audiokinetic

Wwise 基础知识

2017.2.4

Wwise 基础知识

Wwise 2017.1.0 Revision 3176

目录

表

例

第 1 章 Wwise 简介

Wwise 简介

在深刻认识到声音设计师和音频程序员需求的基础上，Audiokinetic 创建了创新的专业音频设计解决方案 Wwise。经过多年的发展，Wwise 在开发中遵循了以下理念：

- 提供完整的制作解决方案。
- 重新定义音频和振动（motion）的制作工作流程。
- 提高管线效率。
- 使用音频和振动拓展游戏沉浸体验。

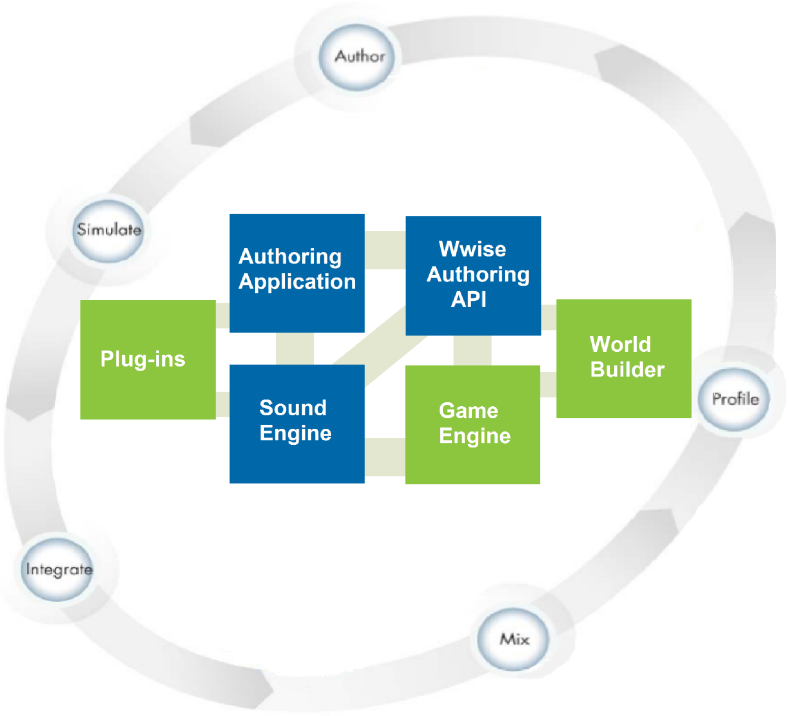
这款强大全面的音频管线解决方案包括以下部分：

- **强大的创作应用程序**——非线性创作工具，用于创建音频和振动素材结构，定义传播，管理声音、音乐和振动集成，分析播放性能，以及创建 SoundBank（声音库）。
- **创新的声音引擎**——先进的声音引擎，用于管理音频和振动处理，执行全面的多样化功能，并针对每款平台进行了高度优化。
- **Game Simulator（游戏模拟器）**——LUA 脚本解释器，用于准确再现声音和振动在游戏中的运行情况，使您能够先验证 Wwise 在每款平台上的特定行为和分析其性能，然后再将 Wwise 集成到游戏的声音引擎中。
- **插件式架构**——全面可扩展的插件式架构，可以快速拓展游戏中的音频沉浸式体验。可提供多个插件，包括：
  - 用于生成音频和振动的源插件（Source plug-in），例如 Tone Generator（乐音发生器）。
  - 用于创建音频效果（例如混响）的效果器插件（Effect plug-in）。
- **Wwise 与游戏编辑器之间的接口（Wwise Authoring API）**——与外部游戏游戏编辑器或 3D 应用程序之间的专用插件接口，使外部应用程序能够无缝地与 Wwise 通信。所有使用声音引擎 API 通常可以修改的内容，在 Wwise Authoring API（声音构架）中您都可以轻松修改。

Wwise 制作管线

Wwise 的基础是制作管线。它是一种紧密集成各种必要工具的创新工作方式，使您可以在游戏中实时执行各种任务。

- **创作**——创建声音、振动和音乐结构，定义属性和行为。
- **模拟**——验证艺术方向和模拟游戏体验。
- **集成**——早期集成，无需额外编程。
- **混音**——在游戏中实时混合属性。
- **性能分析**——实时执行性能分析，确保遵循游戏约束。



Wwise 工程

Wwise 系统是基于工程的，这意味着，对于特定游戏，所有平台的全部音频和振动信息都集中在一个工程中。

在此工程中，您可以执行以下任何或所有操作：

- 管理游戏中的声音、语音、音乐和振动素材。
- 定义对象属性和播放行为。
- 创建触发游戏音频和振动的 Event（事件）（包括动作和对白 Event）。
- 创建原型和模拟。
- 排查故障，分析工程中音频和振动所有方面的性能。

工程还包括为所有指定平台和语言版本而生成的 SoundBank。

Wwise 如何管理工程中的素材

一款典型的游戏可拥有数千个声音、音乐和振动素材，所以您的 Wwise 工程必须能够高效地管理这些素材，特别是在为每个平台和每种语言创建不同的游戏版本时。

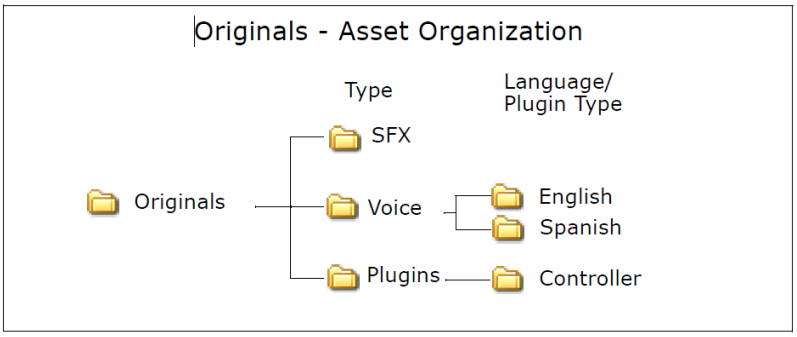
Originals 文件夹

首先需要知道的是 Wwise 无破坏性，这意味着您可以在工程中编辑素材，但不会对原始文件本身造成任何影响。当您将文件导入 Wwise 时，工程的“Originals”文件夹中会保留一份文件副本。

根据您导入的文件类型，文件将保存在以下文件夹之一中：

- 插件：
- SFX
- Voice（语音）

如果文件被标记为 Voice（语音）或 插件文件，它们将按照语言或插件类型进一步细分。下图演示了 Wwise 如何组织导入到工程中的原始素材。



平台版本

在 Wwise 的 “Originals”文件夹中，您可以为每个游戏平台创建对应的版本。这些平台版本存储在工程的“cache”（缓存）文件夹中。

为帮助 Wwise 更加高效地管理“Cache”文件夹的内容，经过转换的素材按以下标准划分：

- 平台（Windows®、Xbox One™、PlayStation®4、Switch 等）
- 类型（插件、SFX 或语音）。如果素材被标记为 Voice 或 Plug-in 文件，它们将按照以下标准进一步细分：
  - 语言 （英语、法语、西班牙语等）
  - 插件类型（控制器）

第 2 章 集成游戏音频

Wwise 基本方法

在开始编写代码和使用 Wwise SDK 前，您应该了解 Wwise 用于创建音频并将其集成到游戏中的独特方法。为了高效地工作和充分利用 Wwise，您还应该熟悉一些概念。

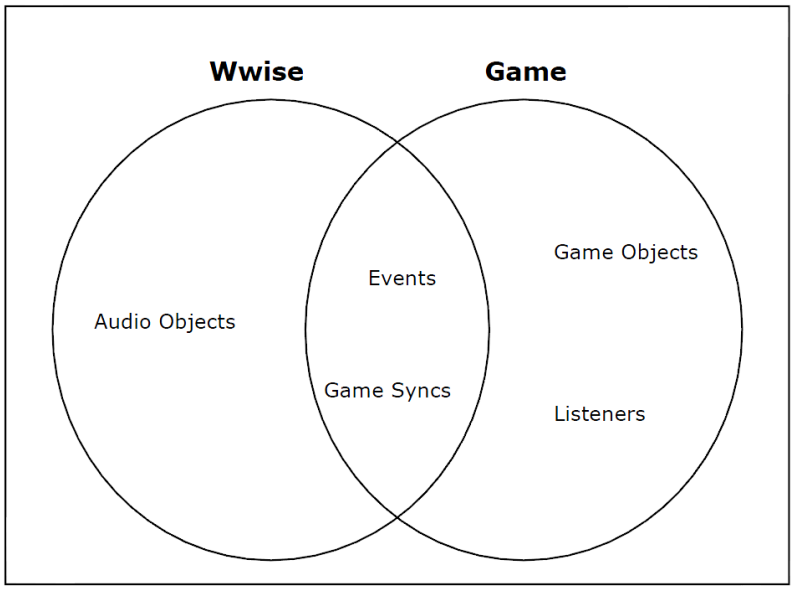
Wwise 用于创建音频并集成到游戏中的方法包括五个主要组件：

- 音频对象
- Event（事件）
- Game Sync（游戏同步体）
- Game Object（游戏对象）
- Listener

下面各节将以更多的篇幅详细讨论每个组件，但是在此之前，您应该了解每个组件的用途，以及它们之间的相互关系。

Wwise 的目标之一是划清程序员与设计师之间的任务分工。例如，代表游戏单个声音的 Audio Object 完全由声音设计师在 Wwise 应用程序中创建和管理。而 Game Object 和 Listener（代表发送或接收音频的特定游戏元素）由程序员在游戏中创建和管理。最后两个部分（Event 和 Game Sync）用于驱动游戏中的音频。这两个组件在音频素材和游戏组件之间架起桥梁，因此是 Wwise 和游戏中都不可分割的一部分。

