audiokinetic

Wwiseの基本的なアプローチ

2017.2.4 編集

Wwiseの基本的なアプローチ

Wwise 2017.2.4 Revision 3176
製作著作 © 2018 Audiokinetic Inc. All rights reserved.

This document (whether in written, graphic or video form) is supplied as a guide for the Wwise® product. This documentation is the property of Audiokinetic Inc. (“Audiokinetic”), and protected by Canadian copyright law and in other jurisdictions by virtue of international copyright treaties.

This documentation may be duplicated, reproduced, stored or transmitted, exclusively for your internal, non-commercial purposes, but you may not alter the content of any portion of the documentation. Any copy of the documentation shall retain all copyright and other proprietary notices contained therein.

The content of the Wwise Fundamentals documentation is furnished for information purposes only, and its content is subject to change without notice. Reasonable care has been taken in preparing the information contained in this document, however, we disclaim all representations, warranties and conditions, whether express, implied or arising out of usage of trade or course of dealing, concerning the Wwise Fundamentals documentation and assume no responsibility or liability for any losses or damages of any kind arising out of the use of this guide or of any error or inaccuracy it may contain, even if we have been advised of the possibility of such loss or damage.

Wwise®, Audiokinetic®, Actor-Mixer®, SoundFrame® and SoundSeed® are registered trademarks, and Master-Mixer™, SoundCaster™ and Randomizer™ are trademarks, of Audiokinetic. Other trademarks, trade names or company names referenced herein may be the property of their respective owners.

目次

表目次

例目次

第1章 はじめに

はじめに

Audiokinetic社は、サウンドデザイナーやオーディオプログラマーのニーズをふまえ、オーディオデザインの革新的なソリューションであるWwiseを開発しました。誕生から数年が経過したWwiseは、以下を目的として開発されました。

- 完全なオーサリングソリューションの提供
- オーディオやモーションの新しい制作ワークフローの導入
- パイプラインの効率化
- オーディオやモーションを使った、これまでの限界を超えた没入感の達成

この強力で包括的なオーディオ専用パイプラインソリューションは、以下を提供しています。

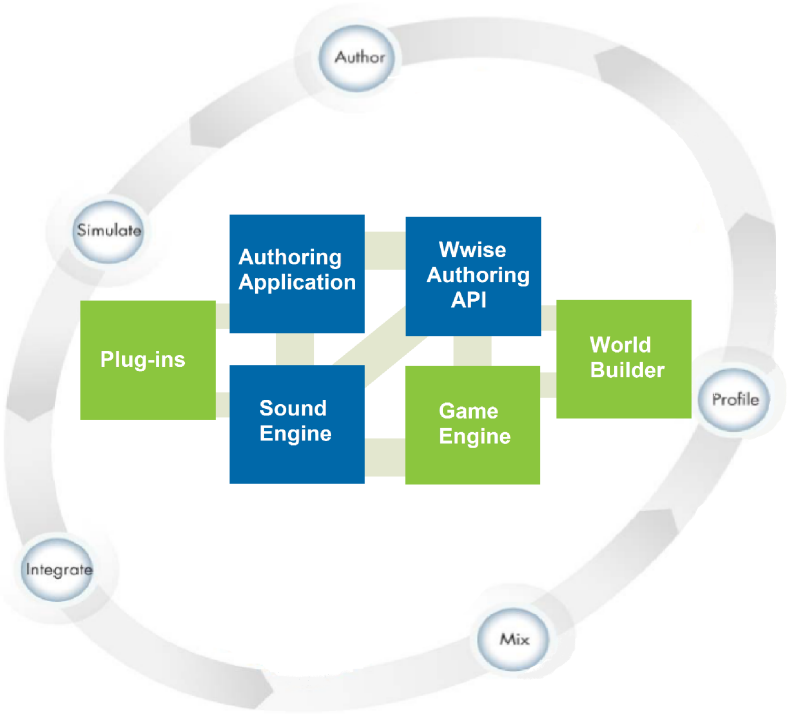
- **強力なオーサリングアプリケーション** — オーディオやモーションのアセット構成の作成、プロパゲーション設定、サウンド、ミュージック、モーションの統合管理、再生時のプロファイリング、サウンドバンクの作成などを可能とする、ノンリニア形式のオーサリングツール。
- **革新的なサウンドエンジン** — オーディオやモーションの処理を管理し、多様な機能を総合的に実行する、各種プラットフォームに最適化された洗練されたサウンドエンジン。
- **ゲームシミュレータ** — ゲーム中のサウンドやモーションの動作を正確に再現するLUAスクリプトインタプリタを搭載しているので、Wwiseをゲームのサウンドエンジンに統合する前に、各プラットフォームにおけるWwiseの個別の動作の確認や、各プラットフォーム上でのパフォーマンスのプロファイリングが可能。
- **プラグインに対応したアーキテクチャ** — ゲーム中のオーディオによる没入感を簡単に拡大させる、完全に拡張可能なプラグインアーキテクチャ。以下は、提供中のプラグインの一例。
  - 音源など、オーディオやモーションを生成するソースプラグイン。
  - リバーブなど、オーディオエフェクトを作成するためのエフェクト用プラグイン。
- **WwiseとワードビルダーのインターフェースWwise Authoring API-外部ゲームワールドビルダ—や3Dアプリケーションとの独自プラグインインターフェースで、外部アプリケーションがWwiseとシームレスにやり取りできます。Wwise Authoring APIを利用して、Sound Engine APIで一般的に変更できることが、全て簡単に変更できます。

Wwiseの制作パイプラインについて

Wwiseは基礎となる制作パイプラインに革新的な考え方を導入し、実際のゲームの中で、リアルタイムに様々なタスクを実行するために必要なツールを、緊密に統合させ提供しています。

- **オーサリング** — サウンド、モーション、ミュージックの構成（structures）をビルドし、プロパティや動作を定義。
- **シミュレーション** — 芸術的な方向性を再確認し、ゲームプレイをシミュレーション。

- **インテグレーション** — 追加のプログラミングなしで、早い段階で統合を実行。
- **ミキシング** — ゲーム上で、リアルタイムにプロパティをミキシング。
- **プロファイリング** — ゲームの要件に確実に準拠するために、リアルタイムでプロファイルを設定。



Wwiseのプロジェクトの考え方

Wwiseはプロジェクト構造を採用したシステムで、ゲームのオーディオやモーション情報は、全てのプラットフォームのものを、1つのプロジェクトに入れて扱います。

プロジェクト内で、以下の機能が利用できます。

- ゲームのサウンド、音声、ミュージック、モーションなどの各アセットの管理。
- オブジェクトのプロパティや再生時の動作の定義。
- ゲームのオーディオやモーションをトリガーさせるアクションイベントやダイアログイベントの作成。
- プロトタイプとシミュレーションの作成。
- プロジェクト内のオーディオやモーションの、全面的なトラブルシューティングやプロファイル設定。

プラットフォームや言語のバージョン毎に生成されたサウンドバンクも、プロジェクトに含まれます。

プロジェクト内のアセット管理

典型的なゲームには、サウンド、ミュージック、モーションのアセットが何千件もあり、Wwiseプロジェクトで効率的かつ効果的に管理する必要があり、異なるプラットフォームや言語に対応させたバージョンがいくつもある場合は、効率的な管理がさらに重要です。

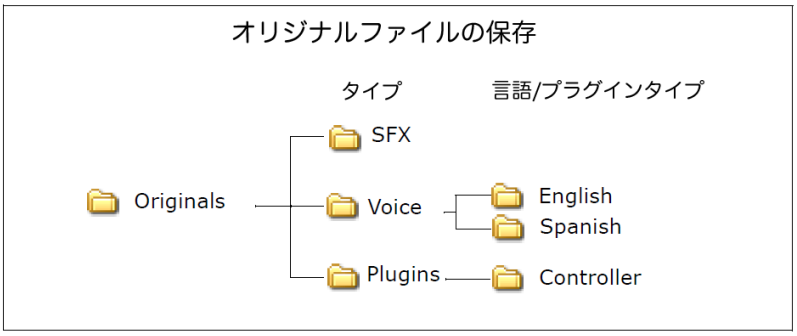
オリジナルのファイル

Wwiseは「非破壊的なシステム」のため、アセットをプロジェクト内で編集しても、元のファイルに影響しません。Wwiseにファイルをインポートすると、プロジェクトの[Originals]フォルダ にそのコピーが保存されます。

インポートされたファイルは、タイプ別に、以下のフォルダに保存されます。

- 以下は、提供中のプラグインの一例。
- SFX
- ボイス

ファイルに、Voiceファイルやプラグインファイルのフラグが立っている場合、さらにランゲージやプラグインタイプによって分けられます。下図は、プロジェクトにインポートされたオリジナルのアセットがWwise内でどのように整理されるかを示します。



プラットフォーム別のバージョン

Wwiseの [Originals]フォルダから、各プラットフォーム向けバージョンを作成します。各プラットフォーム用のバージョンは、プロジェクト内のキャッシュフォルダに保存されます。

Wwiseは、キャッシュフォルダのコンテンツを効率よく管理するために、変換後のアセットを以下の条件で分類します。

- **Platform** (Windows®、Xbox One™、PlayStation®4など)
- **Type** (プラグイン、SFX、またはVoice)。アセットに、Voiceファイルやプラグインファイルのフラグが立っている場合、さらに以下に分けられます:
  - **言語** （英語、フランス語、スペイン語など）
  - **Plug-in Type** (コントローラ)

第2章 ゲームにオーディオを統合するには

Wwiseの基本的なアプローチ

コードを見てWwise SDKを使い始める前に、オーディオのビルドやゲームへの統合に関するWwise独特のアプローチをご理解ください。またWwiseを効率よく利用し、最大限に活用するために必要なコンセプトもご紹介します。

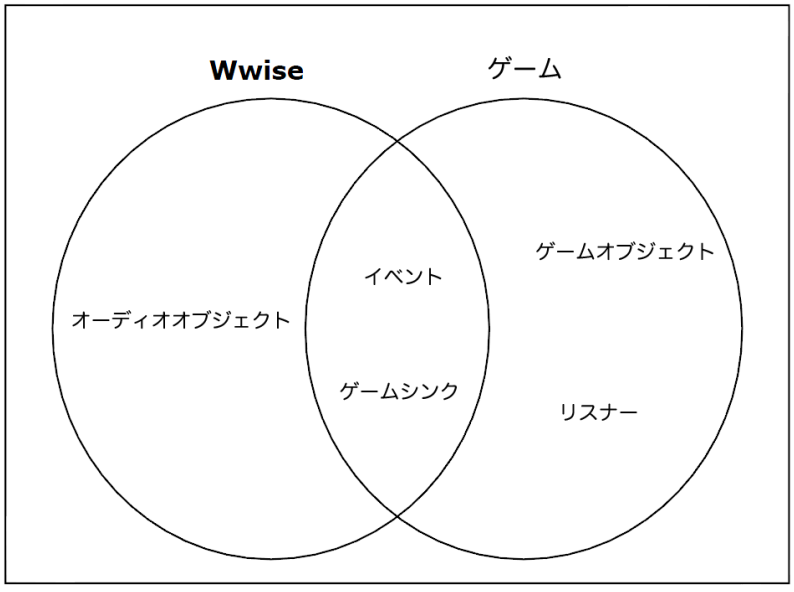
Wwiseのオーディオ構築とゲームへの統合の考え方には、下記の５つの主要なコンポーネントがあります。

