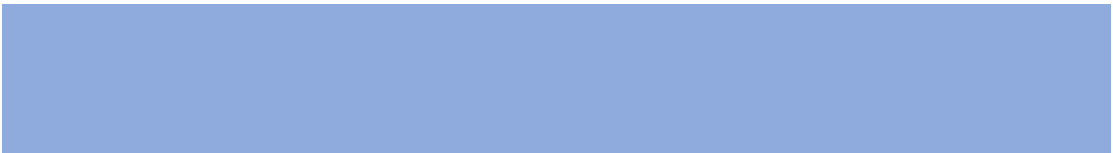




AudioEditor 用户手册

2022 年 07 月



插件介绍

AudioEditor 是西山居音频中心内部开发的基于 Unity 引擎的多功能音频编辑的一套轻量化解决方案，致力于提供丰富的音频功能，满足项目多样化的音频需求。

目前最低版本要求为 2019.3.0，推荐版本为 2020LTS 及后续长期支持版本。

目录

| | |
|-----------------------------------|-----------|
| 第一章 在 Unity 中安装插件 | 1 |
| 1.1 使用本地 Package 安装..... | 1 |
| 1.2 在 PackageManger 中安装 | 2 |
| 1.3 AudioEditor 的初始化..... | 4 |
| 第二章 快速上手..... | 7 |
| 2.1 导入音乐并播放..... | 7 |
| 2.2 暂停、恢复、停止音乐 | 9 |
| 2.3 为停止事件添加淡出效果 | 11 |
| 2.4 为音乐添加属性效果 | 11 |
| 2.5 使用容器对复数音频进行有机组织..... | 12 |
| 第三章 主界面介绍 | 15 |
| 2.1 Project Explorer Window | 15 |
| 2.2 Property Editor Window..... | 17 |
| SoundSFX | 18 |
| ActorMixer..... | 23 |
| Random Container | 23 |
| Sequence Container | 25 |
| Switch Container..... | 26 |
| Blend Container | 26 |
| Event | 28 |
| Switch Group | 29 |
| State Group | 29 |
| Game Parameter..... | 30 |
| 2.3 Help Window..... | 30 |
| 2.4 Preview Window | 30 |
| 2.5 其他说明 | 31 |
| 使用 Override 功能 | 31 |

| | | |
|--------------|---------------------------------|-----------|
| 第四章 | EventLog Window | 33 |
| 第五章 | ProjectSettings 设置 | 35 |
| 第六章 | 如何触发事件 | 37 |
| 4.1 | 通过 EventTrigger 触发 | 37 |
| 4.2 | 通过在脚本中自定义触发 | 37 |
| 第七章 | 脚本 API | 39 |
| 5.1 | AudioEditorManager | 39 |
| 描述 | | 39 |
| 静态变量 | | 39 |
| 静态函数 | | 39 |
| Events | | 39 |
| 5.2 | EventReference | 40 |
| 描述 | | 40 |
| 变量 | | 40 |
| 函数 | | 41 |
| 5.3 | RTPC | 41 |
| 描述 | | 41 |
| 变量 | | 41 |
| 函数 | | 42 |
| 第八章 | 快捷键 | 43 |

第一章 在 Unity 中安装插件

由于 1.2.3 版本起引用了 Unity 官方包 Editor Coroutines，在 Unity 2021.3LTS 版本下新建项目会默认附带此包，而在此之前的版本，请确保工程内具备 Editor Coroutines 包（通过 Package Manager 中的 Unity Registry 页面寻找 Editor Coroutines 包安装）。