下图演示了创建和管理每个组件的位置。



第 3 章 工程层级结构

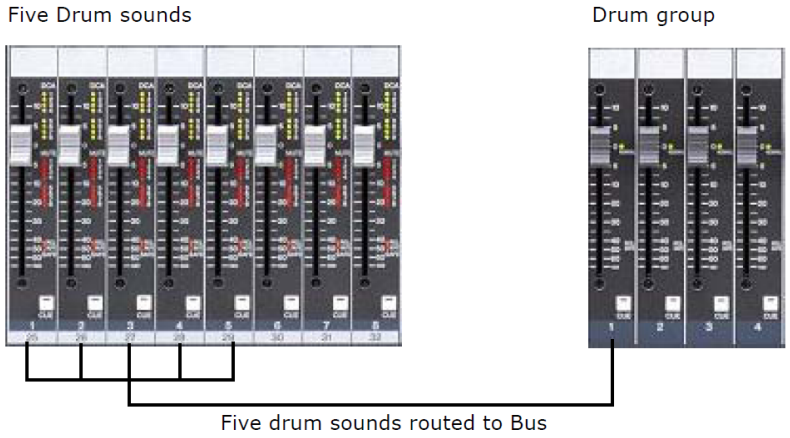
工程层级结构

导入工程中的素材是工程层级结构的基础。工程层级结构从传统混音技术演化而来。在传统混音技术中，通过将不同乐器的通路指派到一条总线，您可以将它们的声音属性作为一个混音进行控制。例如，可以将踩镲声、击镲声、碎音钹声、底鼓声和军鼓声的通路全都指派到一条总线，您可以将它们将它们混合后作为一个整体来控制其音量和其他参数。

Wwise 中采用类似的方法对工程中的声音、振动（motion）对象和音乐进行组织和分组。通过采用这种方法对声音、振动和音乐对象分组，您即开始创建层级式项目结构，在各种对象之间创建主从关系。这种创建和管理游戏音频和振动的独特高效方法可以为您创建具有真实感的沉浸式游戏环境提供更大的控制权和灵活性。

Wwise 工程层级结构包括三个不同的级别：

- **Actor-Mixer Hierarchy（角色混音器层级结构）**——使用一系列 Wwise 专用对象来分组和组织工程中的所有声音和振动素材。
- **Interactive Music Hierarchy（互动音乐层级结构）**——使用一系列 Wwise 专用对象来分组和组织工程中的所有音乐素材。
- **Master-Mixer Hierarchy（主混音器层级结构）**——使用一个或多个输出总线定义不同声音、振动和音乐结构的连线和输出。



Wwise 工程层级结构支持工作组

在当今游戏开发环境中，团队协作至关重要。虽然每个游戏只可使用一个 Wwise 工程，但是您可以将 Wwise 工程层级结构划分为不同的工作单元，这样不同的人员就可以同时参与工程。Work Unit（工作单元）是特殊的 XML 文件，其中包含与工程中某一特定部分或元素有关的信息。这些 Work Unit 可帮助您组织和管理工程中的各种元素。如果您参与团队协作，这些 Work Unit 还可以由版本控制系统来管理，团队中的不同成员可以更加容易地同时参与工程。

理解 Actor-Mixer Hierarchy

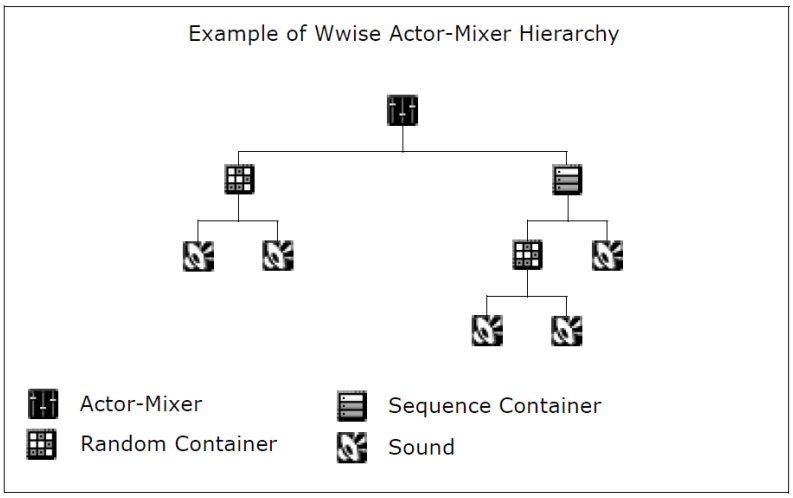
Actor-Mixer Hierarchy（角色混音器层级结构）用来分组和组织工程中的所有声音和振动素材。层级结构的最底层是所有单独的声音和振动对象。您可以定义这些单独对象的属性和行为，还可以将这些对象组合在一起，作为一个整体来定义它们的属性和行为。考虑到游戏中音频的复杂性，Wwise 工程层级结构中允许同时存在多种不同的

对象。每种对象都有一组属性（例如音量、音高和位置）和一组独特的行为（例如随机或顺序播放）。通过使用不同的对象类型对工程层级结构中的声音分组，可以对游戏中的声音组定义特定的播放行为。您还可以在层级结构中的不同层次上来定义这些属性和行为，以此获得不同的结果。

由于振动一般绑定到游戏的音频上，因此 Wwise 采用相同的原理和工作流程来生成振动。这意味着您可以使用层级结构来组织游戏的振动素材，并采用与音频素材相同的方式来指定属性和行为。

您可以组合使用以下对象类型来对素材分组和为工程搭建结构：

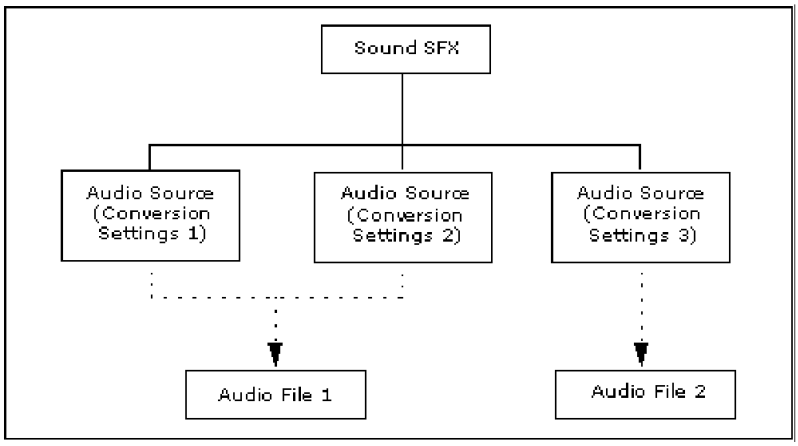
- Sound Object（声音对象）
- 振动 FX（振动效果）对象
- Container（容器）
- Actor-Mixer（角色混音器）



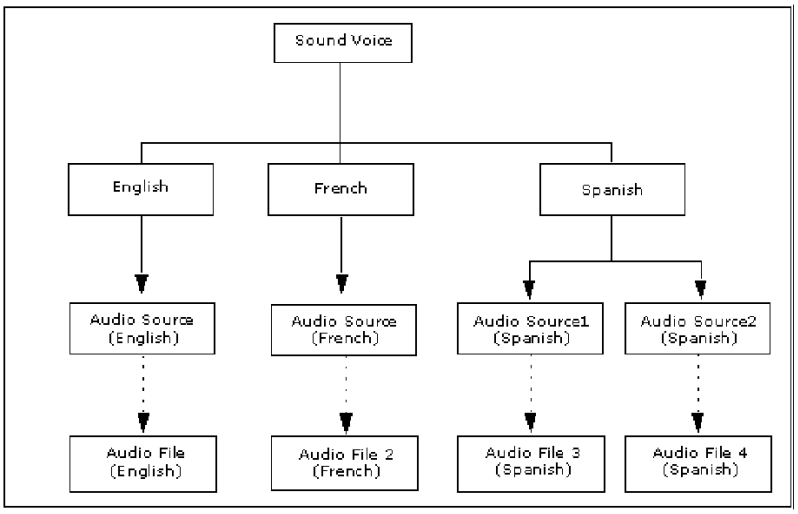
音频对象

在 Wwise 中，游戏中的各种语音和 SFX（音效声）素材用音频对象（Audio Object）来表示。音频对象是一种特殊的音频对象。这些声音对象包含链接到原始音频文件的源（source）。