- オーディオオブジェクト（Audio Objects）
- イベント（Events）
- ゲームシンク（Game Syncs）
- ゲームオブジェクト（Game Objects）
- リスナー（Listeners）

各コンポーネントの詳細は後述しますが、最初に役割や相互関係を理解する必要があります。

Wwiseの目標の一つは、プログラマーとデザイナーの作業を明確に分けることです。例えば、ゲームに出てくる個別サウンドとなるオーディオオブジェクトは、サウンドデザイナーがWwiseで独自に作成・管理します。一方、オーディオを送受信するゲーム内の要素であるゲームオブジェクトやリスナーは、プログラマーがゲーム内で作成・管理します。残るイベントとゲームシンクは、ゲームのオーディオをドライブさせるコンポーネントです。この２つのコンポーネントがオーディオアセットとゲーム側のコンポーネントをつなぐブリッジであり、Wwiseとゲームの双方にとって不可欠です。

下図は、この５つのコンポーネントが、どこで作成・管理されているかを示しています。



第3章 プロジェクト階層

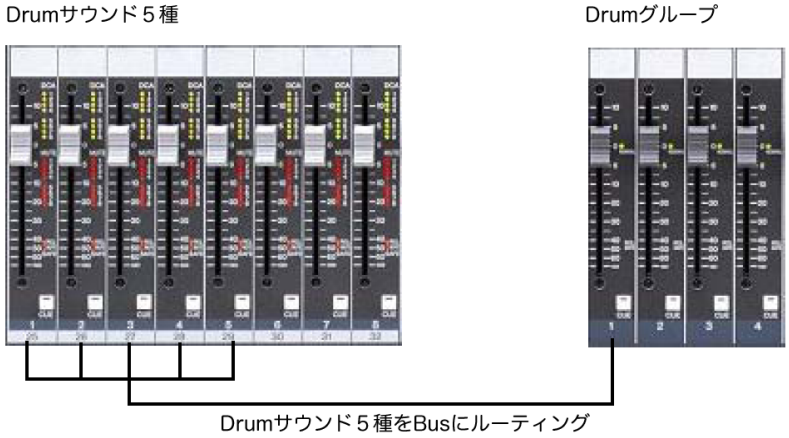
プロジェクト階層

プロジェクト階層の基盤となるのが、プロジェクトにインポートしたアセットです。プロジェクト階層は従来のミキシング技術から発展したもので、複数の楽器をバスにルーティングし、一つのミキシングしたサウンドとしてサウンドプロパティをコントロールするのと同じことです。例えばハイハット、ライド、クラッシュ、バスドラム、スネアなどのサウンドを一つのバスにルーティングすると、一つの要素として音量やその他のパラメータをコントロールできます。

Wwiseは同様のアプローチでサウンド、モーションオブジェクト、そしてミュージックをプロジェクト内で整理し、グループ化します。オブジェクト（サウンド、モーション、ミュージック）をグループ化することにより、階層型のプロジェクト構造が構築され始め、様々なオブジェクト間に親子関係が発生します。ゲームのオーディオやモーションを、このようにユニークで効率的な方法で作成し管理することで、コントロールと柔軟性が向上し、リアルで没入感のあるゲームを実現できます。

Wwiseのプロジェクト階層は、３つの異なるレベルで構成されています。

- **アクター・ミキサー階層（Actor-Mixer Hierarchy）** — Wwise独自のオブジェクトを使い、プロジェクトの全てのサウンドアセットやモーションアセットをグループ化し、整理する。
- **インタラクティブミュージック階層（Interactive Music Hierarchy）** — Wwise独自のオブジェクトを使い、プロジェクトの全てのミュージックアセットをグループ化し、整理する。
- **マスター・ミキサー階層（Master-Mixer Hierarchy）** — サウンド、モーション、ミュージックの構成のルーティングやアウトプットを、１つまたは複数のアウトプットバスを利用して定義する。



チーム作業を支えるWwiseのプロジェクト階層

今日のゲーム開発環境において、チーム作業は欠かせません。１つのゲームにつきWwiseプロジェクトは１つしか使用できませんが、プロジェクト階層をワークユニットに分割することで、複数のチームメンバーがプロジェクトで同時に作業することができます。ワークユニットは独立したXMLファイルで、プロジェクトの一部分または特定の要素に関連した情報が格納されています。ワークユニットはプロジェクトの様々な要素の整理や管理に活用することができます。また、チームで作業をする場合、ワークユニットをソースコントロールシステムで管理することにより、複数のメンバーによる並行作業が円滑に進められます。

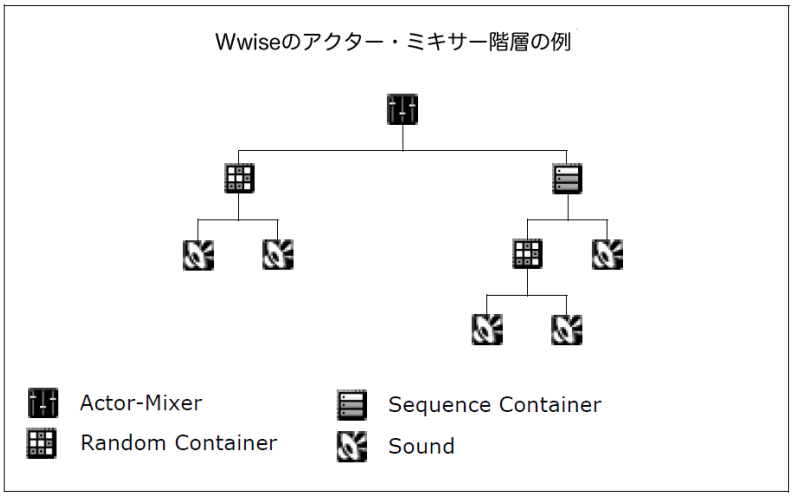
アクター・ミキサー階層の仕組み

プロジェクトの全てのサウンドアセットやモーションアセットが、アクター・ミキサー階層でグループ化され、整理されます。階層のベースには、全てのサウンドオブジェクトやモーションオブジェクトがあります。オブジェクトのプロパティや動作（behavior）は個々に定義できますが、いくつかのオブジェクトをグループ化し、１つのユニットとしてまとめてプロパティや動作を定義することも可能です。ゲームの複雑なオーディオ編成に対応するために、異なるタイプのオブジェクトをWwiseのプロジェクト階層に入れることができます。オブジェクトタイプによって、ボリューム、ピッチ、ポジショニングなどのプロパティセットがあり、ランダム再生やシーケンス再生など、固有の動作セットもあります。プロジェクト階層上でオブジェクトタイプを使ってサウンドをグループ分けすることにより、ゲーム中の同じグループのサウンドに特定の再生動作を設定することができます。また、階層の異なるレベルでプロパティや動作を定義し、違った効果を得ることもできます。

ゲームのモーションは、一般的にオーディオとひも付いているため、Wwiseではモーション制作にもオーディオと同じ考え方やワークフローを採用しています。つまり、ゲームのモーションアセットを階層に整理し、オーディオアセットと同様に、プロパティや動作を割り当てることができます。

以下のオブジェクトタイプを利用して、アセットをグループ分けし、プロジェクトの構造を作ることができます。

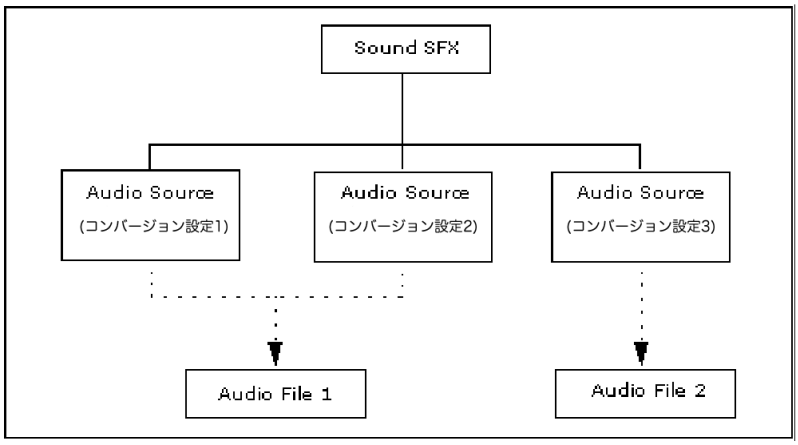
- サウンド（Sound）オブジェクト
- モーションエフェクト（Motion FX）オブジェクト
- コンテナ（Container）
- アクター・ミキサー（Actor-Mixers）



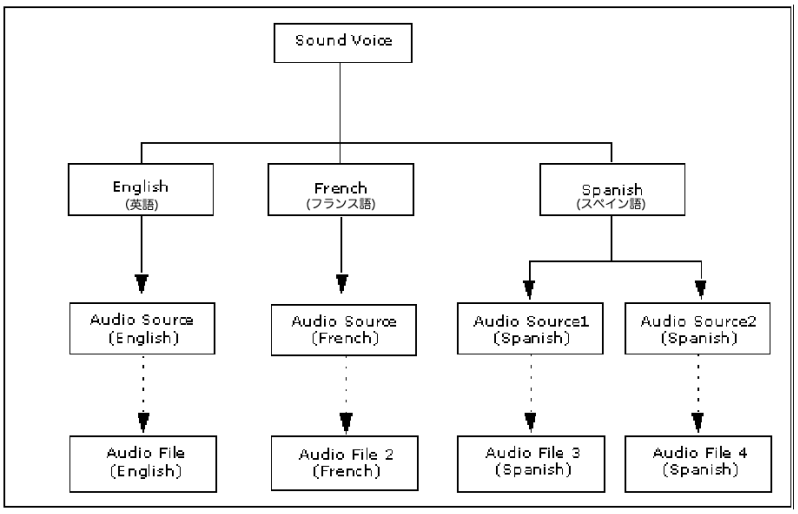
オーディオオブジェクト（Audio Objects）

ゲームのボイスアセットやSFXアセットは、Wwiseではサウンドオブジェクトという特殊なオーディオオブジェクトで表現されます。このサウンドオブジェクトに、元のオーディオファイルとリンクしたソースが格納されています。

オーディオソースは、インポートされたオーディオファイルとサウンドオブジェクトの間にある別のレイヤです。この抽象的なレイヤを追加することで、1つのサウンドオブジェクト内に、複数のソースやオーディオファイルを格納することができます。