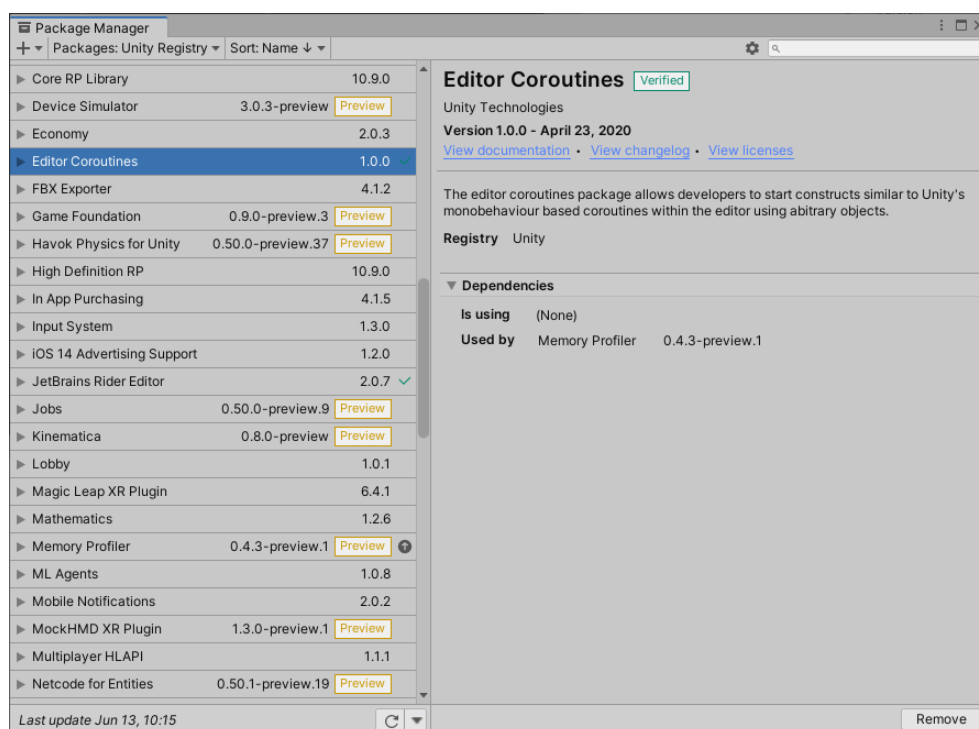


图 1 Editor Coroutines 包界面

1.1 使用本地 Package 安装

如果你已经获取了 AudioEditor 的本地包（后缀为.unitypackage），其名称格式为 AudioEditor_Unity 版本号_上线日期_版本号，可以通过本地的方式将插件载入工程。如果你未获取到 AudioEditor 的本地包，可以通过下一小节安装 AudioEditor。

打开 Unity 目标工程，在左上角菜单中选择 Assets=>Import Package=>Custom Package。选择目标 AudioEditor 本地包，点击 Import 后稍等导入即可。

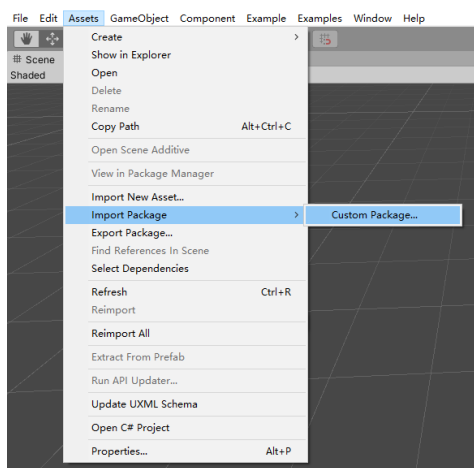


图 2 导入路径选项

1.2 在 PackageManger 中安装

如果你已经通过 1.1 节的内容安装了插件，可以跳过本小节的内容。

打开 Unity 目标工程，在左上角菜单中选择 Window=>Package Manager，打开 PackageManager 窗口，在上方 Advanced 中选择“Show Preview Packages”选项，用于显示还在预览版本的包。