音频源是导入的音频文件与声音对象之间的独立层。通过添加这样的抽象层，您可以将多个源和音频文件全部置于同一个声音对象之中。



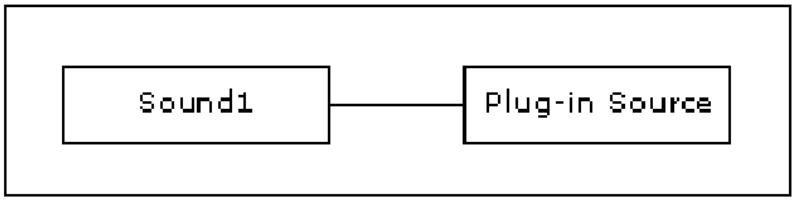
这不仅便于测试不同的转码设置，而且可以高效地管理多语言版本的开发。



注意：Wwise 采用类似的方法管理工程中的音乐和振动素材。

Source 插件

声音对象不仅支持音频源（Source），而且还支持插件源。

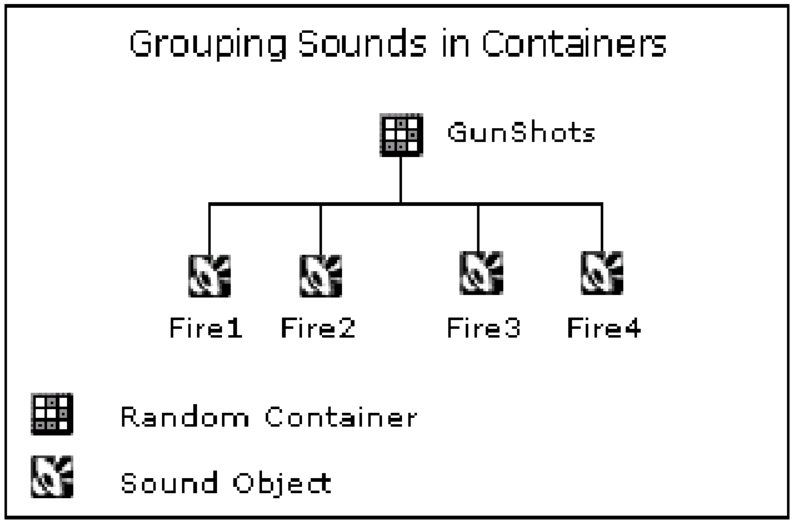


Wwise 配备各种各样的源插件，包括 Tone Generator（乐音生成器）、Silence（空白）和 Audio Input（音频输入）插件。它们除了可以直接用于制作管线外，其主要用途是与随附的源代码一起，供程序员创建自己的源插件时作为参照。由于音频对象的创建和管理由声音设计师在 Wwise 中完成，现在程序员可以解放出来，自由地开发各种源插件，突破音频设计的极限，提升游戏的总体体验。

构建音频对象的层级结构

音频对象可以组合起来创建层级式的工程结构。音频属性和行为可以应用于层级结构的不同级别，为您提供创造真实沉浸式游戏体验所需要的控制权和灵活性。

在工程中，容器用来对声音对象分组。它们主要用于根据特定行为（例如随机、顺序、切换等）来播放一组对象。例如，可以将所有枪击声集中到一个随机容器中，以在游戏中每次开枪时播放不同的声音。



所有音频对象被连线到总线层级结构中后，还可以在总线上作为整体来附加属性和效果。

音频对象——角色和职责

下表说明了与音频对象相关的哪些任务属于声音设计师的职责，哪些属于程序员的职责：

表 3.1. 音频对象——角色和职责

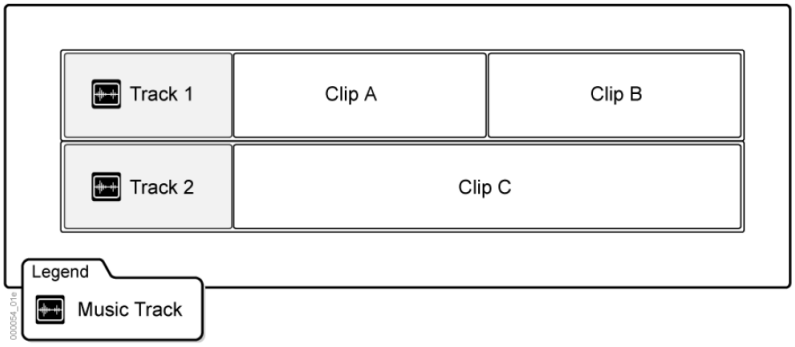
任务	声音设计师（Wwise）	程序员（游戏代码/工具）
为游戏音频素材创建声音对象	X	
对对象进行分组和构建工程层级结构	X	
定义声音属性和行为	X	
通过总线为音频对象连线	X	
开发源插件		X

理解 Interactive Music Hierarchy

在为工程创建互动音乐时，Wwise 可以为您提供高度的灵活性。有无数种方式可以将互动音乐对象汇编成游戏配乐。然而，在结构上保持一致性可以让工作流程变得更加高效。

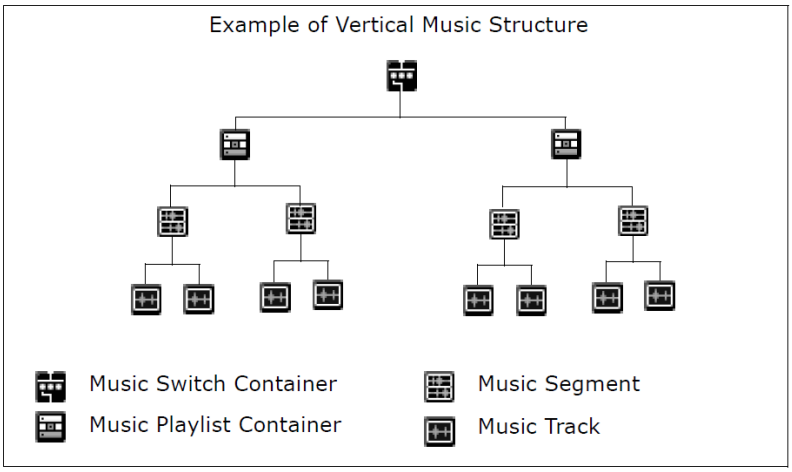
以下是可用于互动音乐工程的两种基本结构：

- **纵向工程结构**。在这种工程结构中，您可以通过打乱音乐段落中包含的音轨来对游戏配乐进行重新排序。它类似于音乐制作中使用的音轨混音，可以帮助您利用多轨长段落制作多样化的配乐。



- **横向工程结构**。在这种工程结构中，您可以在既定时间点上动态选择所要播放的段落来令游戏配乐多样化。为此，您可以像排列 Actor-Mixer Hierarchy 中的对象一样来排列 Interactive Music hierarchy 中短的离散段落。通过这种方法，您可以使用精选的短音乐段落制作极具感染力的配乐，同时尽可能降低游戏机的性能需求。

一般您需要组合使用这两种结构，以便高效地利用工程中可以使用的资源。一个好的结构可以让您制作出动听的音乐，并充分利用游戏机资源。



理解 Master-Mixer Hierarchy

Actor-Mixer 和 Interative Music Hierarchy 之上是 Master-Mixer hierarchy（主混音器层级结构）。Master-Mixer hierarchy 是独立的总线层级结构，可以让您重新编组和混合工程中的大量不同声音、音乐和振动结构，让它们可以输出。Master-Mixer hierarchy 分为两个部分：一个部分为声音和音乐，另一个部分为振动。每个部分包括一个顶级“Master Bus”（主总线）及其下层的任意数量的子总线。

您可以选择使用游戏中的主要音频种类作为总线，并通过它们为声音、音乐和振动结构连线。例如，可以将所有不同的音频结构分成以下四种类别：

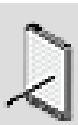
- Voice（语音）
- Ambience（氛围）

- Sound Effect（声音效果）
- Music（音乐）

这些总线不仅为工程中的声音、音乐和振动结构创建最终控制级别，而且还可以确定哪些声音受到环境效果的影响，例如混响。由于它们位于工程层级结构的顶端，因此您可以使用它们为游戏创建最终混音。根据平台情况，也可以在总线上添加特定效果（包括环境效果），以此来创建游戏所需要的沉浸式体验。

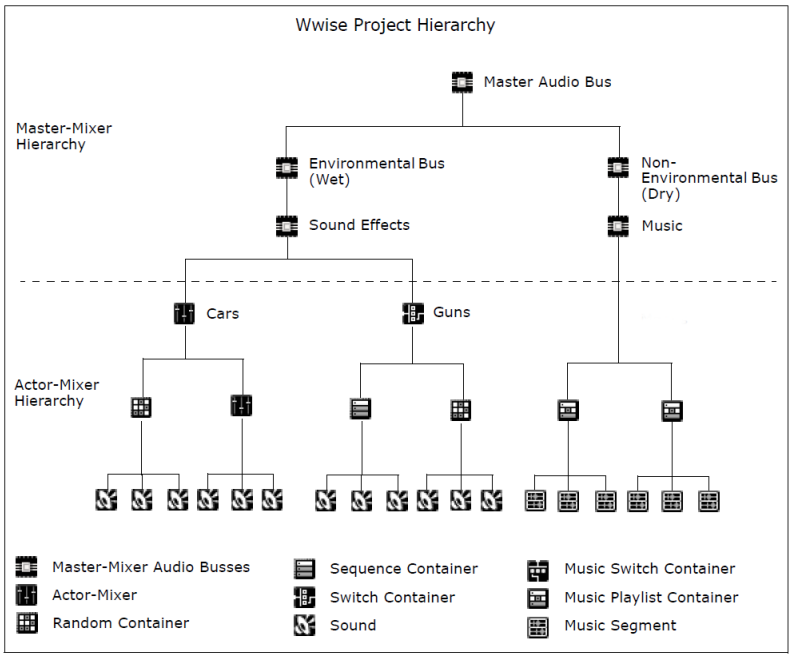
您还可以使用音频总线结构来排查游戏中的故障。例如，可以单独播放特定语音、环境声音或声音效果总线来辨别特定的声音或音乐。

下图为 Master Audio Bus（主音频总线）层级结构的示例。其中首先使用两条总线来初步区分环境声音和非环境声音，然后使用多条其它音频总线对 Actor-Mixer hierarchy 中的部分声音结构和 Interactive Music hierarchy 中的部分音乐结构重新编组。