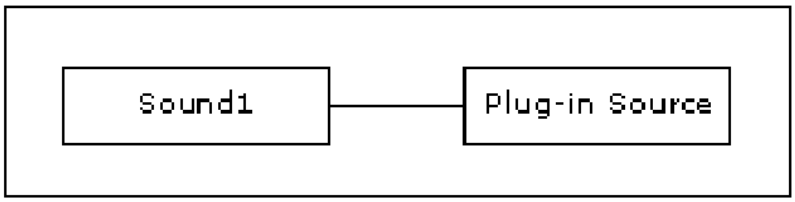
このため、様々な変換（conversion）設定を簡単に試すことができ、複数言語の対応も効率的に管理できます。



注：Wwiseではプロジェクトのミュージックアセットやモーションアセットも同様の方法で管理します。

ソースプラグイン

サウンドオブジェクトは、オーディオソース以外にプラグインソースにも対応しています。

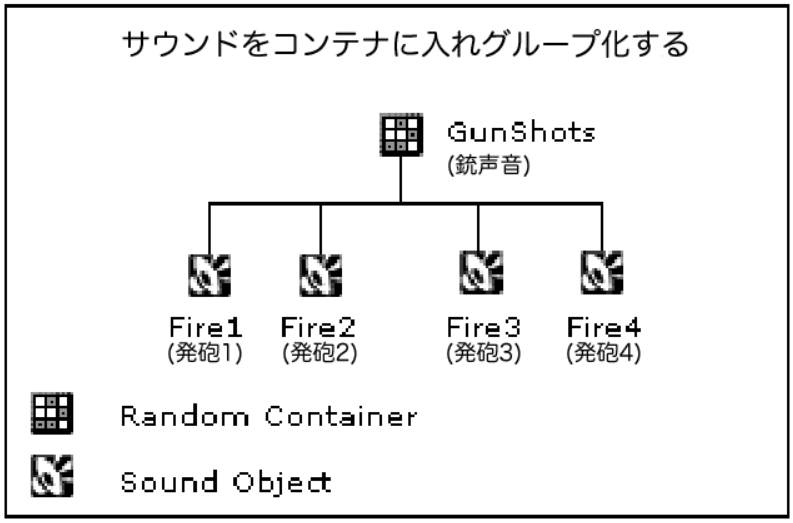


Wwiseには、音源、サイレンス、オーディオ入力など、様々なソースプラグインが同梱されています。これらのプラグインは開発パイプラインですぐに使用できるほか、プログラマーが独自のプラグインを開発する際に、提供されたソースコードと合わせてレファレンスとして利用できます。Wwiseではオーディオオブジェクトの作成と管理をサウンドデザイナーが行うため、プログラマーが様々なソースプラグインを開発する時間が増え、オーディオデザインの可能性が広がり、ゲーム全体の表現力も強化されます。

オーディオオブジェクトの階層を構築

サウンドオブジェクトをグループ化することで、プロジェクトの階層が構築されます。階層の様々なレベルでオーディオのプロパティや動作を設定できるため、コントロールと柔軟性が向上し、リアルで没入感のあるゲームを実現できます。

サウンドオブジェクトは、コンテナを使ってグループ化します。コンテナの主な用途は、ランダム、シーケンス、スイッチなど特定の動作を指定してグループのオブジェクトを再生させることです。例えば、全ての銃声音をランダムコンテナにまとめ、ゲームで発砲がある度に、異なる銃声音が再生されるように設定できます。



オーディオオブジェクトは全て、バス階層にルーティングされ、ここでも更にグローバルレベルで、プロパティやエフェクトを設定できます。

オーディオオブジェクトのタスクの担当者

下表は、オーディオオブジェクトに関連するタスクの担当者を、サウンドデザイナーとプログラマーに分けたものです。

表3.1 オーディオオブジェクトのタスクの担当者

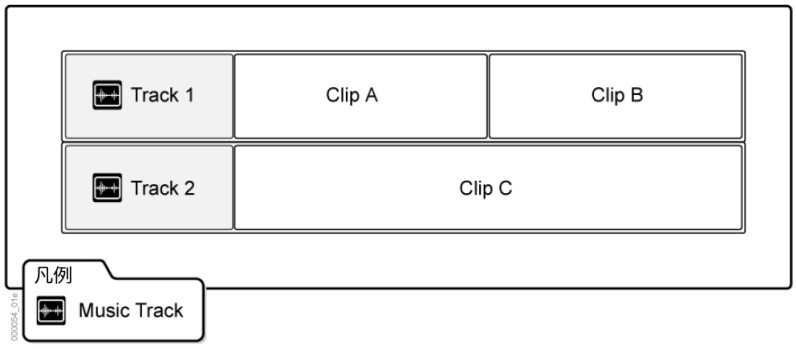
タスクの内容	サウンドデザイナー (Wwise)	プログラマー（ゲームコード、ツール）
サウンドオブジェクトの作成（オーディオアセット用）	X	
オブジェクトのグループ化、プロジェクト階層のビルド	X	
サウンドのプロパティや動作の定義	X	
オーディオオブジェクトをバスにルーティング	X	
ソースプラグインの開発		X

インタラクティブミュージック階層の仕組み

Wwiseでは、非常に柔軟にインタラクティブミュージックを制作することができます。インタラクティブミュージック用のオブジェクトを組み合わせて曲を作る方法は、ほぼ無数にあります。しかし、ある程度決まった構成に従った方がワークフローとして効率的です。

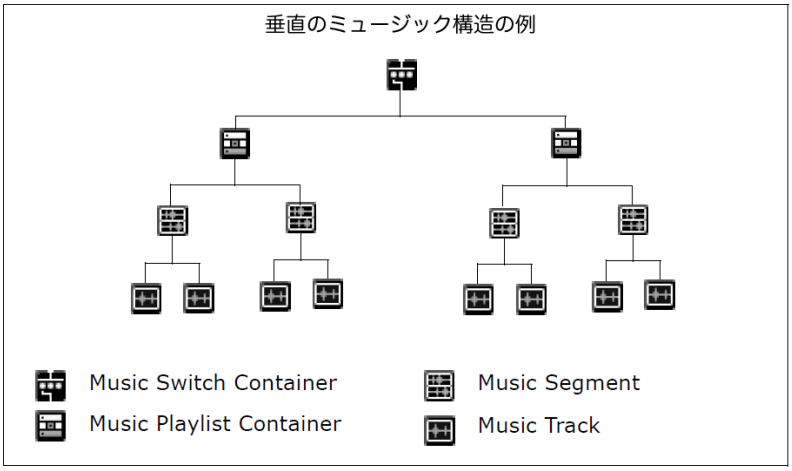
インタラクティブミュージックを制作する際に採用できる、２つの基本的なプロジェクト構造を下記に示します。

- **垂直のプロジェクト構造**とは、音楽セグメントに含まれるトラックをシャッフルして、ゲームの曲を変えていくプロジェクト構造です。音楽制作で行われるトラックのミキシングと似ています。長いマルチトラックのセグメントから、変化ある曲を作ることができます。



- **水平のプロジェクト構造** とは、再生する音楽セグメントを随時変えて、ゲームの曲を変化させるプロジェクト構造です。短い単独セグメントをインタラクティブミュージック階層（Interactive Music hierarchy）で アレンジする方法ですが、アクター・ミキサー階層でオブジェクトをアレンジする方法と似ています。これによって、ゲーム機の負荷を最小限に抑えながら、複数の短い音楽セグメントを使い魅力的な曲を作り出せます。

両方の構造を組み合わせながら、プロジェクトで与えられたリソースを効率的に使うのが一般的な手法です。優れた構造は際立つ音楽を生み出しながら、ゲーム機のリソースを上手に活用します。



マスター・ミキサー階層の仕組み

アクター・ミキサー階層やインタラクティブミュージック階層の上位にあるのが、マスター・ミキサー階層（Master-Mixer hierarchy）です。マスター・ミキサー 階層は複数のバスを有する別個の階層であり、プロジェクトの様々なサウンド構成、ミュージック構成、モーション構成を再度グループ化してミキシングを行い、出力用に準備します。マスター・ミキサー階層は２つに分けられ、サウンドとミュージックの部分と、モーションの部分があります。どちらも、最上位レベルにマスターバス（Master Bus）があり、その下のチャイルドバスの数は自由に設定できます。

サウンド、ミュージック、モーションの各構成をこれらのバスにルーティングする際、ゲーム内の主要カテゴリを適用することができます。例えば全てのオーディオ構造を、以下の４つのカテゴリにグループ化することができます。

- ボイス
- アンビエント
- サウンドエフェクト
- ミュージック

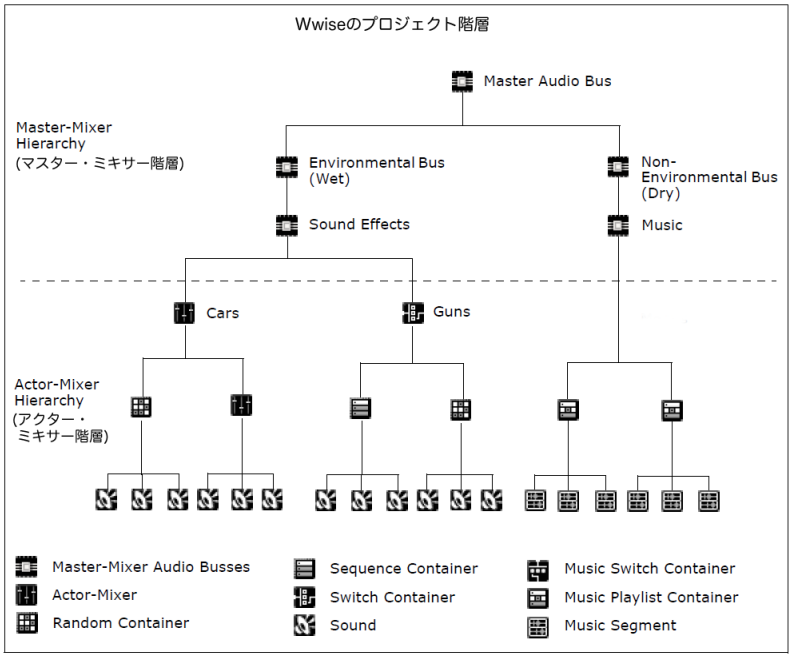
これらのバスは、プロジェクトのサウンド、ミュージック、モーションの各構成をコントロールする最終的なレベルとなるだけでなく、リバーブなど環境エフェクトの対象となるサウンドを決定するバスです。プロジェクト階層の最上位にあるので、ゲームの最終ミキシングに利用できます。またプラットフォームによっては、環境エフェクトなど特定のエフェクトをバスに適用し、ゲームに求められる没入感を創出できます。

また、オーディオバス構造を使ってゲーム内の問題のトラブルシューティングを行うこともできます。例えば、あるサウンドやミュージックを特定するために、ボイス、アンビエントサウンド、サウンドエフェクト・バスなどをソロにすることができます。

