique éditeur de celle du jeu. Cette séparation a aussi permis de n'exploiter que les fonctionnalités nécessaires pour chacun des 3 projets.

Tous les outils n'étaient pas totalement pertinents pour chacun des jeux testés. L'outil le moins pertinent dans le jeu testé est l'apparition de sources sonores car il n'a quasiment pas été utilisé en dehors du jeu Friends Don't Ghost. En fonction de la taille des niveaux jouables et des ressources graphiques développées, les besoins d'automatisation dans le placement des sources ne sont pas identiques. Cette différence est explicable par la variété des environnements et leurs répétitivités.

Dans le cas d'Explorer, l'environnement est constitué d'une mer, d'une plage, d'un récif et d'une grotte concentrés en un espace restreint. La répétitivité d'Explorer réside dans la mécanique temporelle qui impacte toutes les sources.

A contrario, Friends Don't Ghost possède un environnement avec beaucoup plus de réutilisations.

L'outil a été évalué avec différents points de vue et pour diverses attentes.

Voici un tableau comparatif des utilisations du plugin Sound Holder.

Les limitations identifiées sont les suivantes :

- Les projets doivent posséder au moins une classe C++ pour intégrer le plugin dans la compilation finale du jeu.
- Les types de données créent des instances de structure de données. Cette allocation pourrait causer un problème de performance si beaucoup d'acteurs en génèrent au même moment.
- Les « alphas » sont affichés dans une interface. Cependant, leurs paramètres d'entrée ne sont pas visualisables en jeu. Afin d'offrir plus de contrôle et de prévisualisation des données, il serait possible d'étendre le mixeur actor pour produire un graphique via le système de Slate d'Unreal Engine personnalisable en C++. Une structure similaire au Behavior Tree pourrait visualiser les valeurs des entrées et sorties pendant l'exécution du jeu.

Pour conclure, les outils ont été relativement faciles à utiliser grâce au moteur qui possédait une grande diversité d'actions réalisables et automatisables. L'utilisation de celles-ci a été très rapide et efficace en regard de la complexité des objectifs. Dans le cadre des trois jeux testés, ces outils sont bénéfiques au développement.

On peut lister les avantages associés au prototype :

- Facilité de debug par l'interface de visualisation dans l'éditeur.
- Possibilité de s'abstraire de la création de « singleton » manuel pour gérer les effets globaux.
- Il n'est pas nécessaire de lier les acteurs entre eux avec des interfaces et ou des « cast » de classe.
- Indépendance des systèmes permettant l'ajout ou le retrait de fonctionnalité sans modifier le reste du projet.
- Placement automatique s'attachant aux acteurs. Si des acteurs sont déplacés les sons les suivent.
- Même si le placement automatique dans l'éditeur n'est pas nécessaire au projet, les sélections rapides par tags et leurs éditions communes permettent d'augmenter l'efficacité.
- Valeurs d'« alpha » permettant de séparer les valeurs du gameplay de celles du feedback. On peut comparer ce comportement au Widget d'Unreal Engine qui ne s'occupe que de logique de feedback.
- **Fonctionnalités du mixage** permettant à l'acteur de réaliser des comportements plus complexes et adaptatifs en fonction des situations de gameplay.

8. Recommandations

Avec plus de temps, une mise en situation plus approfondie de l'outil aurait pu être réalisée. Les défis qui ont empêché cette évaluation de l'outil sont les contraintes budgétaires et de communications des studios. Tester un outil externe est un pari risqué pour un studio de jeu vidéo. Communiquer des retours à des externes est aussi une limitation courante de l'industrie et nécessite alors la signature de NDAs.

Ces raisons ont été identifiées avec Paschal Adans grâce à son expérience dans son entreprise « Carte Son ».

Pour bénéficier de retours externes, les possibilités seraient de le publier sur le magasin d'Epic Games « Fab » ou de porter l'outil vers des moteurs de jeu open source tels que Godot Engine ou Flax Engine. Pour collecter des données, l'outil peut proposer de remplir un formulaire utilisant la méthode SUS (Brooke, 1995) permettant d'évaluer la satisfaction de l'outil.

La création d'outils pour la sonorisation d'environnement est un sujet d'actualisé.

Des acteurs sont actifs dans la création de ce genre d'outil sont :

- SWEETECH, qui développe des outils d'instanciation de sources sonore, mais aussi d'outil de visualisation et d'édition dans le moteur Unreal Engine 5. (Sweetech, s. d.)
- Rhys Anthony, cette personne a développé des outils permettant de peindre des points de sonorisation lié au plugin Soundscape. Rhys explique le fonctionnement de ses outils et la facilitation du placement manuel de source audio.
- Epic Game, les systèmes Metasound et Soundscape évolue constamment avec le moteur. D'autres système existe permettant d'analyser l'audio en jeu et de synchroniser les évènements audios.
- Chris Zukowski est un directeur technique à Terrible Posture Games. Il a développé un jeu nommé Mix Universe se rapprochant de ce qu'est une station audionumérique comme il l'explique dans sa présentation à la Game Sound Con 2022. Il est possible de placer des nœuds qui se connectent et génère de la musique. Son concept intègre entièrement l'environnement et le gameplay pour sonoriser la scène. (Chris Zukowski, 2022)

Ce [plugin](#) est téléchargeable via Github.