备注

在 Master Motion Bus（主振动总线）下，可以同样为工程中的所有振动结构创建类似的层级结构。

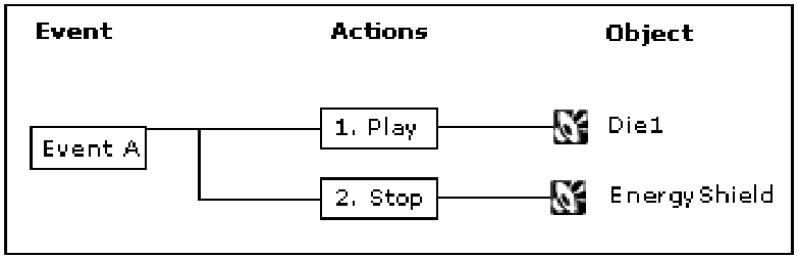


第 4 章 理解 Event

理解 Event

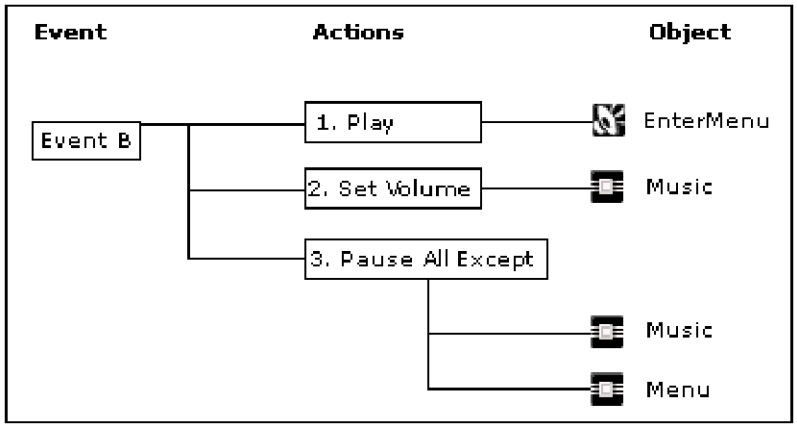
Wwise 使用 Event（事件）驱动游戏中的音频。这些 Event 将 Action（动作）应用于工程层级结构中的不同声音对象或对象组。您选择的动作用于指定 Wwise 对象是否播放、停止、暂停等。例如，假设您正在研发一款第一人称射击游戏，想要创建一个在玩家被打死时使用的事件。此事件将播放一个特殊的“Die”（死亡）声音，并停止当前正在播放的“EnergyShield”（能量护盾）声音。

下图演示了此 Event 在 Wwise 中如何工作：



声音设计师可以从长串的动作类型列表中选择一个动作类型来驱动游戏中的音频，包括 Mute（静音）、Set Volume（设定音量）和 Enable Effect Bypass（启用效果旁通）等。例如，假设您创建了当玩家退出游戏进入菜单时使用的第二个 Event。此 Event 将播放“Enter_Menu”（进入菜单）声音，将音乐总线音量减小 10dB，并暂停其他一切声音。

下图演示了此 Event 在 Wwise 中如何工作。



为了满足尽可能多的情况，Event 分为两种不同的类型：

- **Action Event（动作事件）**——这些 Event 使用一个或多个动作（例如播放、停止、暂停等）来驱动游戏中的声音、音乐和振动。
- **Dialogue Event（对白事件）**——这些事件使用一种带 State 和 Switch 的决策树来动态地确定被播放的对象。

在 Wwise 中创建 Event 后，可以将它们集成到游戏引擎中，以在游戏中的恰当时间调用它们。Event 可以在开发流程的初期阶段创建并集成到游戏引擎中。您可以继续优化 Event，而无需将它重新集成到游戏引擎中。

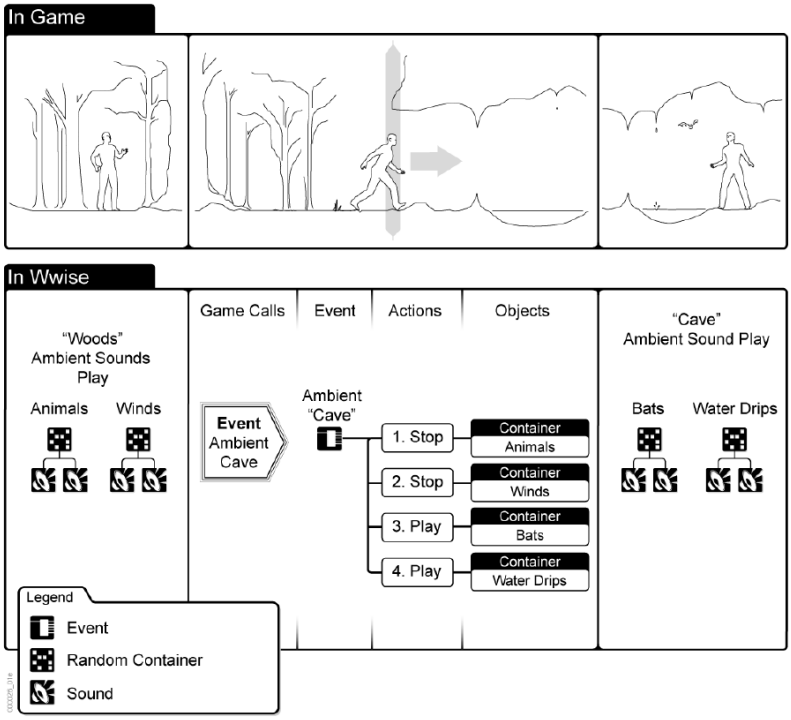
Action Event

Wwise 使用“动作”Event 驱动游戏中的声音、音乐和振动。这些 Event 将动作应用于工程层级结构中的不同结构。每个 Event 都可以包含一个或一系列动作。用户选择的动作指定 Wwise 对象是否将播放、暂停、停止等。

例 4.1. 使用 Action Event——示例

假设游戏中的角色必须进入洞穴获取一些隐藏的文件。当该角色从树林进入洞穴时，游戏中的环境声音应发生改变。要触发此改变，必须创建一个包含一系列动作的 Event，这一系列动作将停止“Woods”（树林）环境声音并播放“Cave”（洞穴）环境声音。此 Event 将被集成到游戏引擎中，当该角色进入洞穴时，游戏引擎就会调用您在 Wwise 中创建的特定 Event。

下图演示了游戏引擎如何触发 Event 来更改游戏中正在播放的环境声音：



为了处理声音、音乐或振动（motion）对象之间的过渡段，每个 Event 动作还拥有一组参数，您可以使用这些参数延迟或者淡入淡出来衔接前后的对象。

Dialogue Event

Wwise 使用 Dialogue Event 来驱动游戏中的动态对白。Dialogue Event 基本上是一组决定播放哪条对白的规则或条件。Dialogue Event 可用于重新创建游戏中存在的各种不同场景、条件或结果。为了确保用户能够覆盖所有情形，Wwise 还允许用户创建默认或备用条件。

所有条件都用一系列 Stae 和 Switch 值来定义。这些 State 和 Switch 值组合成路径，用来定义游戏中的条件或者结果。每条路径再与 Wwise 中的特定声音对象相关联。在游戏中调用 Dialogue Event 时，游戏根据 Dialogue Event 中设定的条件对现有条件进行验证。满足游戏当前状况的条件（也就是 State / Switch 路径）决定播放哪条对白。

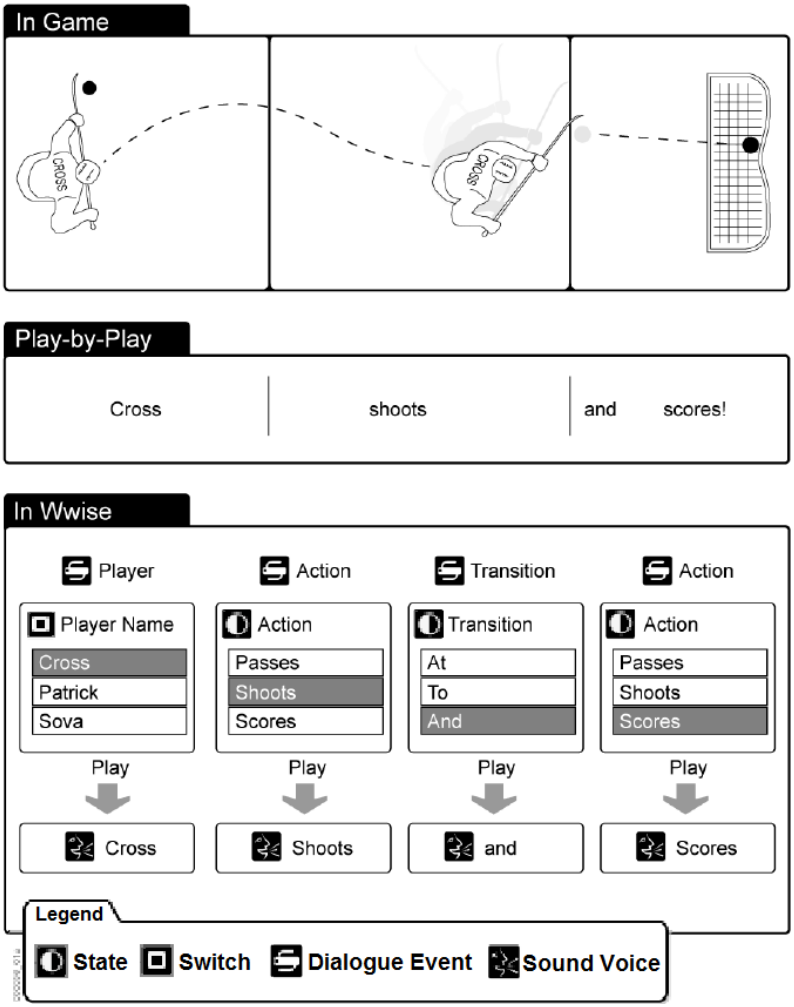
备注

虽然创建 Dialogue Event 的初衷是处理游戏对白，但它们并非仅限于为对白所用，还可用于游戏中的很多其它场合。

例 4.2. 使用 Dialogue Event——示例

假设您正在创建一款配有实况解说的曲棍球游戏。当球员射门得分时，您希望实况解说能够与游戏中的动作一致。为了在 Wwise 中建立不同的可能性和结果，您需要为解说词中的 Players（球员）、Actions（动作）、Transitions（连接词）等句子成分创建 Dialogue Event。各个 Event 包含一组您创建的 State 和 Switch 值。您必须创建 State / Switch 路径来定义各个条件或结果，然后将合适的语音对象指派给各条 State / Switch 路径。游戏会实时匹配当前的 State / Switch 值与您在 Wwise 中定义的路径，来决定播放哪个语音对象。