下図の例では、Master Audio Bus（マスターオーディオバス）階層において、まず２つのバスでEnvironmental（環境）サウンドとNon-Environmental（非環境）サウンドを分け、さらにアクター・ミキサー階層にあるサウンド構成や、インタラクティブミュージック階層にあるミュージック構成を、別のオーディオバスで改めてグループ化しています。

注記

Master Motion Bus（マスターモーションバス）においても、この例と同じレベルで同様の階層をつくり、プロジェクトの全てのモーション構成を編成することができます。

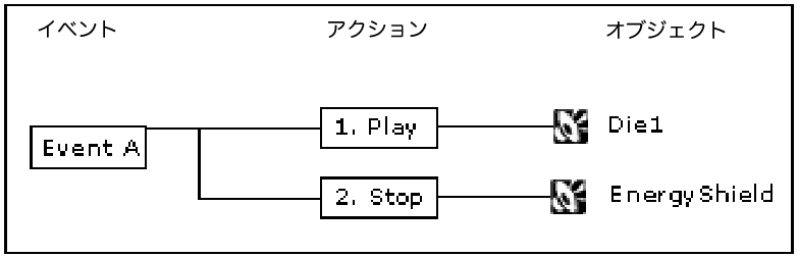


第4章 イベントの仕組み

イベントの仕組み

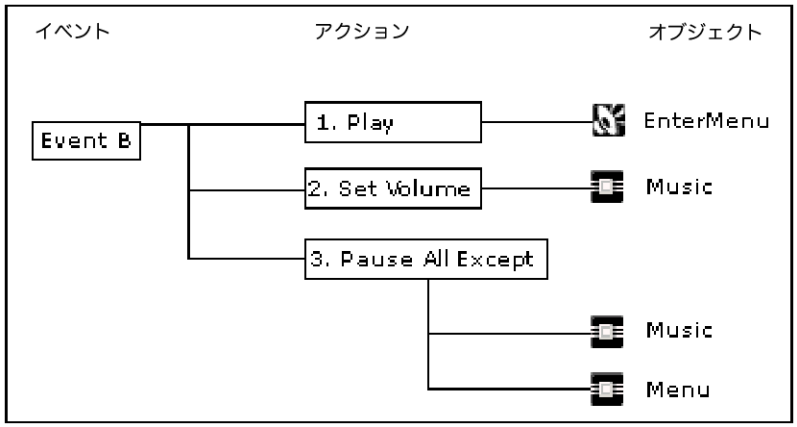
Wwiseでは、ゲーム内のオーディオをイベントでドライブします。プロジェクト階層のサウンドオブジェクトやオブジェクトグループに対して、イベントがアクションを適用させます。イベントに設定されたアクションによって、Wwiseオブジェクトの再生、停止、一時停止などが決まります。例えば、ファーストパーソン・シューティングゲームでプレイヤーが死ぬ時のイベントを作ってみましょう。このイベントは、 Die（死ぬ）サウンドを再生させ、再生中の EnergyShield（エネルギーシールド）のサウンドを停止させるものです。

下図は、このイベントに対応するWwiseの様子を示しています。



サウンドデザイナーはゲーム内のオーディオをドライブするMute（ミュート）、Set Volume（ボリューム設定）、Enable Effect Bypass（エフェクトのバイパス）などのアクションタイプを、多数の選択肢から選べます。例えば、プレイヤーがゲーム本編を出てメニューに入る時のイベントを作ってみましょう。このイベントは Enter_Menu サウンドを再生させ、ミュージックバスの音量を10dB下げ、それ以外を全て一時停止させるものです。

下図は、このイベントに対応するWwiseの様子を示しています。



できるだけ多くの状況に対応するために、Wwiseでは下記の２種類のイベントを使います。

- **アクションイベント** — 再生、停止、一時停止などのアクションを１つ以上用いて、ゲームのサウンド、ミュージック、モーションをドライブするイベント。
- **ダイアログイベント** # StateやSwitchの入った一種のデシジョンツリーを使って、どのオブジェクトを再生するかをダイナミックに判断するイベント。

Wwiseで作成したイベントはゲームエンジンに統合して、ゲーム中に適宜呼び出します。イベントは開発の初期段階でゲームエンジンに統合できます。その後、イベントの微調整を続けても、ゲームエンジンに改めて統合する必要はありません。

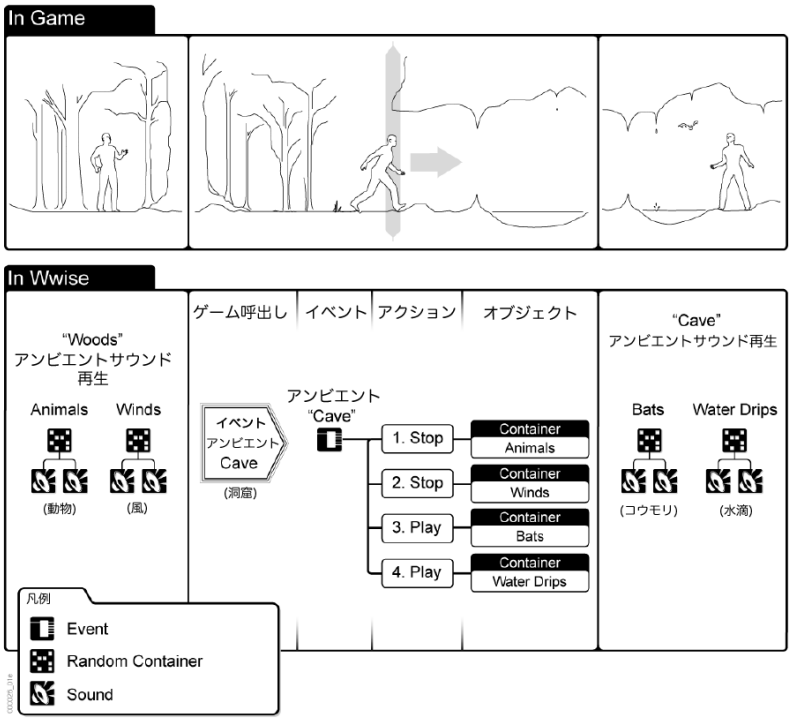
アクションイベント

ゲーム中のサウンド、ミュージック、モーションをドライブするのは、Wwiseのアクションイベントです。アクションイベントが、プロジェクト階層に入っているサウンドなどの構成に対してアクションを適用します。1つのイベントにアクションを1つ、または一連のアクションを複数、入れることができます。選択したアクションによって、Wwiseのオブジェクトの再生、一時停止、停止などが決まります。

例4.1 アクションイベントの活用例

例えば、ゲーム中のキャラクターが、隠されたドキュメントを手に入れるために洞窟に入る場面を考えてみましょう。キャラクターが森から洞窟に入る時、ゲーム中のアンビエントサウンドを変化させなくてはなりません。この変化をトリガーさせるために、Woods（森）のアンビエントサウンドを停止し、Cave（洞窟）のアンビエントサウンドを再生する一連のアクションを含むイベントを作成します。Wwiseで作成したイベントをゲームエンジンに統合すると、キャラクターが洞窟に入った瞬間に、ゲームエンジンがこのイベントを呼び出します。

下図は、再生中のアンビエント音を変えるために、ゲームエンジンがイベントをトリガーさせる様子を示しています。



サウンド、ミュージック、モーションの各オブジェクト間の切り替わりに対処するために、イベントアクションで開始するオブジェクトや終了するオブジェクトのディレイやフェードイン・フェードアウトを設定するパラメータもあります。

ダイアログイベント

ゲーム中のダイナミックダイアログ（随時変化する台詞）をドライブするのが、Wwiseのダイアログイベントで、再生する台詞を判断するためのルールや条件セットが設定されています。ダイアログイベントを使い、ゲームの様々なシナリオ、条件、結末などを計画できます。あらゆる状況に確実に対応するために、デフォルトや予備条件も設定できます。

これらの条件は全て、一連のState値やSwitch値を使って定義します。これらのStateやSwitchの値を組み合わせて、ゲーム中の具体的な条件や結果を定義するパスが作られます。次に各パスに対して、Wwiseサウンドオブジェクトをひも付けます。ゲーム中にダイアログイベントが呼び出されると、ゲームは現在の条件とダイアログイベントで定義された条件を比較します。ゲームの現在の状況と合致する条件やStateパスまたはSwitchパスで、どのダイアログを再生するかが決まります。

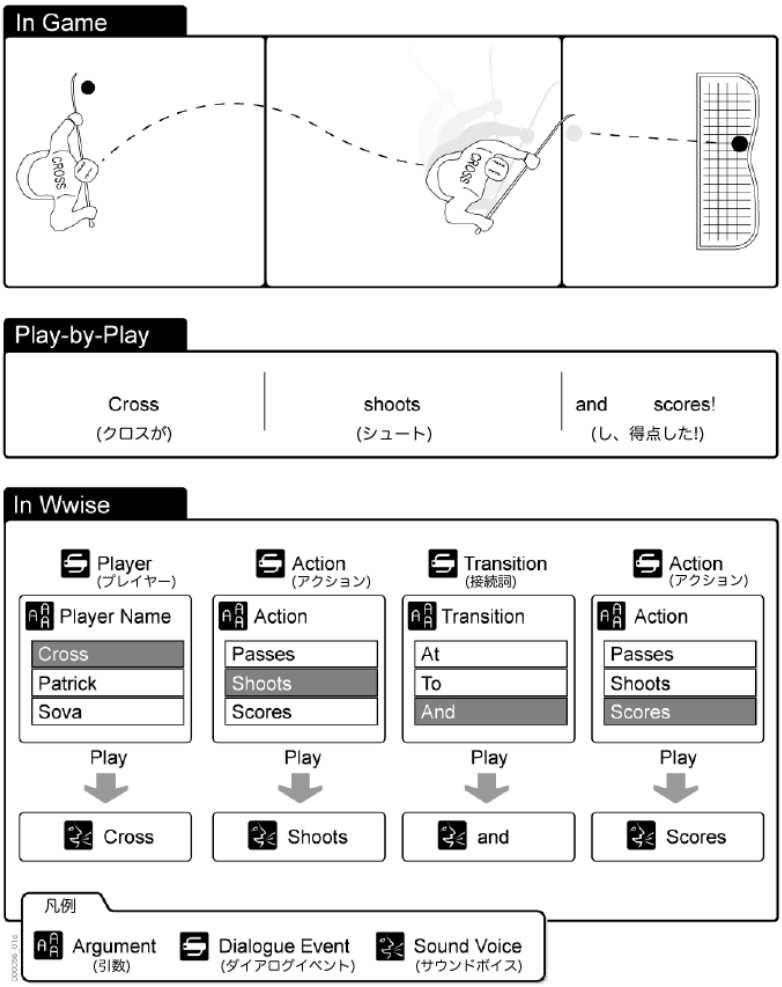
注記

ダイアログイベントはゲーム中の台詞や会話のために開発されましたが、台詞専用にする必要はなく、ゲーム内の他の様々な目的に使用できます。

例4.2 ダイアログイベントの活用例

例えば、実況中継付きのアイスホッケーゲームを考えてみましょう。プレイヤーがシュートして得点した時は、プレイヤーのアクションに対応した実況中継が流れるべきです。Wwiseで様々な可能性や結末を設定するために、まず Player（プレイヤー）、Action（アクション）、Transition（接続詞）などのダイアログイベントを作成します。これらイベントには、ゲーム用に作成したState値やSwitch値のセットが入っています。ゲームの条件や結果が導かれるようにStateやSwitchのパスを作成してから、そのStateパスやSwitchパスに、適切なVoiceオブジェクトをアサインする必要があります。ゲームプレイ中に、現在のState値またはSwitch値とWwiseで定義したパスを、ゲームが比較して、どのVoiceオブジェクトを再生するのかを判断します。