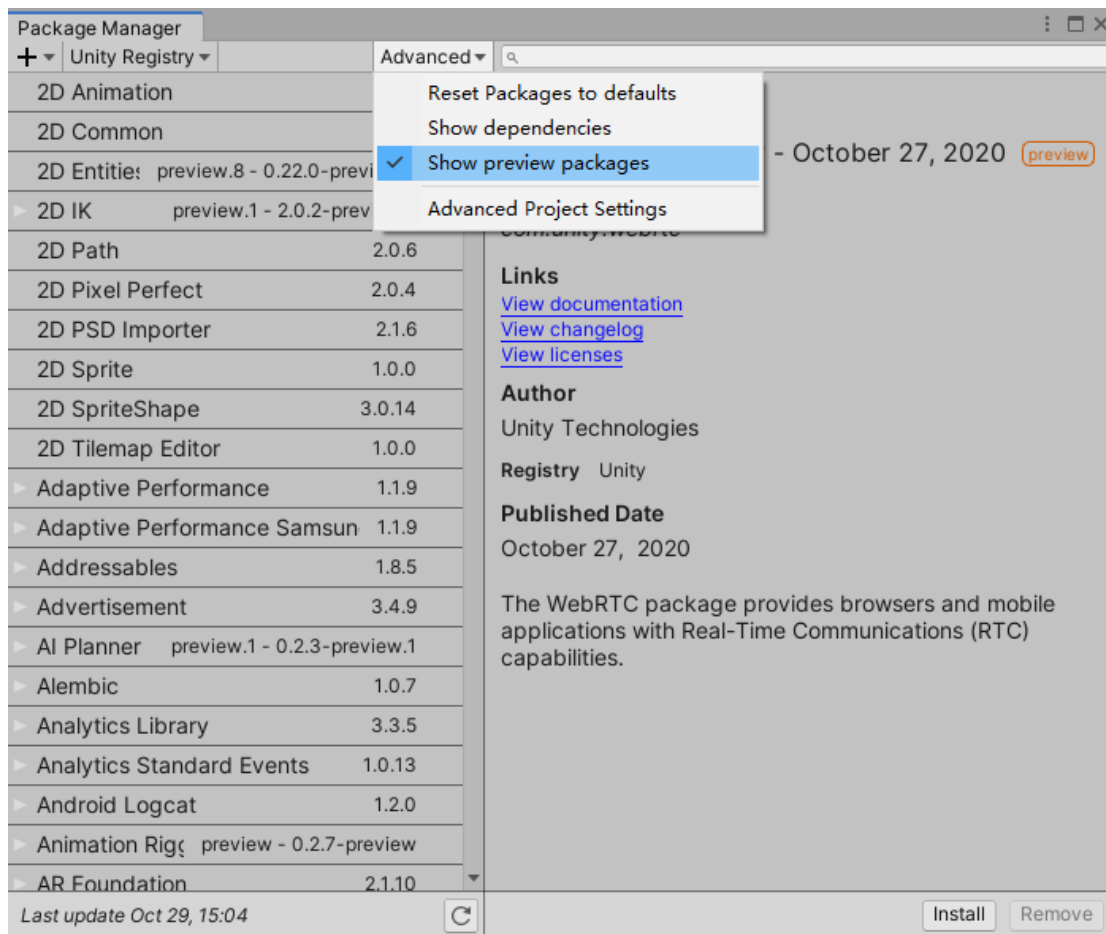



图 3 Package Manger 窗口

单击状态栏中的添加  按钮，出现添加软件包的选项，选择“Add Package from git URL”（图 1），从弹出的窗口中输入：
`https://github.com/Lijunsen/com.lijunsen.audioeditor.git`，然后单击“Add”添加，稍等片刻插件即安装并显示在软件包列表中。

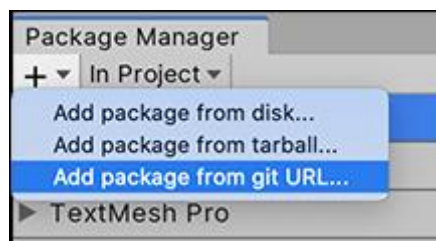


图 4 从 Git 链接中获取包

有可能会出现的其他情况：

- 如果点击添加按钮时没有弹出“Add Package from git URL”选项，请先根据以下链接安装 Git 客户端，再重新执行上述操作。

- 如果 Unity 无法安装包，则 Unity Console 控制台将显示一条错误信息，若错误信息为需要安装 Git 客户端，请同样先安装 Git，再重新执行操作。若为其他错误信息，请截图并飞书联系 LIFUAN。