下图演示了 Wwise 中创建的 Dialogue Event 如何能够生成实况解说“克劳斯（Cross，人名）射门，进球了！”：



定义 Event 作用范围

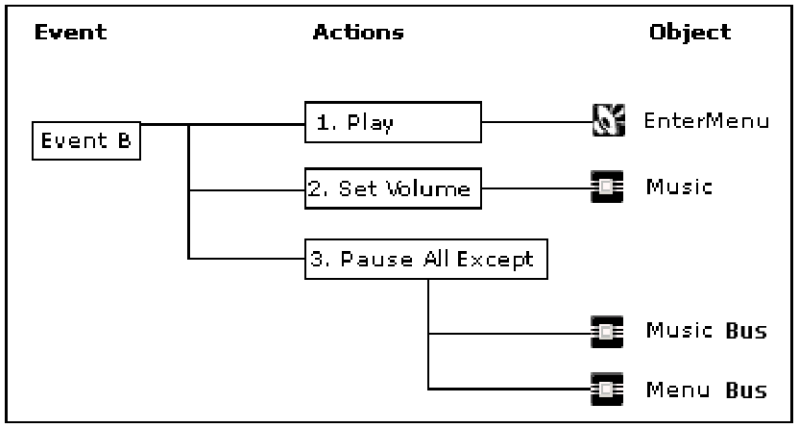
Event 中的每个动作都有相应的作用范围设置。作用范围确定 Event 动作是应用于全局所有 Game Object 还是触发该事件的特定 Game Object。对于某些动作，声音设计师可以选择作用范围，而另一些动作的范围是预先确定的。

再次以 EventB 为例，每个 Event 动作的作用范围可为如下：

表 4.1. 定义 Event 作用范围

Event 动作	作用范围	备注
Play（播放）> Menu_Enter（进入菜单）	Game Object	因为播放 Event 总是由单一 Game Object 触发，所以作用范围设定为 Game Object。
Set Volume（设定音量）> Music（音乐）	Global（全局）	因为 Set Volume 动作应用于总线，而总线从本质上讲属于全局，所以范围设定为 Global。
Pause All Except（全部暂停，例外）> Music（音乐）	Global（全局）	因为 Pause All Except 动作应用于音乐总线，而音乐总线从本质上讲属于全局，所以范围自动设定为 Global。

下图演示了此 Event 在 Wwise 中如何工作。



备注

作用范围是适用于 Wwise 中许多元素的一个重要概念。了解每个元素的作用范围可帮助您决定在不同的情形下使用哪个元素。

将 Event 集成到游戏中

在为游戏创建 Event 后，声音设计师可以将它们打包成 SoundBank（声音库）。然后将这些 SoundBank 加载到游戏中。在游戏中，游戏代码可以触发这些事件。例如，当玩家被杀后，通过触发相应事件来播放特殊的“Die”声音并停止播放“EnergyShield”声音。

为了将这些事件集成到游戏中，程序员必须指定对哪个Game Object执行事件动作。这通过发送各个事件来完成。当您希望更改音频时，应由游戏代码来发送事件。您可以使用字符串或 ID 来发送事件。

使用 Wwise 事件的好处

使用这种方法触发游戏声音的一个主要优势是声音设计师拥有更大的控制权和灵活性，无需任何额外的编程工作。所有事件由声音设计师在 Wwise 中创建，然后由程序员将它们集成到游戏中。在事件集成到游戏中后，声音设计师可以继续处理它们，更改或修改它们包含的动作或者它们所指向的对象。由于游戏仍将触发同一个事件，因此设计师所做的更改将在游戏中生效，无需开发人员额外加工，也无需重新编译代码。

Event——角色和职责

下表说明与事件相关的哪些任务属于声音设计师的职责，哪些属于程序员的职责：

表 4.2. Event——角色和职责

任务	声音设计师（Wwise）	程序员（游戏代码/工具）
创建事件	X	
将事件动作指定给音频对象	X	
定义事件动作的作用范围	X	
在游戏中发送事件		X

第 5 章 什么是 Game Object?

什么是 Game Object?

Game Object（游戏对象）是 Wwise 中的核心概念，因为声音引擎中被触发的每个 Event（事件）都与一个 Game Object 相关联。Game Object 通常指游戏中能够发出声音的特定对象或元素，包括角色、武器、环境对象（比如火把）等。然而在某些情况下，您可以将 Game Object 指定给某个游戏元素的不同部分。例如，您可以将不同的 Game Object 指定给巨人角色的不同部分，以使巨人的脚步声和配音听起来好象来自 3D 声音空间中的不同位置。

备注

如果您熟悉 Unreal 虚幻游戏引擎，Wwise 中的 Game Object 就类似于 Unreal 中的 Actors（角色）。

Wwise 为每个 Game Object 存储了各种信息，Game Object 将使用这些信息来确定在游戏中如何播放每个声音。以下所有类型的信息均可与 Game Object 相关联：

- 与 Game Object 相关联的音频对象的属性偏置值，包括音量和音高。
- 3D 位置和朝向。
- Game Sync 信息，包括 State（状态）、Switch（切换开关）和 RTPC（实时参数控制）。
- 环境效果。
- 声障（Obstruction）和声笼（Occlusion）。

备注

与其他属性不同，衰减应用于音频对象而非游戏对象。这样声音设计师可以更加灵活地单独控制每个音效的衰减。Wwise 中的 3D Game Object（3D 游戏对象）视图可以让声音设计师查看音效所关联的 Game Object、Game Objec 相对于 Listener（听者）的位置以及每个音效的衰减半径。

注册 Game Object

在使用 Game Object 之前，程序员需要在游戏代码中注册 Game Object。当您不再需要它们时，您应该注销它们，因为在注销与这些值相关联的游戏对象之前，声音引擎将继续存储它们的相关信息（3D 位置、RTPC、切换开关等）。

Scope——游戏对象范围与全局范围的比较

通过使用 Game Object，Wwise 引入了 Scope（作用范围）概念，事件一节中将对 Scope 进行简短的讨论。Scope 决定把属性和事件应用在游戏的哪个层次上。现在您可以选择在 Game Object 层次上或全局范围内应用这些元素。游戏的特定情景和/或正在发生的动作将决定作用范围，以及最终您在 Wwise 中所采取的方法。

例如，假设您在创建第一人称射击游戏。游戏主人公必须穿过城市街道才能抢到敌人的旗帜。当该角色在城市中行走时，您将听到他的脚步声。如果您想更改与脚步相关联的属性或音效，您只能在与对应主人公双脚的 Game Object 层次上局部应用这些更改。另一方面，如果游戏角色潜入水中，则需要更改周围环境中继续播放的所有音效，例如爆炸和车辆。在此类情形下，您需要进行全局更改。

使用 Game Object 的优势

使用 Game Object 可以简化音频管理，因为程序员仅需跟踪 Game Object ，无需跟踪单个音效。

创建 Game Object 后，程序员仅需发送事件、设置 Game Sync（包括切换开关、状态和 RTPC）以及游戏环境。在 Wwise 中，由声音设计师确定播放哪个音效和如何播放该音效的具体细节。通过这种方法，您可以在处理与各种游戏实体相关联的诸多音效时节省大量的时间。

Game Object —— 角色和职责

下表说明了哪些与 Game Object 相关的任务属于声音设计师的职责，哪些属于程序员的职责：

表 5.1. Game Object —— 角色和职责

任务	声音设计师（Wwise）	程序员（游戏代码/工具）
将 Game Object 关联到游戏中的 3D 对象	X	X
注册/注销 Game Object		X
更新 Game Object 定位信息		X
设定每个音频对象的衰减	X	
定义事件作用范围	X	

第 6 章 什么是 Game Sync?

什么是 Game Sync?