下図は、「クロスがシュートし、得点した！」という実況中継を、Wwiseのダイアログイベントが生成する様子を示しています。



イベントのスコープの定義

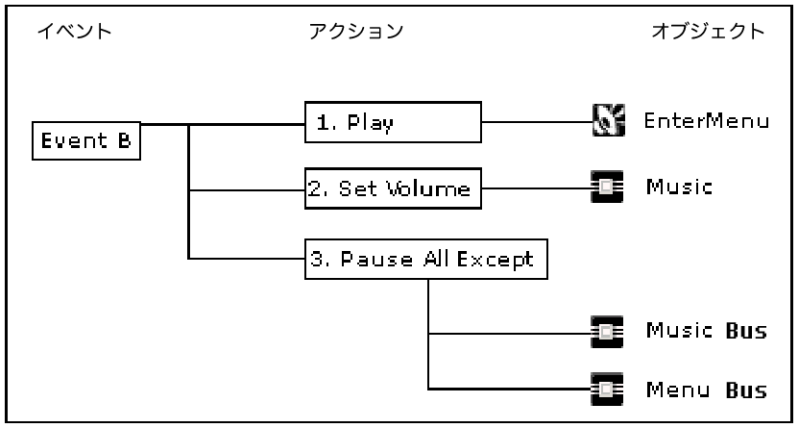
イベント内の全てのアクションに、スコープ（対象）が設定されています。スコープの設定によって、そのイベントのアクションを全ゲームオブジェクトにグローバルに適用するのか、またはイベントをトリガーさせたゲームオブジェクト限定で適用するのかを決定します。アクションによって、サウンドデザイナーがスコープを選択できたり、スコープが既に決まっていたりします。

下表は、前述のEvent Bの３つのイベントアクションについて、それぞれのスコープを示しています。

表4.1 イベントのスコープの定義

イベントアクション	スコープ（対象）	備考
Play > Menu_Enter	ゲームオブジェクト	Playイベントは常に単独のゲームオブジェクトによってトリガーされるため、スコープを「ゲームオブジェクト」に設定。
Set Volume > Music	グローバル	Set Volumeアクションはバスに適用され、バスは本質的にグローバルであるため、スコープを「グローバル」に設定。
Pause All Except > Music	グローバル	Pause All Exceptアクションはミュージックバスに適用され、ミュージックバスは本質的にグローバルであるため、スコープは自動的に「グローバル」に設定。

下図は、このイベントに対応するWwiseの様子を示しています。



注記

スコープは、Wwiseの多くの要素に適用される重要な概念です。各要素のスコープを理解すれば、様々な状況でどの要素を使うべきかが判断しやすくなります。

イベントをゲームに統合するには

ゲームのイベントを作成した後、サウンドデザイナーはイベントをサウンドバンク（SoundBank）にパッケージします。サウンドバンクをゲームにロードすれば、ゲームのコードでイベントをトリガーできます。例えば、プレイヤーが殺された時に対応するイベントをトリガーすることによって、 Die サウンドが再生され、EnergySheild サウンドが停止されます。

イベントをゲームに統合するには、イベントアクションをどのゲームオブジェクトに対して実行させるのかを、プログラマーが指定する必要があります。これはイベントを送出（post）して行います。オーディオを変えたい時は、ゲームコードによってイベントが送出される必要があります。イベントの送出には、文字列やIDを使います。

Wwiseイベントの利点

ゲーム内のサウンドをトリガーする時にイベントを採用する主な利点は、サウンドデザイナーのコントロールと柔軟性が、追加のプログラミングをせずに広がることです。イベントは全てサウンドデザイナーがWwiseで作成し、プログラマーがゲームに統合します。ゲームに統合されたイベントに対して、サウンドデザイナーが作業を続けることが可能で、中に含まれるアクションや対象のオブジェクトを変更し、修正できます。ゲームは相変わらず同じイベントをトリガーするので、サウンドデザイナーによる変更は、開発者による追加作業やコードの再コンパイル抜きでゲーム内に反映されます。

イベントのタスクの担当者

下表は、イベントに関連するタスクの担当者を、サウンドデザイナーとプログラマーに分けたものです。

表4.2 イベントのタスクの担当者

タスクの内容	サウンドデザイナー (Wwise)	プログラマー（ゲームコード、ツール）
イベントの作成	X	
オーディオオブジェクトのイベントアクションの割り当て	X	
イベントアクションのスコープの定義	X	
ゲーム内でイベントの送出		X

第5章 ゲームオブジェクトとは？

ゲームオブジェクトとは？

サウンドエンジンでトリガーされるイベントは必ずゲームオブジェクトに割り当てられるので、ゲームオブジェクトはWwiseの中心的な存在です。一般的にゲームオブジェクトはサウンドを出す物体やエレメントを表し、キャラクター、武器、環境内のトーチなどが含まれます。ただし 1 つのエレメントの複数の部分に、それぞれ別のゲームオブジェクトを設定する場合もあります。例えば、巨大なキャラクターの複数の部分にゲームオブジェクトを設定し、足音のサウンドとキャラクターの音声が3Dサウンド空間の異なる位置から発せられるようにできます。

注記

ゲームエンジンUnrealを使用したことがあれば、Unrealの「Actors」とWwiseのゲームオブジェクトが似ていることに気づくでしょう。

ゲーム内で各サウンドをどのように再生させるかは、Wwiseに保存されたゲームオブジェクトに関する様々な情報によって決まります。ゲームオブジェクトには、下記の情報などが割り当てられます。

- ボリュームやピッチなど、ゲームオブジェクトにひも付いているオーディオオブジェクトのプロパティオフセット値。
- 3Dの位置（position）と方向性（orientation）。
- ステート、スイッチ、RTPCなど、ゲームシンク情報。
- 環境エフェクト。
- オブストラクションや、オクルージョン。

注記

他のプロパティと異なり、減衰（attenuation）は、ゲームオブジェクトではなくオーディオオブジェクトに適用されます。このため、サウンドデザイナーが個々のサウンドの減衰を、より柔軟にコントロールできます。Wwiseの3D Game Object ビューを使うと、サウンドを割り当てたゲームオブジェクトを見て、リスナーとゲームオブジェクトの位置関係や各サウンドの減衰半径を確認できます。

ゲームオブジェクトの登録

ゲームオブジェクトを使う前に、プログラマーがゲームオブジェクトをゲームコードに登録する必要があります。不要になったゲームオブジェクトは、登録の解除がない限りサウンドエンジンがゲームオブジェクトの関連情報（3Dポジション、RTPC、スイッチなど）を保存し続けてしまうので、登録を解除して下さい。

スコープの設定 ー ゲームオブジェクト単位か、グローバルレベルか

ゲームオブジェクトを使うことで、イベントの説明でも触れたスコープ（対象）という概念がWwiseに導入されます。スコープの設定で、ゲームのサウンドにプロパティやイベントを適用させる範囲を決定します。適用の対象をゲームオブジェクトとするのか、グローバルレベルとするのかを、スコープを使って選べます。スコープはゲーム内で実際に起きる状況やアクションによって決まり、Wwise上のアプローチの仕方も最終的に決まります。

例えば、ファーストパーソン・シューティングゲームの制作を考えてみましょう。メインキャラクターが、敵の旗を獲得するために街中を歩くとします。キャラクターが街を歩き回る時に、彼の足音がします。この足音に関連するプロパティやサウンドを変更する場合は、メインキャラクターの足に関係のあるゲームオブジェクトに限った範囲で、つまりローカルレベルで変更することになります。一方、キャラクターが水中に潜った場合は、周囲の環境で再生が続く爆発音や車両音などのサウンドを全て、変える必要があります。この場合、変更をグローバルスケールで適用します。

ゲームオブジェクトの利点

ゲームオブジェクトを使用することで、プログラマーは個別のサウンドではなく、ゲームオブジェクトを管理すれば良いので、オーディオ全体の管理が簡素化されます。

ゲームオブジェクトを作成した後、プログラマーの作業はイベントの送出、ゲームシンク用のスイッチ、ステート、RTPCなどの設定、そしてゲーム内の環境の設定だけとなります。具体的にどのサウンドを、どのように再生させるかといった詳細は、サウンドデザイナーがWwiseで設定できます。このアプローチで、ゲーム中の多数のエンティティに割り当てられた膨大な量のサウンドを扱う作業時間を、大幅に短縮することができます。

ゲームオブジェクトのタスクの担当者

下表は、ゲームオブジェクトに関連するタスクの担当者を、サウンドデザイナーとプログラマーに分けたものです

表5.1 ゲームオブジェクトのタスクの担当者

タスクの内容	サウンドデザイナー (Wwise)	プログラマー（ゲームコード、ツール）
ゲームオブジェクトを、ゲームの3Dオブジェクトに割り当て	X	X
ゲームオブジェクトの登録と解除		X
ゲームオブジェクトの位置情報の更新		X
オーディオオブジェクトの減衰設定	X	
イベントのスコープ設定	X	

第6章 ゲームシンクとは？

ゲームシンクとは？

ゲームの初期デザインが完了したら、Wwiseのゲームシンク（Game Syncs）を使いゲームで起きる変化や変更を整理し、扱い方を検討します。以下の５種類のゲームシンクから、ゲームのビジュアルを強化する最適の手段を選んで下さい。

- **ステート（States）** — ゲーム内の現行のサウンド、ミュージック、またはモーションのプロパティに、グローバルスケールで影響するゲーム中の変化。
- **スイッチ（Switches）** — あるゲームエレメントに関して存在する複数の選択肢の提示であり、全く新しいサウンド、ミュージック、またはモーションが要求されることもある。
- **RTPC** — ゲームパラメータの変数値にマッピングされたプロパティであり、ゲームパラメータ値を変えるとプロパティ自体が変更される。
- **トリガー（Triggers）** — ゲーム中に発生した事象へのレスポンスのことで、再生中のミュージックの上から重ねてミキシングされる短いミュージックフレーズであるスティンガーを起動させる。