- Git 安装链接: <https://git-scm.com/download/win>

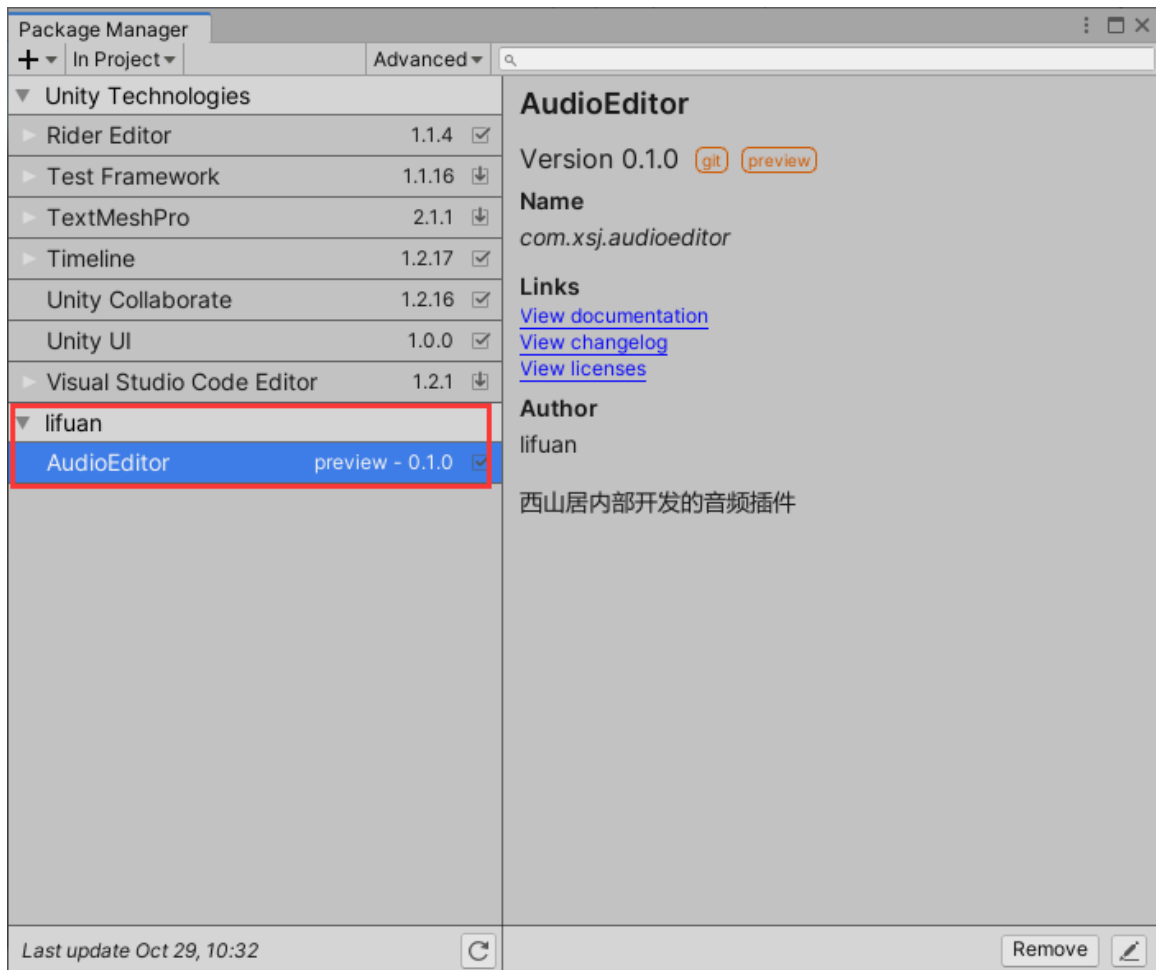


图 5 安装成功后插件显示在列表中

1.3 AudioEditor 的初始化

插件安装成功后，在 Unity 的菜单栏中 Window 标签下会新增“Audio Editor”选项，选择 Window=>Audio Editor=>EditorManager，打开主编辑器窗口。

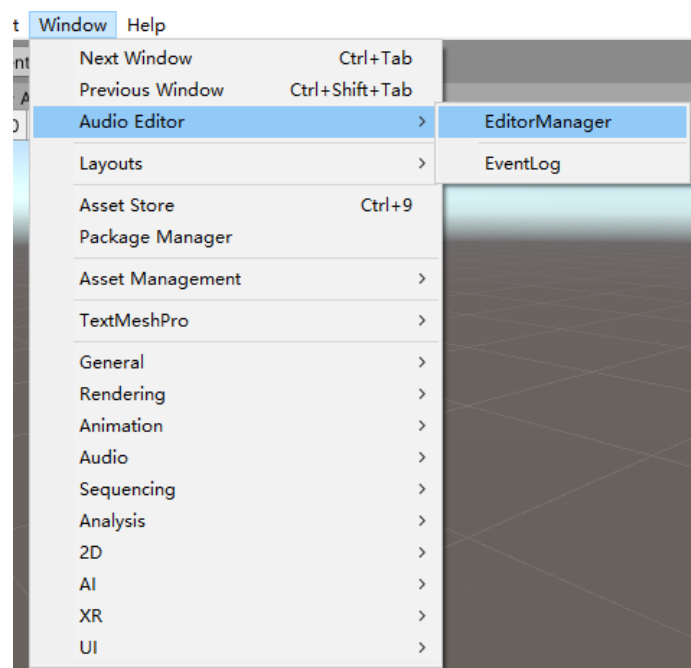


图 6 编辑器窗口的打开路径

打开主编辑器窗口后，会提示场景中没有 AudioEditorManger，单击“点击挂载 Manager”按钮，即可完成编辑器的初始化。

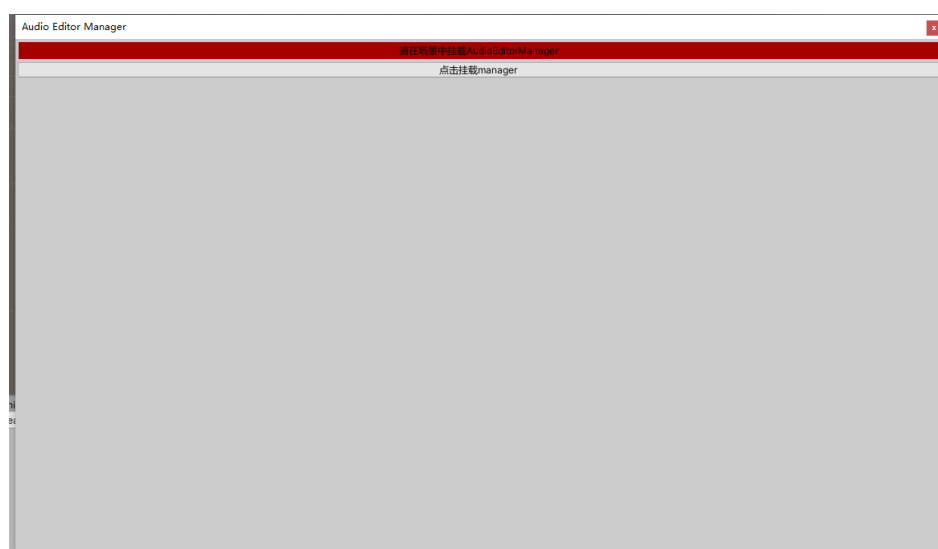


图 7 未初始化的编辑器窗口

通过初始化之后，场景中会添加一个名为“AudioEditorManger”的 GameObject，插件的许多功能都依赖此 GameObejct 运行，请勿轻易删除，编辑器也会时刻监测场景中是否有此 GameObject。