在完成初步的游戏设计后，您可以开始考虑如何使用被称为 Game Sync（游戏同步体）的 Wwise 元素来串接和处理互动音频中的变化和替换行为，这些行为也是游戏内容的一部分。您可以在五种不同的 Game Sync 类型中指明您所需要的类型，以配合提升游戏画面的质量，取得最佳的效果。

- **State（状态）**——游戏中发生的变化，将将在全局范围内影响现有音效、音乐或振动（motion）的属性。
- **Switch（切换开关）**——为在不同条件下需要不同声音、音乐或振动的特定游戏元素而提供的替换项。
- **RTPC（实时参数控制）**——以某种方式映射到可变游戏参数值的属性，使得游戏参数值的改变将导致属性本身改变。
- **Trigger（触发器）**——对游戏中自发事件的响应，将启动插播乐句（Stinger）。插播乐句是叠加在当前正在播放的音乐上并与之混音的简短乐句。

在构建游戏项目时，您必须兼顾应对质量需求、内存使用限制以及您所面临的时间约束。使用 Game Sync 可以从战略上简化您的工作，节省存储空间，帮助创建真正沉浸式的游戏体验。

理解 State

State 就如同“混音快照”，也即对游戏的音频和振动属性施加的全局偏置或调整，代表游戏中的物理和环境条件发生了变化。使用 State 可以简化设计音频和振动的方式，帮助您优化使用素材。

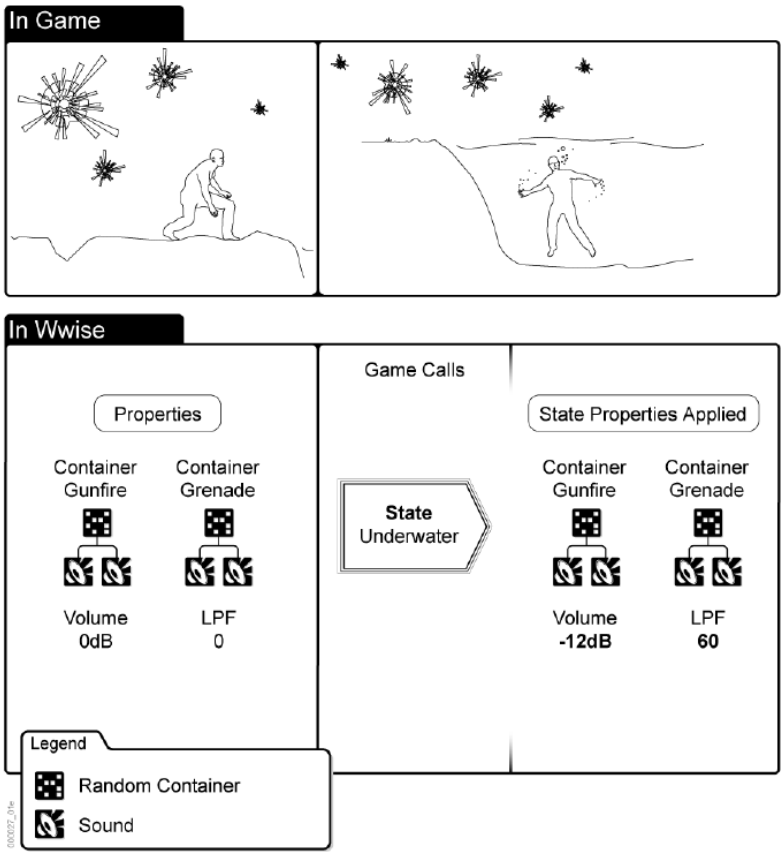
State 用作“混音快照”可以充分控制最终的声音输出，令其充满多层次的细节，并可与多种 State 组合，获得预期的结果。当一个对象上注册了多个 State 时，多个值的变化都可能会影响该对象的单个属性。在这种情况下，每个值的变化将累加在一起。例如，当两个不同 State Group（状态组）中的两个状态都让音量产生了-6 dB 的变化，且两者同时起作用时，产生的最终音量将为 -12 dB。

当创建并定义这些“混音快照”时，您实际是在为音效、音乐或振动对象创建不同的属性集，但不会增加内存或磁盘空间的占用。这些属性集定义了特定状态下播放某音效所适用的一组规则。当您将这些属性上的更改在全局范围内施加到大量对象上时，就可以快速创建逼真的音景，更好地展现音频并提升游戏体验。通过更改已经在播放的音效、音乐或振动的属性，您可以复用素材，节省宝贵的内存空间。

例 6.1. 使用 State ——示例

假设您想模拟角色入水时的声音处理。在这种情况下，您可以使用 state（状态）来更改已经在播放的声音的音量和低通滤波器。这些属性改变会造成声音上的变化，当角色位于水下时再现枪声和手榴弹爆炸声的效果需要这种声音上的变化。

下图演示了当游戏调用水下状态时，将如何影响枪声和手榴弹爆炸声对象的音量和低通滤波器属性。



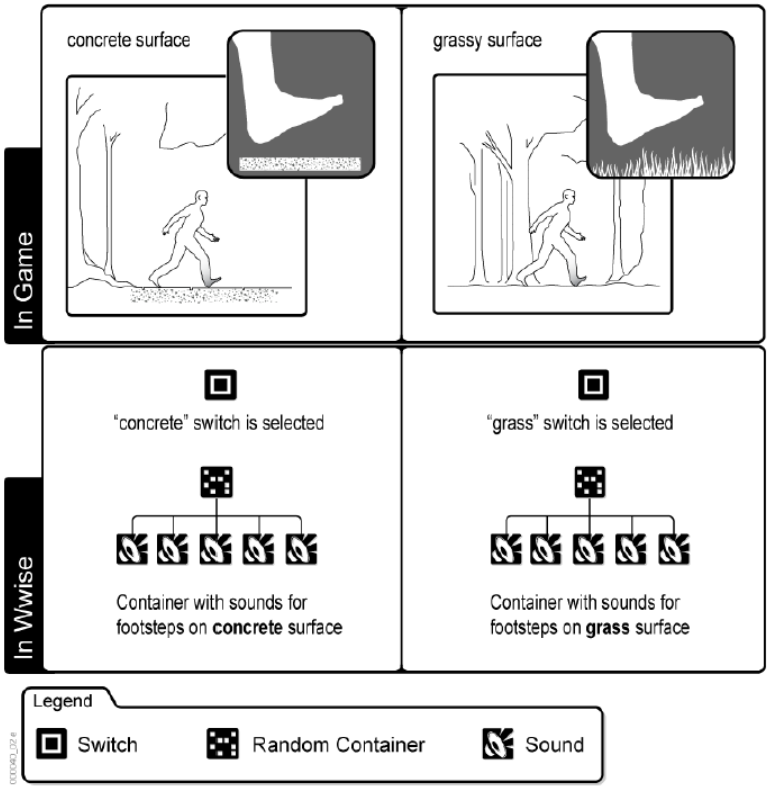
理解 Switch

在 Wwise 中，Switch（切换开关）代表游戏在不同条件下为特定 Game Objec t提供的条件替换项。声音、音乐和振动对象被组织起来并指派给各个 switch，从而当游戏中一个条件变为另一个条件时，都能够播放相应的音效或振动对象。指定给一个切换开关的各个 Wwise 对象被组合到一个 switch container（切换容器）中。当 Event（事件）发出改变信号时，切换容器确保开关被切换，并播放正确的声音、音乐或振动对象。

例 6.2. 使用 Switch ——示例

假设您正在创建一款第一人称射击游戏。在游戏中，主人公可以行走和跑步通过各种不同的环境。每个环境中有不同的地面条件，例如混凝土地面、草地和泥土地面。您希望在每种地面上听到不同的脚步声。在这种情况下，您可以为不同地面创建 switch，然后将不同的脚步声指定给适当的 switch。当主人公在混凝土地面上行走时，“混凝土”这个 switch 将被激活，并将播放其相应的音效。如果主人公接着从混凝土地面来到草地上，“草地”这个 swtich 将被激活，并将播放它相应的音效。

下图演示了激活的切换开关如何确定播放哪个脚步声。



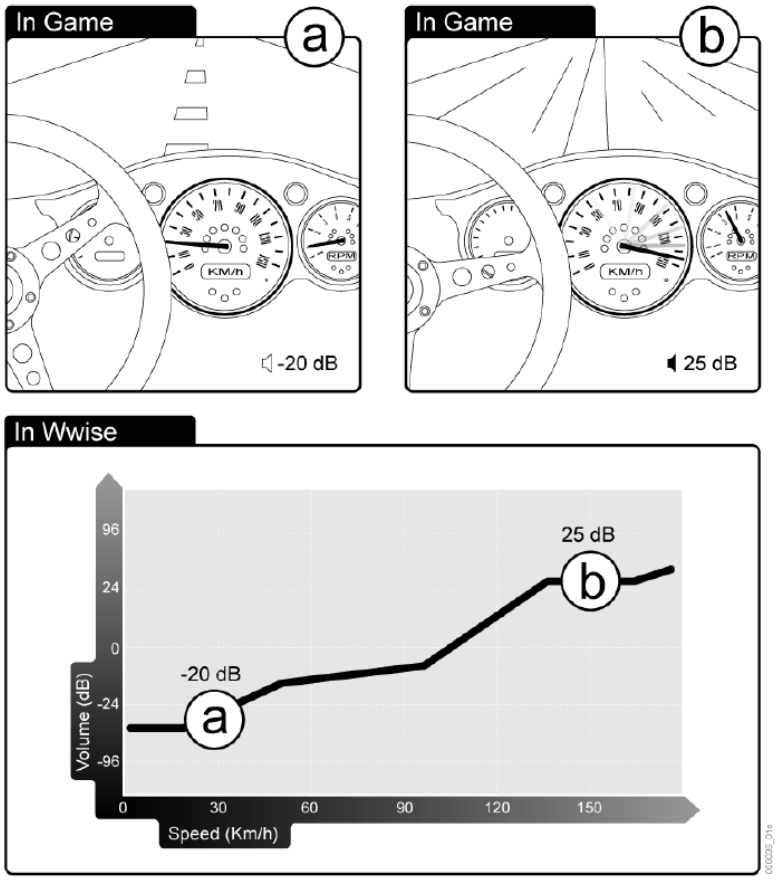
理解 RTPC

实时参数控制（RTPC）用于根据游戏中发生的实时参数值变化，来实时编辑特定的对象属性。通过使用 RTPC，您可以将游戏参数映射到属性值，并“自动执行”属性更改，增强游戏的真实感。参数值以坐标图视图形式显示，其中，一条轴表示 Wwise 中的切换开关组或属性值，另一条轴表示游戏中的参数值。通过将属性值映射到游戏参数值，您将创建一条定义两个参数之间总体关系的 RTPC 曲线。您可以创建任意数量的曲线，为游戏玩家创造丰富的沉浸式体验。