ゲームプロジェクトを進めていく上で、常に品質、メモリ制限、そして開発スケジュールをバランスさせる必要があります。ゲームシンクを戦略的に活用することで、作業を簡素化しメモリ使用量を削減できるため、真に没入感のあるゲームの制作に役立ちます。

ステートの仕組み

ステートとは基本的に「ミキサースナップショット」であり、オーディオやモーションのプロパティに対するグローバルなオフセットや調整を設定して、ゲーム内の物理的条件や環境条件の変化を表現します。ステートの活用でオーディオやモーションのデザイン行程が効率化され、アセットの最適化に役立ちます。

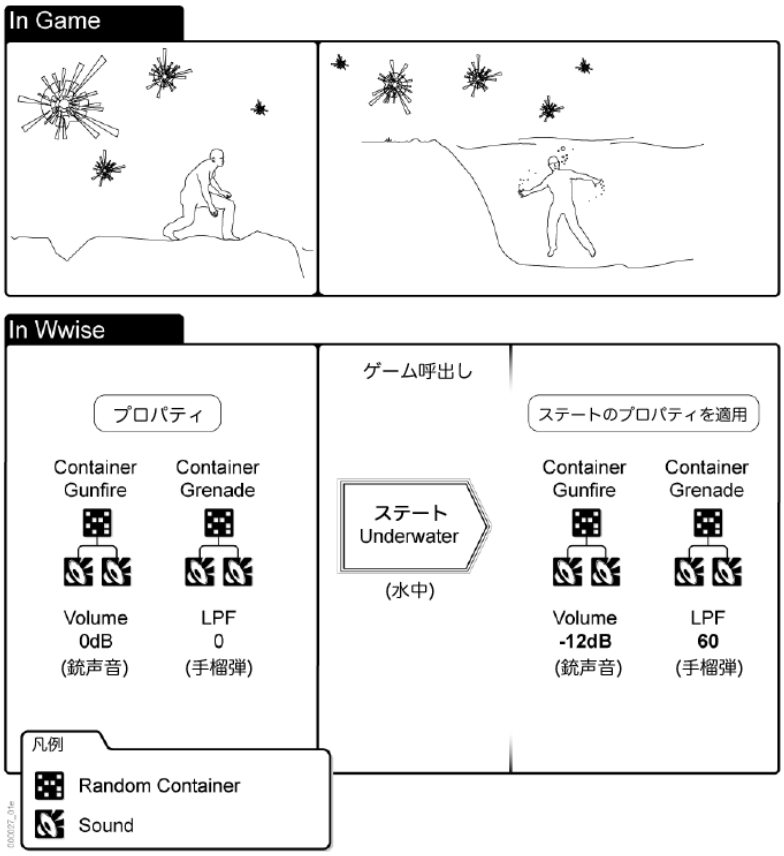
「ミキサースナップショット」であるステートを使うことで、アウトプットされるサウンドを細かくコントロールでき、複数のステートを組み合わせて目的を達成することもできます。また、１つのオブジェクトを複数のステートに登録すると、１つのプロパティが複数の値の変更によって影響されます。この場合、値の変化の合計が適用されます。例えば、別々のステートグループに所属する２つのステートのボリュームチェンジがそれぞれ「-6 db」であり、両者が同時に有効となった場合、最終的なボリュームは「-12 db」になります。

このようにして「ミキサースナップショット」を作成し定義すると、オブジェクト（サウンド、ミュージック、またはモーション）の新たなプロパティセットを設定しても、メモリやディスクの使用量は増加しません。ここで定義したプロパティセットが、あるステート（単一または複数）の時のサウンド再生のルールとなります。また、多数のオブジェクトにグローバルレベルでプロパティチェンジを適用することでリアルなサウンドスケープを簡単に作れ、オーディオとゲーム全体の表現力が強化されます。既に再生中のサウンド、ミュージック、モーションのプロパティを変更することで、アセットを再利用でき貴重なメモリを節約できます。

例6.1 ステートの活用例

例えば、キャラクターが水中に潜る時のサウンド処理を考えてみましょう。この場合、ステートを利用して再生中のサウンドのボリュームやLPF（ローパスフィルター）を変更できます。キャラクターが水中にいる時に聞こえる銃声音や手榴弾の爆発音を表現するサウンドのシフトを、プロパティ変更によって作り出します。

下図は、ゲーム側が Underwater（水中）ステートを呼び出した時に、Gunfire（銃声音）と Grenade（手榴弾）の２つのサウンドオブジェクトのボリュームとLPFのプロパティが変更される様子を示しています。



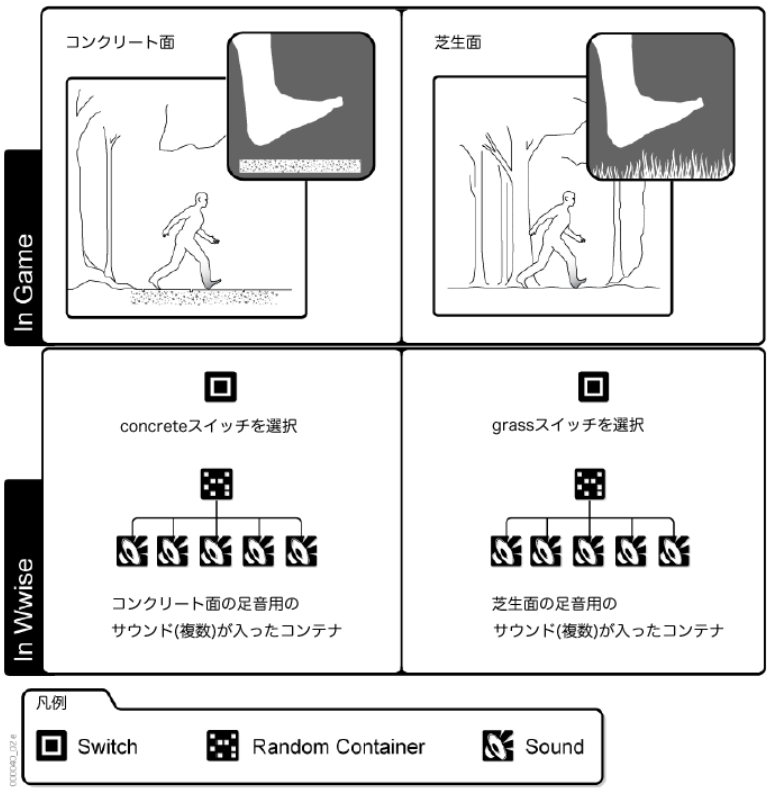
スイッチの仕組み

Wwiseのスイッチは、ゲーム内のゲームオブジェクトについて、様々な選択肢を提示するものです。スイッチにオブジェクト（サウンド、ミュージック、モーション）を整理して割り当てることによって、ゲーム中の状況がある選択肢から別の選択肢へ移った場合に、適切なサウンドまたはモーションオブジェクトが再生されます。スイッチに割り当てられるWwiseのオブジェクトは、まとめてスイッチコンテナに格納します。イベントが変化を通知すると、スイッチコンテナがスイッチを検証し、正しいサウンド、ミュージック、またはモーションオブジェクトを再生させます。

例6.2 スイッチの活用例

例えば、ファーストパーソン・シューティングゲームで、メインキャラクターが次々と変わる環境を歩いたり走ったりする状況を考えてみましょう。環境に応じて地面も変化し、コンクリート、芝生、土などの地面素材によって足音も変えていきます。この場合、それぞれの地面素材に対してスイッチを作成し、適切な足音サウンドを割り当てます。メインキャラクターがコンクリート面を歩くとconcrete（コンクリート）スイッチが有効になり、該当するサウンドが再生されます。続いてコンクリート面から芝生面に移動した場合は、grass（芝生）スイッチが有効になり、該当するサウンドが再生されます。

下図は、有効になったスイッチによって、再生する足音サウンドが決まる様子を示しています。



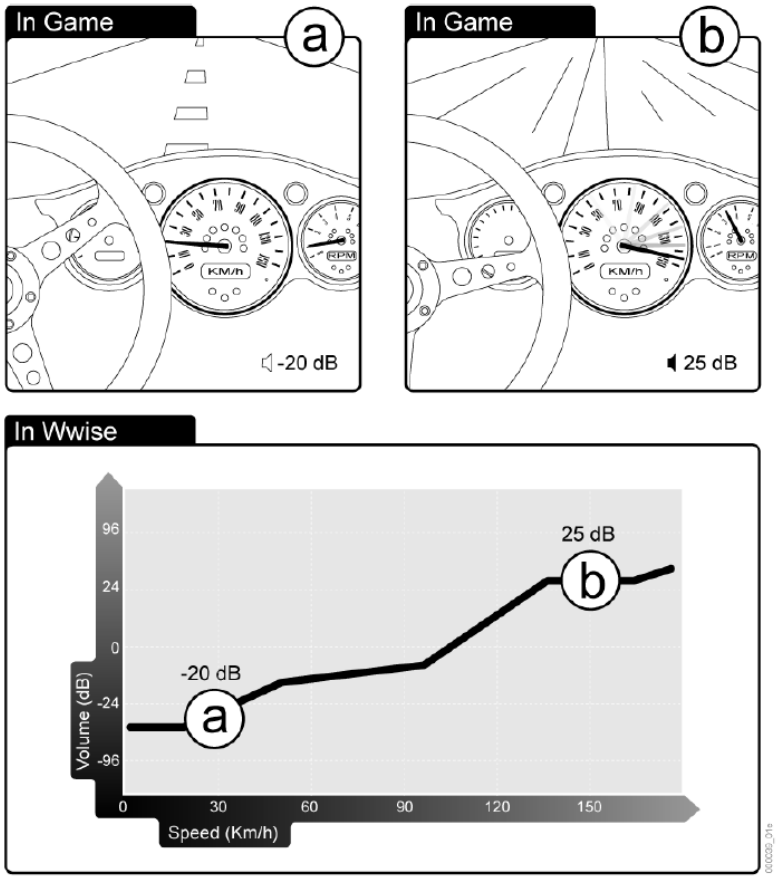
RTPCの仕組み

RTPC （リアルタイム・パラメータコントロール、Real-time Parameter Control）を使うことによって、ゲーム内でリアルタイムに起きる様々なパラメータ値の変化に合わせて、オブジェクトのプロパティをリアルタイムで編集できます。RTPCでゲームパラメータをプロパティ値にマッピングし、プロパティ変更を「自動化」することにより、ゲームがさらにリアルになります。パラメータ値はグラフ化され、Y軸にWwise内のスイッチグループ値またはプロパティ値が、X軸にゲーム内のパラメータ値が表示されます。プロパティ値をゲームパラメータ値にマッピングすることで、両者の全体的な関係を定義するRTPCカーブが作成されます。リッチで没入感あふれるゲーム体験を提供するために、必要なだけの数のカーブが作れます。