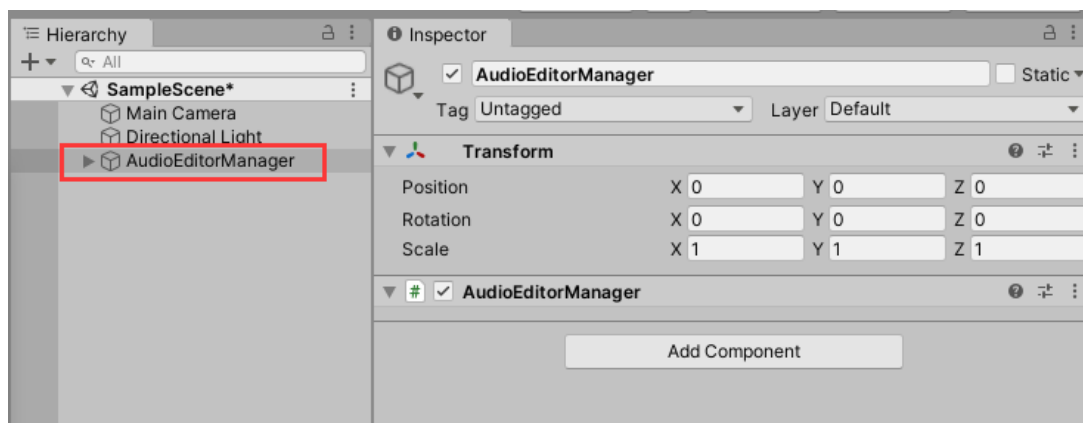


图 8 场景中添加的 AudioEditorMnager

同时在 Project Window 中新增了名为“AudioEditor”的目录结构，储存插件的配置数据，请勿轻易改动其结构。

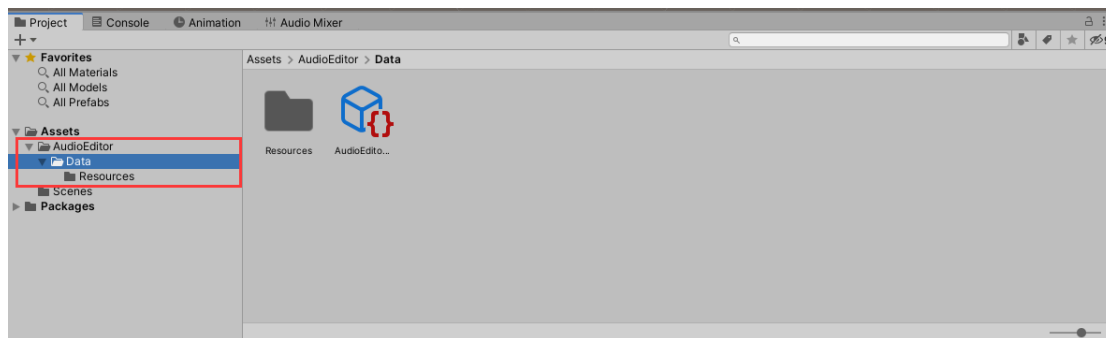


图 9 Project 窗口中新增的数据目录

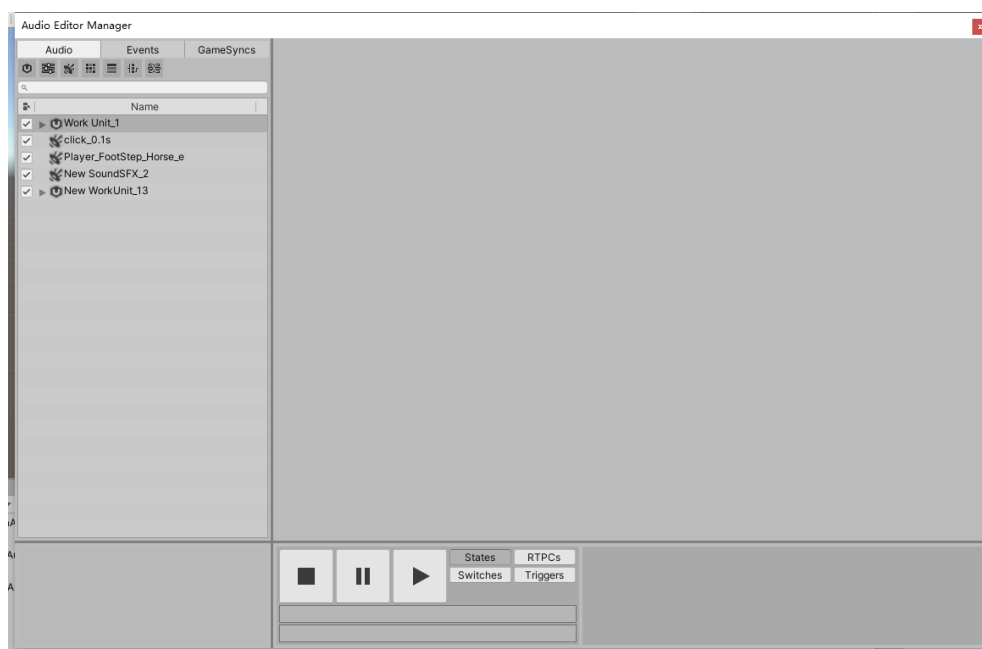


图 10 初始化完成后的编辑器界面

第二章 快速上手

本章将介绍如何快速实现音频逻辑相关功能，浅显地说明如何播放和停止音频，并实现更好地过渡等功能。如果具有 DAW 和 Wwise 相关知识的基础，本章节可以快速查看或略过。