例 6.3. 使用 RTPC——示例

假设您正在创建一款赛车游戏。发动机声音的音量和音高需要随着赛车速度和 RPM（每分钟转速）的上升和下降而波动。在这种情况下，您可以使用 RTPC 将赛车发动机的音高和音量电平值映射到游戏中汽车的速度和 RPM 值上去。当赛车加速时，音高和音量属性值将根据它们的映射方式来做出反应。

下图演示了游戏中的赛车速度如何根据 Wwise 中的映射方式来影响音量。



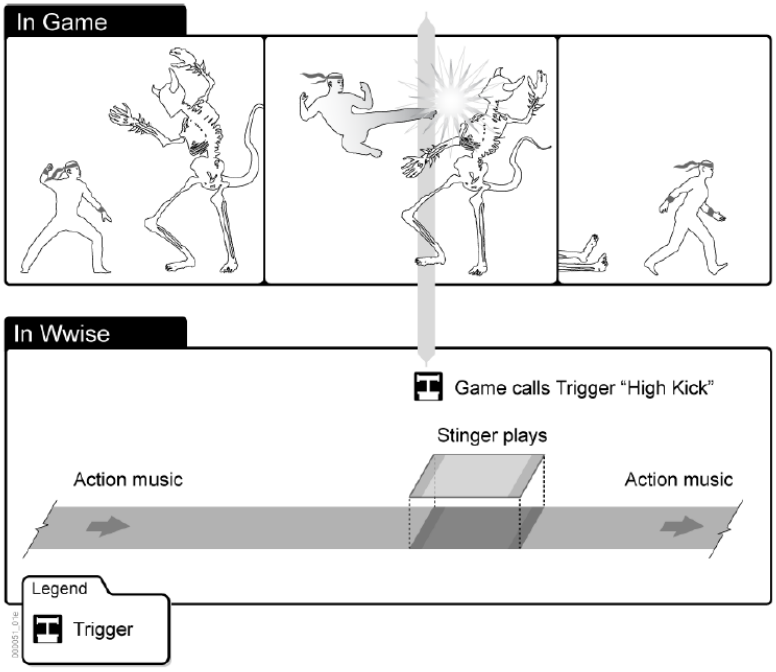
理解 Trigger

与所有 Game Sync 一样，Trigger（触发器）也是一个 Wwise 元素，游戏先对它进行调用，然后在 Wwise 中根据游戏中发生的事件来定义合适的反馈方式。更具体地说，在互动音乐中，触发器响应游戏中的自发事件，并启动 Stinger（插播乐句）。Stinger 是叠加在正在播放的音乐上并与之混音的简短乐句，也是以音乐的方式对游戏所做出的反应。例如，当一个忍者拔出武器时，您可以在已在播放中的动作音乐上插入强调类型的音乐效果，以增加场景的感染力。游戏将调用 Trigger，然后 Trigger 将启动插播乐句，于是在正在播放的配乐之外将播放一段音乐片段。

例 6.4. 使用 Trigger ——示例

假设您创建了一款格斗游戏，主人公是一位忍者斗士。在游戏的多个位置，主人公将进入战斗模式，与敌人格斗。当主人公施展强力疾风腿时，您希望设置一段音乐，加强场景的听觉感染力。为了为这些游戏动画序列创建音乐，您需要创建一个Trigger，可将它命名为“强力疾风腿”，供在游戏中的对应位置调用。另外，您需要定义这一小段音乐来提供疾风声，增加“飞踢”效果。

下图演示了在游戏的关键时刻播放插播乐句的触发器机制。



Game Sync ——角色和职责

下表说明与 Game Sync 相关的哪些任务属于声音设计师的职责，哪些属于程序员的职责：

表 6.1. Game Sync ——角色和职责

任务	声音设计师（Wwise）	程序员（游戏代码/工具）
创建 Switch Group 和 Switch	X	
创建 State Group 和 State	X	
定义State Transition Time（状态过渡时间）	X	
为 Switch 和 State Group 注册 Swtich Container	X	
设置 Trigger	X	
将游戏引擎的 State 和 Switch 信息发送至 Wwise 音频引擎		X

第 7 章 创建模拟

创建模拟

每个游戏工程都需要不断进行大量的试验，确保一切运行正常。为了帮助您完成这些任务，Wwise 拥有一个被称为 Soundcaster（声音选角器）的强大模拟环境。Soundcaster 可用于开发过程中的任一环节，可以使用工程中的任何 Wwise 对象和事件来创建音频和振动（motion）模拟。

您可以使用 Soundcaster 完成各种任务，包括：

- 原型设计和试验。
- 验证概念。
- 同时试听声音和音乐对象。
- 对游戏中的音频和振动进行性能分析。
- 对音频和振动进行混合和测试。

您不仅可以使用 Wwise 事件、声音、振动和音乐对象在 Wwise 中创建模拟，而且也可以连接到游戏，使用游戏本身触发的声音、振动和音乐来开展模拟。您创建的模拟可另存为 Soundcaster Session（声音选角器会话），您可以在后面的开发过程中随时调出存好的特定模拟环境。

第 8 章 性能分析和故障排除

性能分析和故障排除

游戏开发人员面临的一个最大挑战是在为游戏玩家创建丰富沉浸式体验的同时，需要符合各个平台的限制要求。在 Wwise 中，针对各种平台量身定制游戏音频和振动（motion）的方式有很多。然而，您可以更进一步，利用 Wwise 的 Game Profiler（游戏性能分析器）和 Game Object Profiler（Game Object 性能分析器）来测试音频和振动在每款平台上的性能表现。这两套工具可以用在制作流程的任何环节中，针对任何平台对游戏音频和振动的特定方面进行分析。您可以连接到远程游戏机，然后直接从声音引擎获取性能分析信息。通过监控声音引擎的活动，您可以检测和排查与内存、语音、媒体流、效果器、SoundBank（声音库）等相关的特定问题。您可以在游戏中执行性能分析，甚至可以在使用 Game Simulator（游戏模拟器）和 Soundcaster（声音选角器）或者使用 Wwise Authoring API 应用程序对原型进行性能分析后，再将它们集成到游戏中。

为帮助您查找需要的信息，Game Profiler 布局分为以下三个视图：

- **Capture Log（捕获日志）**——捕获并记录来自声音引擎的所有信息的日志。
- **Performance Monitor（性能监视器）**——声音引擎执行每项活动的性能图示，例如 CPU、内存和带宽。当从声音引擎中捕获到信息时，将实时显示该信息。
- **Advanced Profiler（高级性能分析器）**——一整套声音引擎衡量指标，可帮助您监控性能和排查故障。

Game Object Profiler 布局包含以下视图：

- **Game Object Explorer（游戏对象浏览器）**——Wwise Game Object 性能分析工具的控制中心，在此您可以选择要实时监视的Game Object 和 Listener。
- **Game Object 3D Viewer（游戏对象3D 查看器）**——Game Object 和 Listener 的三维视觉表示。
- **Game Sync Monitor（游戏同步体监控器）**——用于实时分析 RTPC（实时参数控制）值的工具。游戏运行期间将绘制 RTPC 值的曲线图，被监视的不同 Game Object 具有不同的 RTPC 值。

这些视图配合紧密，因此您可以找到问题区域，判定哪些事件、动作或对象造成了问题，确定声音引擎是如何处理不同元素的，然后快速高效地解决问题。

第 9 章 理解 SoundBank

理解 SoundBank

为了高效地管理游戏的音频和振动（motion）组件，Wwise 将游戏的所有音频和振动数据置于bank（库）中。bank 大体上是指包含游戏音频和振动数据、媒体或两者兼有的文件。这些 bank 在游戏的特定时间点加载到游戏平台内存中。通过仅加载必要的资源，可以针对每个平台优化音频和振动的内存占用。bank 是您所有工作成果的集合，其中包含构成游戏所需的最终音频和/或振动内容。

Wwise 中有两类 bank：

- **初始化 bank**——包含工程所有一般信息的特殊 bank，包括有关总线层级结构的信息和有关state、switch、RTPC 和环境效果的信息。Wwise 在生成 SoundBank 时将自动创建初始化 bank。初始化 bank 通常在游戏开始时立即加载，因此在游戏期间可以轻松访问所有一般工程信息。在默认情况下，初始化 bank 被命名为“Init.bnk”。
- **SoundBank**——包含事件数据、声音和振动结构数据和/或媒体的文件。SoundBank 不同于初始化 bank，一般在游戏的不同时间点加载和卸载，这样可以提高平台内存利用率。事件与工程结构元数据也可以和媒体放到不同的 SoundBank 中，这样您可以只在需要时才加载媒体文件。由于所有平台各不相同，因此 Wwise 允许您轻松地为每款平台量身定制 SoundBank，并同时生成所有平台的 SoundBank。Wwise 还为您提供排查 SoundBank 相关故障的工具，确保您符合不同平台的要求。

为帮助您更加高效地工作，Wwise 中提供了 SoundBank 界面布局。此布局包括为您的项目创建、管理和生成 SoundBank 所需要的所有视图，包括 SoundBank Manager（声音库管理器）、SoundBank Editor（声音库编辑器）、Project Explorer（工程浏览器）和 Event Viewer（事件查看器）。