例6.3 RTPCの活用例

例えば、レーシングゲームの制作を考えてみましょう。エンジンサウンドのボリュームやピッチは、車のスピードやRPMに合わせて変動させる必要があります。この場合RTPCを使い、車のエンジンサウンドのピッチとボリュームのレベルを、ゲーム内のスピードやRPMにマッピングします。車が加速すると、マッピングに従ってピッチやボリュームのプロパティ値が反応します。

下図は、Wwiseのマッピングに基づき、ゲーム内のレーシングカーのスピードに合わせてボリュームが変化する様子を示しています。



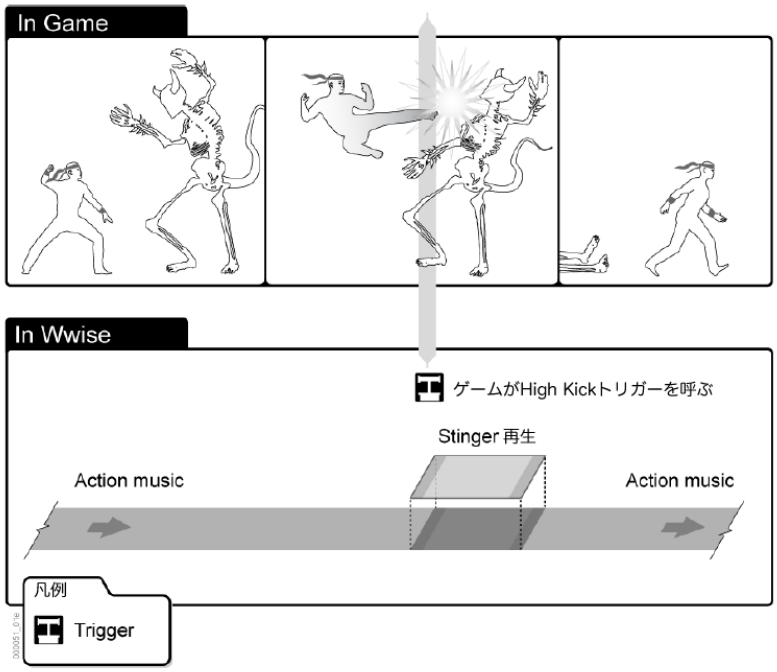
トリガーの仕組み

他のゲームシンクと同じく、トリガーもゲームに呼び出され、ゲーム内の状況に応じてレスポンスを決定するWwiseの要素です。具体的には、インタラクティブミュージックにおいて、ゲームで発生した事象にトリガーが反応して、スティンガー（stinger）を起動させます。スティンガーは再生中の音楽に重ねてミキシングされる短いミュージックフレーズのことで、ゲームに対する音楽的な反応となります。例えば、忍者が武器を取り出した時、シーンのインパクトを更に強めるために、既に再生中のアクションミュージックに重ねてスフォルツァンド風の音楽エフェクトを挿入できます。この場合、ゲームがトリガーを呼び出し、そのトリガーでスティンガーが起動し、流れているスコアに重ねてスティンガーのミュージッククリップが再生されます。

例6.4 トリガーの活用例

例えば、忍者ファイターがメインキャラクターの格闘ゲームを考えてみましょう。ゲーム中のいくつかのポイントで、キャラクターが敵と戦うアクションモードに入ります。ファイターが強力なキックを決めた時は、そのシーンの聴覚的な効果を強めるミュージッククリップを流すとします。このような場面の音楽を構築するには、ゲーム中の特定ポイントで呼び出されるトリガーが必要で、今回の例では、High Kick（ハイキック）というトリガーを設定します。同時に、興奮を高める瞬間的なブラス音楽を鳴らす、短いミュージックセグメントを定義します。

下図は、ゲームの重要ポイントでスティンガーを再生させるトリガーのメカニズムを示しています。



ゲームシンクのタスクの担当者

下表は、ゲームシンクに関連するタスクの担当者を、サウンドデザイナーとプログラマーに分けたものです。

表6.1 ゲームシンクのタスクの担当者

タスクの内容	サウンドデザイナー (Wwise)	プログラマー（ゲームコード、ツール）
スイッチグループ作成、スイッチ作成	X	
ステートグループ作成、ステート作成	X	
ステート移行時間の定義	X	
スイッチとステートグループに、スイッチコンテナを割り当て	X	
トリガーの設定	X	
ゲームエンジンからWwiseオーディオエンジンへ、ステート情報とスイッチ情報を送出		X

第7章 シミュレーションの作成

シミュレーションの作成

ゲーム開発では全てを完璧にするまで、試行錯誤が続くのが常です。このような作業のために、WwiseはSoundcasterという強力なシミュレーション環境を提供します。開発のどの段階にあっても、Wwiseオブジェクトやイベントを使いオーディオやモーションのシミュレーションをビルドできます。

Soundcasterを利用できるタスクの一例を、以下に示します。

- プロトタイプの制作や実験
- プルーフオブコンセプトの開発
- サウンドオブジェクトとミュージックオブジェクトの同時試行
- ゲーム内のオーディオやモーションのプロファイリング
- オーディオとモーションの、ミキシングやテスト

Wwiseのイベントやオブジェクト（サウンド、モーション、ミュージック）をシミュレーションするだけでなく、ゲームに接続し、ゲーム自体がトリガーするサウンド、モーション、ミュージックを用いたシミュレーションも可能です。作成されたシミュレーションをSoundcasterのセッションとして保存すれば、開発中にいつでも過去のシミュレーションに戻れます。

第8章 プロファイリングとトラブルシューティング

プロファイリングとトラブルシューティング

ゲーム開発者にとって最も高いハードルの１つは、各プラットフォームの制限や制約に配慮しながらも、リッチで没入感のあるゲームを開発することです。ゲームのオーディオやモーションを様々なプラットフォームに対応させる手段が、Wwiseには多数あります。さらにWwiseのGame Profiler（ゲームプロファイラ）やGame Object Profiler（ゲームオブジェクトプロファイラ）を使い、各プラットフォームにおけるオーディオやモーションのパフォーマンスをテストできます。この２つのツールで開発プロセスのどの段階においても、全てのプラットフォームで、ゲームのオーディオやモーションの具体的な箇所をプロファイリングできます。さらにリモートゲームコンソールに接続し、サウンドエンジンからプロファイル情報を直接キャプチャすることもできます。サウンドエンジンの動きを監視することで、メモリ、音声、ストリーミング、エフェクト、サウンドバンクなどに関連する具体的な問題を検知し、トラブルシューティングできます。ゲーム中にプロファイルしたり、Game SimulatorやSoundcasterを使ったり、Wwise Authoring APIを使ってゲームに実装する前からプロトタイプをプロファイリングしたりできます。

必要な情報が見やすいように、Game Profilerのレイアウトは以下の３つのビューに分かれています。

- **Capture Log** — サウンドエンジンから来る全ての情報をキャプチャし、記録する。
- **Performance Monitor** — サウンドエンジンが実行する全てのアクティビティのパフォーマンスについて、CPU、メモリ、帯域幅などを図式化したもの。サウンドエンジンからキャプチャされた情報をリアルタイムで表示する。
- **Advanced Profiler** — パフォーマンスモニターやトラブルシューティングに使用できる、サウンドエンジンの総合的な指数の一覧。

Game Object Profilerのレイアウトでは、以下のビューが表示されます。

- **Game Object Explorer** — Wwiseゲームオブジェクトのプロファイリングツールを扱うコントロールセンターであり、リアルタイムで見たいゲームオブジェクトやリスナーをここで選択する。
- **Game Object 3D Viewer** — ゲームオブジェクトやリスナーを3Dで表示。
- **Game Sync Monitor** — RTPC値をリアルタイムで分析するツール。ゲームプレイ中、表示中のゲームオブジェクトのRTPC値の変化をグラフ表示する。

これらのビューは緊密に統合されているため、問題エリアを発見し、原因となるイベント、アクション、またはオブジェクトを検出し、様々な要素をサウンドエンジンがどう扱っているかを判断して問題を迅速かつ効率的に解決できます。

第9章 サウンドバンクの仕組み

サウンドバンクの仕組み

ゲームのオーディオコンポーネントやモーションコンポーネントを効率的に管理するために、Wwiseは全てのオーディオデータとモーションデータをバンクに入れます。バンクは基本的に、データ（オーディオデータやモーションデータ）とメディアのどちらか、または両方が格納されたファイルです。バンクはゲーム中の特定ポイントでプラットフォームのメモリにロードされます。必要なものだけをロードするため、オーディオやモーションによるメモリ使用量を各プラットフォームで最適化できます。バンクの内容はそれまでの作業の生産物であり、ゲームの一部となる最終的なオーディオコンテンツやモーションコンテンツが入っています。

Wwiseには、以下の２種類のバンクがあります。

- **初期化バンク（Initialization Bank）** — プロジェクトの一般的な情報を全て入れた特別なバンクで、バス階層、ステート、スイッチ、RTPC、環境エフェクトなどの情報が含まれます。Wwiseがサウンドバンクを生成すると、初期化バンクが自動的に作成されます。通常、初期化バンクはゲームの最初に１回ロードされ、ゲームプレイ中にプロジェクトの全般情報に簡単にアクセスできるようにします。デフォルトでは初期化バンクの名前は「Init.bnk」です。
- **サウンドバンク（SoundBank）** — イベントデータ、サウンド構成データ、モーション構成データ、メディアファイルなどのセットが入ったファイルです。一般的にサウンドバンクは初期化バンクと異なり、ゲームの最中に適宜ロードやアンロードされプラットフォームのメモリを効率的に使います。またイベントやプロジェクト構造のメタデータをメディアとは別のサウンドバンクに追加することもでき、メディアファイルのロードを必要最小限に抑えられます。プラットフォームは全て異なるので、Wwiseで簡単にサウンドバンクを各種プラットフォームに対応させることができ、全プラットフォーム用のサウンドバンクを同時に生成できます。さらにWwiseは、各プラットフォームの制限に準拠していることを確認するために、サウンドバンク用のトラブルシューティングのツールも提供しています。

効率的に作業するために、Wwiseにはサウンドバンク用のレイアウトがあります。プロジェクトのサウンドバンクを作成、管理、生成するのに必要な全てのビューがこのレイアウトに入り、SoundBank Manager、SoundBank Editor、Project Explorer、Event Viewerなどのビューが表示されます。

ファイルパッケージャ

Wwiseプロジェクトで生成されたサウンドバンクやストリームしたメディアファイルは、スタンドアロンユーティリティのファイルパッケージャ（File Packager）を使い、１つまたは複数のパッケージにまとめることができます。ファイルパッケージはファイルシステムを抽象化した内蔵ユニットで、プラットフォーム側のファイルシステムにみられる、ファイル名の長さやファイル数などに関する制限を、ある程度回避できます。また複数ある言語バージョンや、リリース後に出されるダウンロード用コンテンツなどを扱いやすくします。