2.1 导入音乐并播放

当我们在第一章中完成 AudioEditor 的初始化后，通过 Window=>Audio Editor=>EditorManager 打开编辑器界面，可以看到在 Audio 标签页面下的树状列表中只有一个 WorkUnit，其功能可以参见后续章节《主界面介绍》，现在我们要做的第一步是为编辑器导入音频文件。

将我们所需的音频文件导入到 Unity 工程中，并将需要的音频批量拖入编辑器的 WorkUnit 中。

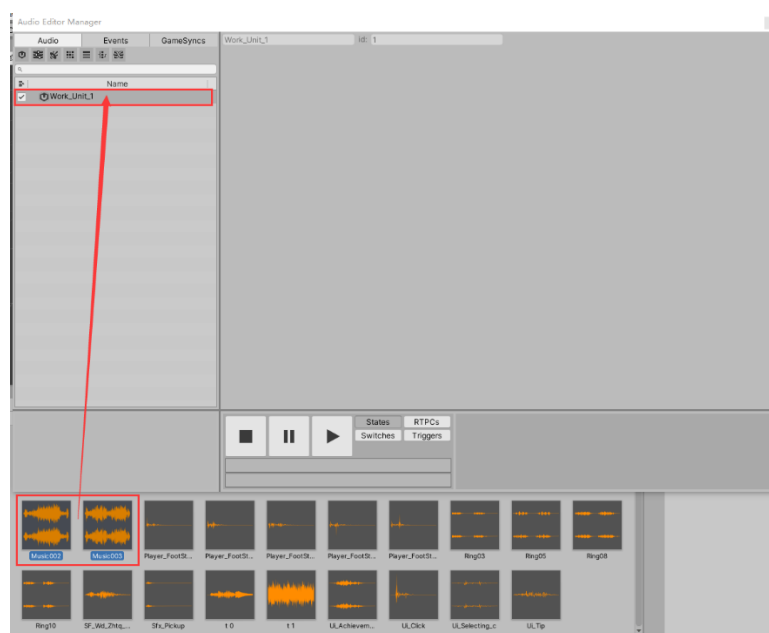


图 1 将音频导入编辑器中

导入后，编辑器会为每个音频文件单独生成 SoundSFX 组件，单击某个 SoundSFX 组件，可以看到属性窗口中有许多属性和对应音频的波形图，我们暂时先不对其进行设置，下一步我们需要为 SoundSFX 组件生成播放事件。

选择需要播放的 SoundSFX 组件（可批量选择），右键调出菜单，选择 New

Event=>Play，界面会自动转跳到 Event 标签页，并且在之前选择的位置或者默认位置生成对应的事件。

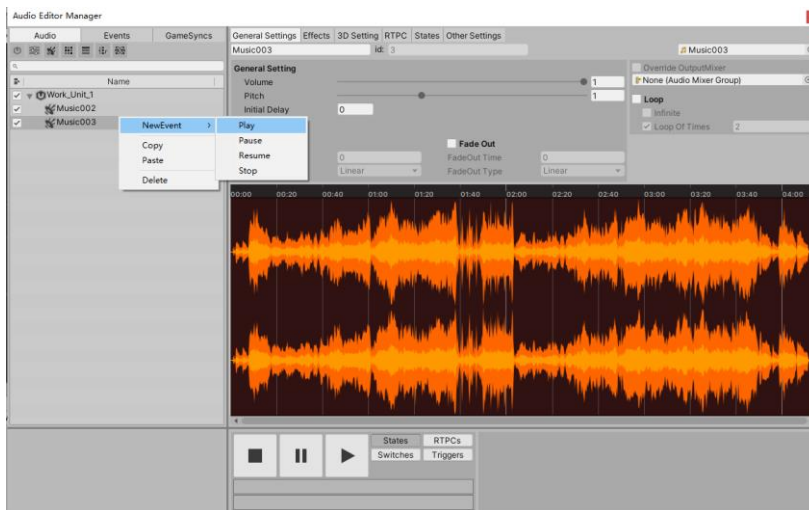


图 2 为组件生成播放事件

在 Events 标签页下，我们可以看到刚才生成的事件“Music003_Play”，单击生成的事件，可以看到右侧属性栏的动作列表中仅有一行“Play”动作，对象是“Music003”，而最右侧则显示了这项动作的相关执行属性（如何触发、触发概率、缓动方式等），我们现在暂时不需要对其进行设置。到了这步，编辑器中的工作已经完成了，使用 Ctrl+S 或关闭编辑器保存我们刚才进行的操作。我们需要在游戏中触发刚才生成的播放事件，这样音乐就会播放出来了。