9. Bibliographie

Sources de logiciels et documentations

- Audiokinetic. (s. d.). *Wwise Unreal Integration*. Consulté le 7 mai 2025, à l'adresse https://www.audiokinetic.com/fr/public-library/2024.1.4_8780/?source=UE4&id=index.html
- Epic Games. (s. d.). *Working With Audio In Unreal Engine | Unreal Engine 5.1 Documentation*. Epic Developer Community. Consulté le 28 mai 2024, à l'adresse <https://dev.epicgames.com/documentation/en-us/unreal-engine/working-with-audio-in-unreal-engine>
- Epic Games. (2023). *Electric Dreams Environment* [Logiciel]. <https://www.unrealengine.com/en-US/electric-dreams-environment>
- Fernandes Pereira, T., & Martins, M. (2024). *Parkware* [Logiciel].
- Martins, M., Storino, G., & Zylka, Y. (2025). *Friends Don't Ghost* [Logiciel].
- Inc, O. E. (2019, octobre 29). *The Sound of The Outer Worlds: Part 1*. Audiokinetic. <https://www.audiokinetic.com/en/blog/the-sound-of-the-outer-worlds-part-1/>
- SweeJTech. (s. d.). *SweeJTech*. Consulté le 2 juin 2025, à l'adresse <https://sweej.tech>

Livres et articles académiques

- Brooke, J. (1995). *SUS: A quick and dirty usability scale*. *Usability Eval. Ind.*, 189.
- Ciesla, R. (2022). *Sound and Music for Games: The Basics of Digital Audio for Video Games* (1re éd.). Apress.
- Stevens, R., & Raybould, D. (2016). *Game Audio Implementation: A Practical Guide Using the Unreal Engine*. Routledge. <https://www.routledge.com/Game-Audio-Implementation-A-Practical-Guide-Using-the-Unreal-Engine/Stevens-Raybould/p/book/9781138777248>
- Verfaillie, V., Zölzer, U., & Arfib, D. (2006). *Adaptive digital audio effects (A-DAFx): A new class of sound transformations*. *IEEE Transactions on Speech and Audio Processing*, 14(5). <https://doi.org/10.1109/TSA.2005.858531>
- Flamme, K. (2021). *Approche méthodologique de l'enquête auto-ethnographique dans l'étude des organisations. ¿ Interrogations ? Revue pluridisciplinaire de sciences humaines et sociales*, 33. <https://hal.science/hal-03485797>
- Soule, B. (2007). *Observation participante ou participation observante? Usages et justifications de la notion de participation observante en sciences sociales. Recherches qualitatives*, 27. <https://doi.org/10.7202/1085359ar>

Conférences et présentations

- Bas Bertrand (Réalisateur). (2024, février 14). *Project Overview—Procedural Ambience System with Unreal Engine 5 and Wwise* [Enregistrement vidéo]. YouTube. <https://www.youtube.com/watch?v=9TlXtKVHqJl>

- Game Camp (Réalisateur). (2023, août 9). *Outcast 2: Les défis techniques de la création sonore d'un open world* | Anthony Deneyer | GCF 2023 [Enregistrement vidéo]. YouTube. <https://www.youtube.com/watch?v=3x4Rh3DfKqE>
- MacGregor, A. (2014, mars). *The Sound of Grand Theft Auto V*. <https://gdcvault.com/play/1020587/The-Sound-of-Grand-Theft>

Sources audiovisuelles

- Zukowski, C. (Réalisateur). (2022, décembre 7). *Bringing Music to the 'Mix Universe'* | GameSoundCon 2022 | Unreal Engine [Enregistrement vidéo]. YouTube. <https://www.youtube.com/watch?v=gQMOMpM2Y-o>
- Rhys Anthony (Réalisateur). (2024, février 28). *UE5.2 Scriptable Tools for Data Entry & Editor-Time Infographics* [Enregistrement vidéo]. YouTube. <https://www.youtube.com/watch?v=HTiidLtsFrQ>
- Unreal Engine (Réalisateur). (2023, juin 29). *Behind the Sounds of Electric Dreams* | Inside Unreal [Enregistrement vidéo]. YouTube. <https://www.youtube.com/watch?v=0ZcOpaZqMqo>
- Weaver, D. (Réalisateur). (2023, octobre 16). *Raycast and Line Trace Your Audio In Unreal!* [Enregistrement vidéo]. YouTube. <https://www.youtube.com/watch?v=AkXOMrluiwo>.

10. Résumé

Ce mémoire examine les différentes méthodes nécessaires à la sonorisation d'un jeu vidéo en utilisant Unreal Engine 5 dans le but de les faciliter via le développement d'outils et de fonctionnalités. Ces outils et fonctionnalités éditeur sont ensuite mises en situation pour déterminer leurs efficacités pour des développeurs de jeux vidéo.

11. Mots-clés

Environnement sonore, Automatisation, Outils, Audio adaptatif, Unreal Engine 5