File Packager（文件打包器）

为 Wwise 工程生成的 SoundBank 和任何媒体流文件可使用 File Packager 这个独立的实用工具打包。文件包（file package）将文件系统抽象出来，封装成为独立单元，这意味着您可以避免平台具体文件系统的某些限制，包括文件名长度和实际文件数量。文件包还可以帮助您更好地管理游戏的多语言版本以及游戏发布后提供的可下载内容。

第 10 章 Wwise 声音引擎

Wwise 声音引擎

您可以使用 Wwise 创建漂亮的声音、音乐和振动结构，并将它们打包成 SoundBank（声音库），但是要呈现您设计的声音和振动（motion），您还需要强大、可靠的声音引擎。Wwise 声音引擎的最基本功能是实时管理和处理游戏音频和振动的各方面需求。它可以轻松地集成到游戏开发管线中，从而融入最终的游戏作品。

这些集成和处理任务通常需要大量编程工作，但 Wwise 声音引擎可以动态创建处理管线，令开发人员可以自由定制引擎，以满足任何游戏在任何支持平台上的特殊需求。

凭借它的成熟性能以及与 Wwise 创作工具（Wwise Authoring Application）的紧密集成，声音引擎还能够执行以下功能：

- 处理由声音设计师在创作工具中创建和定义的常见播放行为，包括随机和顺序播放。
- 可直接处理由声音设计师在创作工具中创建和细调过的淡变和交叉淡变。
- 使用声音设计师在创作工具中设定的播放限制、特定优先级设置和虚声部（virtual voice）管理声音和振动对象的优先级。
- 通过简单的 API 即可直接支持变化无穷的环境效果。另外，由于声音引擎动态创建和销毁指派给环境用的通路，它可以确保所有平台上具有一致的体验，并降低内存占用率和 CPU 工作负载。
- 当游戏中的元素部分或完全阻挡了声源时，引擎支持施加听起来很自然的声障和声笼条件效果。
- 支持游戏中多达 8 个不同的 Listener（听者）。
- 包含程序调试信息，在创作工具中可以实时显示这些信息，其结果是，声音设计师可以在游戏运行时实时进行性能分析，并根据分析结果采取正确的措施。

有关 Wwise 声音引擎功能的更多信息，请参阅 Wwise SDK 文档。

第 11 章 Listener

Listener

Listener（听者）在游戏中代表话筒。Listener在游戏 3D 空间中拥有位置和朝向。在游戏期间，Listener 的坐标与 Game Objec t的位置进行比较，以便将与 Game Object 相关联的 3D 声音指定给相应的扬声器，来模模拟实的 3D 环境。程序员必须确保 Listener 的位置信息为最新状态；否则声音将会通过错误的扬声器来呈现。

多个 Listener

在单人版游戏中，您永远只能有一个视角，因此一个 Listener 就够了。然而，如果多人同时在一个系统上玩游戏，或者同时显示多个视角，则每个视角需要属于其自己的 Listener，这样才能正确地为所有视角渲染音频。Wwise 声音引擎支持多达八个 Listener。

在默认情况下，经过声明的每个 Game Object 只能指定给第一个 Listener。然而，程序员可以灵活地将任何 Game Object 动态地指定给任何 Listener，或取消这种指定。

由于声源的定位相对于每个玩家的视角并不总是合理，因此为多个 Listener 实现音频会遇到诸多挑战。这主要是因为游戏只使用一组扬声器为多个玩家再现 3D 环境。Wwise 提供各种工具和方法来弥补这一局限，为所有玩家带来尽可能真实的音频体验。有关 Wwise 如何处理多个 Listener 的更多信息，请参阅 SDK 中的“集成 Listener”一节。

Listener——角色和职责

下表说明了与 Listener 相关的哪些任务属于声音设计师的职责，哪些属于程序员的职责：

表 11.1. Listener——角色和职责

任务	声音设计师（Wwise）	程序员（游戏代码/工具）
设置 Listener 的位置信息		X
将 Game Object 指定给 Listener		X
管理多个 Listener		X

第 12 章 设计师和程序员之间的任务分工

设计师和程序员之间的任务分工

也许您已经清楚，Wwise 采取不同于其他声音引擎的声音设计和集成方式。声音设计师被赋予了更大的控制权，这意味着过去通常由开发人员处理的、耗费时间的重复性工作现在已降到了最低水平，因此声音设计师和开发人员可以专门处理更加有趣的创造性工作。省去大量对接工作后，两组人员可以各显神通，更加高效地进行协作。Wwise 认识到声音设计师和音频程序员两者之间存在角色差异，因此为他们分派了特定的任务。

声音设计师的职责

声音设计师负责：

- 创建音频层级结构和行为。
- 创建音频 Event（事件）。
- 将音频 Event 集成到世界编辑应用程序中。
- 设置声音属性和源。
- 定义 3D 声音定位。
- 为所有声音确定音频信号连线和混音要素。
- 指定实时参数控制和游戏状态。
- 为游戏中的各种环境定义效果属性。
- 定义音量、LPF 声障和声笼属性。
- SoundBank（声音库）管理和优化。
- 为多个平台定制音频。
- 平台语言本地化。

音频程序员的职责

音频程序员负责：

- 将 Wwise 声音引擎集成到游戏引擎中。
- 使用代码集成音频事件。
- 注册游戏中发声的 Game Object。
- 调用 AK::SoundEngine::PostEvent() 方法触发包含一个或多个音频动作的事件。
- 使用 Soundbank Definition File（SoundBank 定义文件）和 File Packager（文件打包工具）管理由声音设计师或游戏指定的 Soundbank 的加载和卸载。
- 根据 Listener 的坐标系统设定 3D Game Object 的位置。
- 触发状态和切换开关，并更新实时参数控制。
- 设定应用于游戏声音的每个环境效果的百分比。
- 计算每个 Game Object 相对于 Listener 的声障和声笼值。
- 管理内存资源，包括加载和卸载 SoundBank、处理流、注册插件等。
- 编写源和效果器插件。声音引擎的插件架构允许游戏开发人员扩展 Wwise 的功能来满足他们的特定游戏需求。
- 将 Wwise Authoring API 集成到开发工具中。

项目规划

在项目初期，音频程序员和声音设计师应碰头讨论，将如何为游戏音频所分配各种资源。这些资源包括但不限于内存、磁盘空间、流数量和 SoundBank 大小。

根据游戏设计和技术限制，设计师和开发人员必须共同决定：

- 游戏引擎如何集成和触发 Event。
- Project Hierarchy（工程层级结构）和 Workgroup（工作组）的 Work Unit（工作单元）组织。
- 游戏玩家视角和 Listener 定位。
- Soundbank 加载/卸载策略。
- 音乐集成与互动音乐的特别要求。
- 哪些参数应该用作实时参数控制（RTPC）。
- 游戏的哪些全局元素可以驱动 Mix（混音）和 State（状态）机制。
- 哪些声音结构需要 Switch（切换开关）机制。

第 13 章 结论

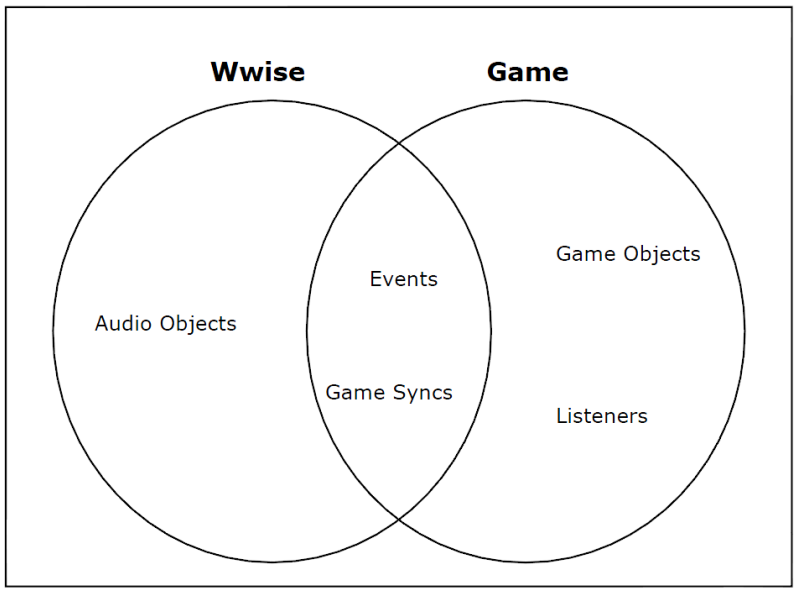
结论

自问世以来，Wwise 一直试图划清声音设计师和程序员之间的角色分工。他们各有所长，应该让他们都能各显神通，全面提升游戏音频，增强整体的游戏体验。

Wwise 的核心由五个主要组件构成，通过这些组件来划分任务：

- Audio object（音频对象）
- Event（事件）
- Game Sync（游戏同步体）
- Game Object（游戏对象）
- Listener

每个组件被分别划分到声音设计师或程序员的职责范围内，如下图所示。



Event 和 Game Sync 是 Wwise 和游戏中都不可分割的两个组件。这两个组件在游戏中负责驱动音频，在 Wwise 的音频素材和游戏中的组件之间架起必要的桥梁。

Wwise 标志着电子游戏的音频开发和集成方式发生了根本性的变化。虽然它要求游戏设计师和开发人员采用全新的工作方式，但可以帮助他们更加高效地工作，并专注于各自的专业领域。

至此您已基本了解了 Wwise 的游戏音频开发方式，现在您可以放开手脚，充分利用 Wwise 提供的所有功能了。