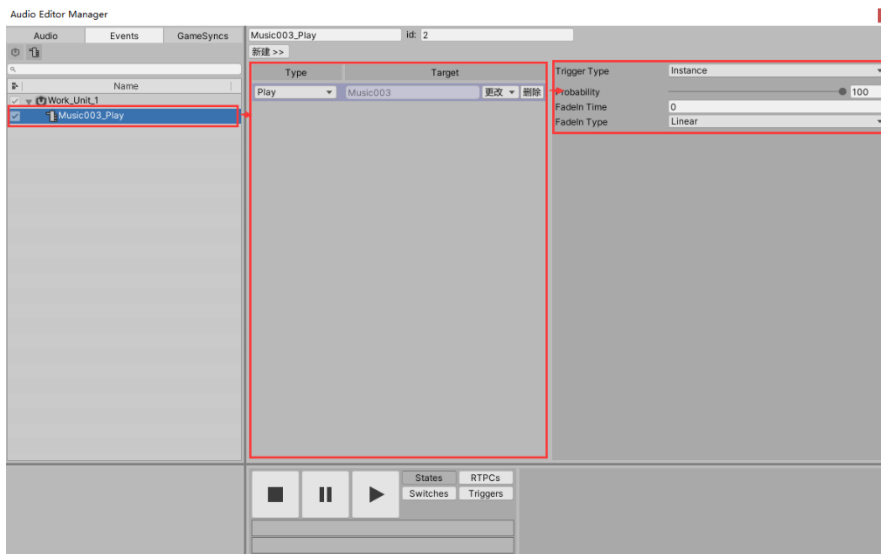


图 3 Event 标签页的层次结构

接下来我们需要在 GameObject 中进行事件触发。在 Hierarchy 面板中选择任意 GameObject，在其对应的 Inspector 面板中单击 Add Component，选择

AudioEditor=>EventTrigger 以添加 EventTrigger 组件。在组件中的 Trigger On 一栏选择对应的触发时机，我们在此选择“Start”（如果需要在脚本中执行事件触发，请参见《如何触发事件》章节）。Event 一栏选择事件“Music003_Play”，这样在此 GameObject 的 OnStart 阶段就会触发对应的事件，执行“Play”动作，播放对应的音乐。在 Unity 中点击 Play 进入 RunTime Mode，你应该能听过音乐响起。你也可以在非 Runtime 阶段点击“触发事件”和“结束事件”快速地预览事件。

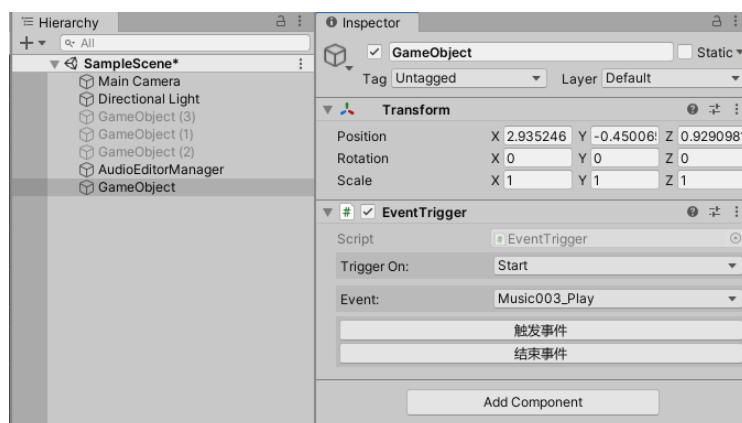


图 4 EventTrigger 组件设置后的界面

现在你应该了解了音频素材如何通过 AudioEditor 进行播放，但插件能做的远远不止这些，先别急，我们接下来学习如何控制音乐的进程。

2.2 暂停、恢复、停止音乐

在游戏过程中我们会需要频繁地将音乐暂停、恢复和停止，这一节我们使用另一种方式来生成事件（你也依旧可以使用上一节的方式快速部署事件）。

在编辑器中的 Events 标签页下，在左侧的项目浏览器中，点击“Event”组件图标，即可新建一个空事件，我们将其改名为“Music003_Pause”以更好地标识此事件，可以看到右侧属性栏中动作列表是空的，我们需要为其添加动作。点击“新建>>”按钮新建一个动作，可以看到动作的默认 Type 为“Play”，而 Target 中显示“null”，我们将 Type 选择为 Pause，在 Target 栏点击“更改”按钮，在弹出的选择框中选择“Music003”对象组件，右侧的动作属性栏我们暂时不需要设置，这样一个对“Music003”组件进行暂停的事件生成好了，重复相应的步骤，分别新建“Music003_Resume”和“Music003_Stop”事件并设置对应的动作类型