第10章 Wwiseのサウンドエンジン

Wwiseのサウンドエンジン

Wwiseを使って、素晴らしいサウンド、ミュージック、モーションの構成をビルドし、サウンドバンクにパッケージ化できますが、デザインしたサウンドやモーションを再生するには、強力で安定したサウンドエンジンが必要です。Wwiseのサウンドエンジンはベーシックなレベルにおいて、ゲーム中のオーディオやモーションのあらゆる面をリアルタイムで管理し処理します。ゲームの完成に向けて、ゲーム開発パイプラインと簡単に統合できるように設計されています。

通常は、このような統合タスクや処理タスクには膨大な量のプログラミングが発生しますが、Wwiseのサウンドエンジンは処理パイプラインをダイナミックに作り上げ、開発者がどのようなゲームやプラットフォームの独自のニーズに対しても、自由にエンジンをカスタマイズできるようにしています。

このサウンドエンジンの高度な能力と、Wwiseのオーサリングアプリケーションとの緊密な連携のおかげで、以下の機能も実現します。

- サウンドデザイナーがオーサリングアプリケーションで作成し定義した、ランダム再生やシーケンス再生など、一般的な再生動作の実行。
- 既製品を基に、サウンドデザイナーがオーサリングアプリケーションで作成し調整したフェードやクロスフェードの実行。
- サウンドデザイナーがオーサリングアプリケーションで設定したプレイバック制限、プライオリティ設定、バーチャルボイスなどに基づき、サウンドオブジェクトやモーションオブジェクトの優先順位を管理。
- シンプルなAPIで、既製の環境製品を無制限にサポート。さらに環境のルーティングをサウンドエンジンが動的に作成・破棄するので、全てのプラットフォームで一貫したゲーム体験が確保され、メモリフットプリントやCPU稼働率を削減。
- ゲーム内でソースが部分的または完全にエレメントでブロックされた場合、自然に発生するオブストラクションやオクルージョンをサポート。
- ゲーム内で最大８つのリスナーをサポート。
- オーサリングアプリケーション内で、デバグのインストルメントコードを、リアルタイムで表示。これにより、サウンドデザイナーはゲームのラン中にリアルタイムでプロファイル出力を分析し、適切に対処できる。

Wwiseのサウンドエンジンの詳細については、Wwise SDKドキュメントを参照ください。

第11章 リスナー（Listeners）

リスナー（Listeners）

リスナーは、ゲーム内のマイクを表します。リスナーはゲームの3D空間で位置と方向性を有します。ゲームプレイ中に、リスナーの座標とゲームオブジェクトの位置を比較し、ゲームオブジェクトに付随する3Dサウンドを適切なスピーカーに割り当てることによって、実際の3D環境を模倣します。プログラマーが確実にリスナーの位置情報を更新させなければ、サウンドは誤ったスピーカーでレンダリングされてしまいます。

複数リスナーの設定

プレイヤーが１人のゲームで視野が１つしか表示されない場合は、リスナーは１つで十分です。一方、同一システムで複数のプレイヤーが参加する場合、または複数の視点を同時に表示させる場合は、それぞれの視点のオーディオが正しくレンダリングされるよう、ビュー１件に対してリスナー１個が必要です。Wwiseのサウンドエンジンは、最大８個のリスナーをサポートします。

デフォルトとして、登録されたゲームオブジェクトは第１リスナーだけに割り当てられます。しかしプログラマーが動的に、好きなリスナーにゲームオブジェクトを割り当てたり、解除することができます。

複数のリスナーに対してオーディオを設定した時、１人１人のプレイヤーが目にする場面とサウンドソースの位置が必ずしも一致しないため、課題が多くあります。主な原因は、複数のプレイヤーの3D環境を１セットのスピーカーで再現していることにあります。Wwiseではこのような制限に対処する様々なツールやテクニックを提供し、プレイヤー全員が可能な限りリアルなオーディオを体験できるようにします。Wwiseが複数のリスナーをどう扱うかについては、SDKの Integrating Listeners をご参照ください。

リスナーのタスクの担当者

下表は、リスナーに関連するタスクの担当者を、サウンドデザイナーとプログラマーに分けたものです。

表11.1 リスナーのタスクの担当者

タスクの内容	サウンドデザイナー (Wwise)	プログラマー（ゲームコード、ツール）
リスナーの位置情報を設定		X
リスナーにゲームオブジェクトを割り当て		X
複数リスナーを管理		X

第12章 デザイナーとプログラマーのタスク分担

デザイナーとプログラマーのタスク分担

すでにお気付きでしょうが、Wwiseのサウンド制作や統合へのアプローチは他のサウンドエンジンと異なります。サウンドデザイナーのコントロール範囲を広げ、これまで主に開発者が担当した、時間のかかる繰り返しの作業を最小限に減らしたことで、サウンドデザイナーも開発者も、よりクリエイティブでおもしろい作業に集中できるようになります。依存関係がほぼ排除され、どちらも得意とする分野に専念し、より効率的な共同作業ができます。Wwiseはサウンドデザイナーやオーディオプログラマーの異なる役割と、それぞれのタスクを認識した製品です。

サウンドデザイナーの責任

サウンドデザイナーは、下記のタスクを担当します。

- オーディオ階層や動作の作成。
- オーディオイベントの作成。
- ゲームのワールドビルディングアプリケーションへの、オーディオイベントの統合。
- サウンドプロパティやソースの設定。
- サウンドのポジショニング（3D）の定義。
- 全サウンドのオーディオシグナルルーティングとミキシングの設定。
- RTPCやゲームステートの割り当て。
- ゲーム内の様々な環境における、エフェクトのプロパティセットの設定。
- ボリュームや、LPFのオブストラクション・オクルージョンプロパティの設定。
- サウンドバンクの管理と最適化。
- 複数プラットフォームに対応するためのカスタマイズ。
- 他言語へのローカリゼーション。

オーディオプログラマーの責任

オーディオプログラマーは、下記のタスクを担当します。

- ゲームエンジンに、Wwiseサウンドエンジンを統合。
- コードを使い、オーディオイベントを統合。
- ゲームのサウンドを出すゲームオブジェクトの登録。
- 1つ以上のオーディオアクションを含むイベントをトリガーさせる、AK::SoundEngine::PostEvent()メソッドの呼び出し。
- サウンドデザイナーやゲームの指示に従い、サウンドバンクの定義（definition）ファイルやファイルパッケージャユーティリティを使用サウンドして、サウンドバンクのロードとアンロードを実行。
- リスナーの座標システムに対する3Dゲームオブジェクトのポジション設定。
- ステートやスイッチの変更のトリガー実行と、RTPCの更新。
- ゲームのサウンドに適用される環境エフェクト率（%）の設定。
- リスナーに対するゲームオブジェクトのオブストラクション値やオクルージョン値の算出。

- サウンドバンクのロード・アンロード、ストリーミング処理、プラグイン登録など、メモリーリソースの管理。
- ソースプラグインやエフェクトプラグインの作成。サウンドエンジンのプラグインアーキテクチャにより、ゲームの独自要件に合わせてWwiseの機能を拡張可能。
- 開発ツールに、Wwise Authoring APIを統合。

プロジェクトプラン

プロジェクトを開始するにあたり、オーディオプログラマーとサウンドデザイナーが協力して、ゲームタイトルのオーディオに割り当てられる様々なリソースについて確認して下さい。リソースとは、メモリ、ディスクスペース、ストリーム数、サウンドバンクのサイズなどを指します。

デザイナーと開発者は、ゲームのデザインや技術的な制約を考慮した上で、下記の項目を協議して決めて下さい。

- ゲームエンジンにイベントを統合し、トリガーさせる方法。
- プロジェクト階層と、チーム作業のためのワークユニット。
- プレイヤーの視野とリスナーの位置。
- サウンドバンクのロード・アンロード戦略。
- 音楽の統合方法とインタラクティブミュージック独特の要件。
- RTPCとして採用するゲームパラメータの選定。
- ミックスやステートのメカニズムをドライブする、ゲーム内のグローバルエレメントの選定。
- スイッチメカニズムを使うサウンド構成の選定。

第13章 まとめ

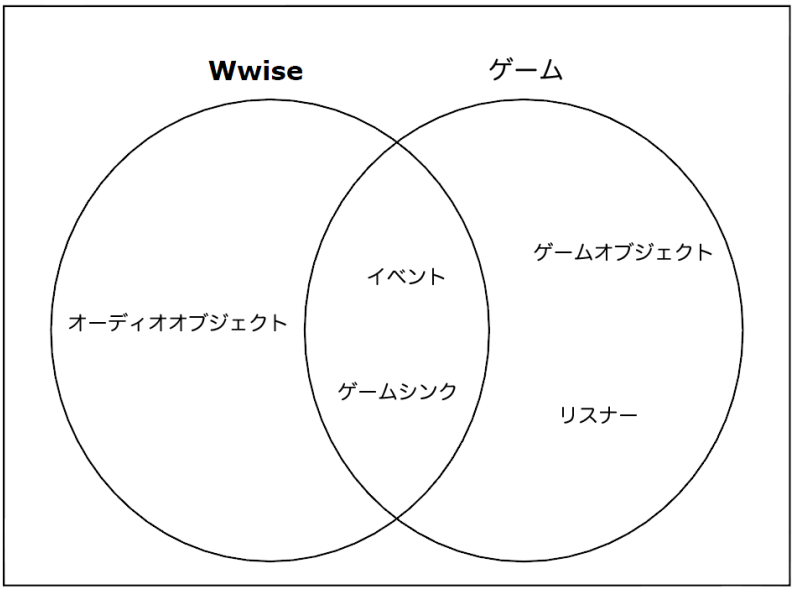
まとめ

Wwiseは開発当初より、サウンドデザイナーとプログラマーの明確な役割分担を目指してきました。どちらも専門性の高い分野であり、互いにゲームオーディオを更に高いレベルに引き上げるタスクに専念できれば、ゲームの質が格段に向上するでしょう。

サウンドデザイナーとプログラマーの役割分担に配慮することで、Wwiseの核となる５つの主要コンポーネントが明らかになりました。

- オーディオオブジェクト
- イベント（Events）
- ゲームシンク（Game Syncs）
- ゲームオブジェクト
- リスナー（Listeners）

下図の通り、各コンポーネントはサウンドデザイナーまたはプログラマーのいずれかの担当となります。



Wwiseとゲーム本体にとって不可欠な２つのコンポーネントが、イベントとゲームシンクです。この２つのコンポーネントがゲームのオーディオをドライブし、Wwise内にあるオーディオアセットとゲームが管理するコンポーネントをつなぐ、重要なブリッジとなります。

Wwiseは、オーディオの開発プロセスとビデオゲームへの統合における、パラダイムシフトを表しています。確かにゲームデザイナーと開発者が新しいアプローチを習得する必要がありますが、作業の効率化につながり、自分の専門分野に専念できる環境が整います。

これで、ゲームオーディオの開発に対するWwiseのアプローチの基本をご理解いただけたと思いますので、Wwiseの機能を最大限にご活用ください。