和目标对象。使用 **Ctrl+S** 或者关闭编辑器保存内容。

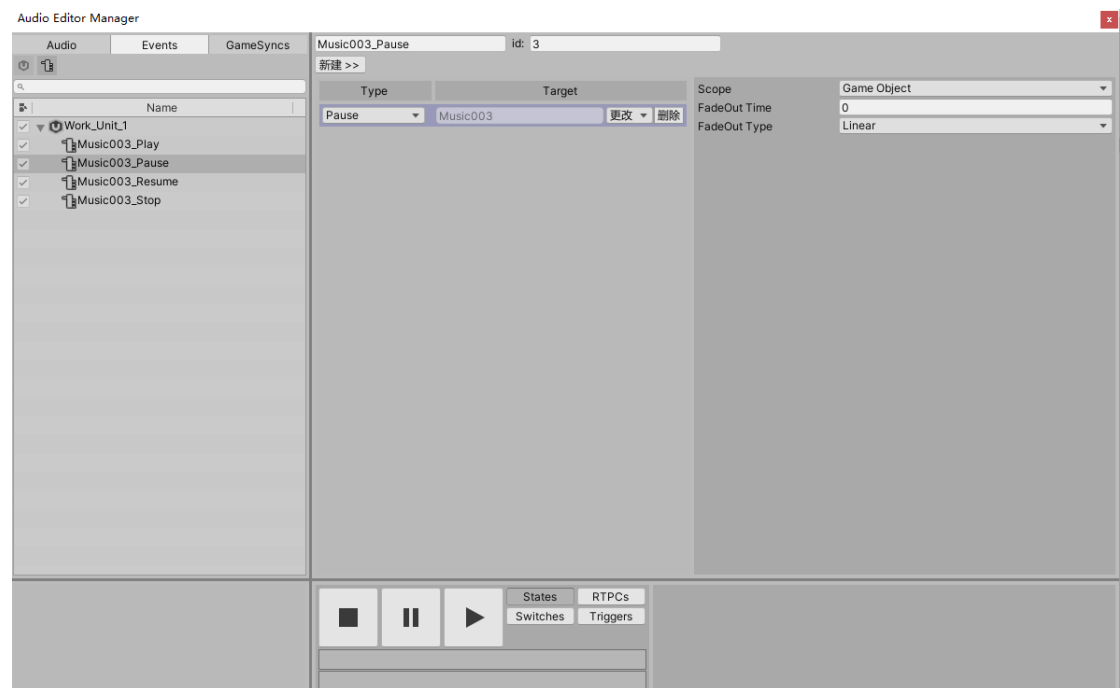


图 5 事件创建完毕后的工程内容

就像上一节一样，我们为 **GameObject** 添加另外三个 **EventTrigger** 组件，并设置对应的事件。依次点击 **Play**、**Pause**、**Resume**、**Stop** 事件对应的 **EventTrigger** 的“触发事件”按钮，查看效果。

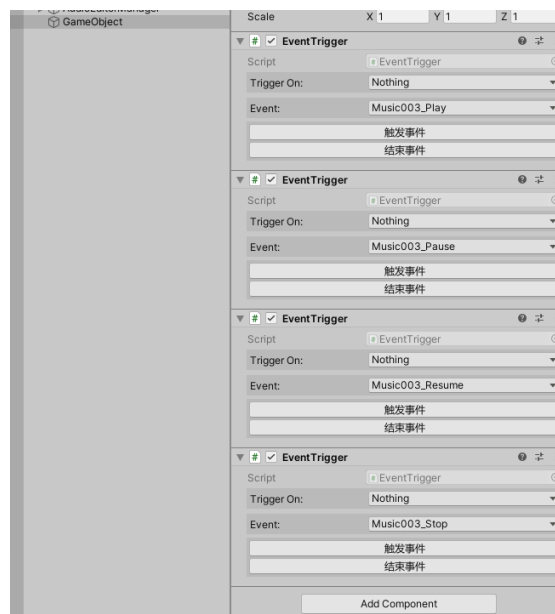


图 6 EventTrigger 的设置

2.3 为停止事件添加淡出效果

有时候我们会觉得直接暂停或者停止音乐有突兀感，希望能以缓和的方式停止音乐，那么我们可以在事件中进行设置，让事件触发后淡出一段时间。

打开编辑器窗口，切换到 Events 标签页，选择之前创建的“Music003_Stop”事件，可以看到 Stop 动作右侧有相关属性可以设置。将 FadeOut Time 选项设置为 1（秒），FadeOut Type 选择淡出曲线，默认为线性淡出。每种动作类型都有不同的属性设置，详情请参见章节《主界面介绍》中的 Event 模块。

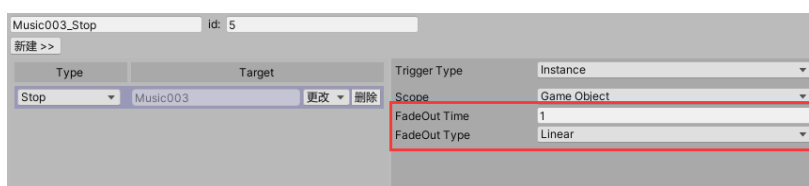


图 7 Stop 事件的淡出设置

现在重新依次触发 Play 和 Stop 事件，感受效果。

2.4 为音乐添加属性效果

AudioEditor 比 Unity 原生的 AudioSource 强大之处在于，它不仅保留了 AudioSource 的基本选项，还具有各类游戏设计中常见的音频功能。

在编辑器中切换到 Audio 标签页，选择“Music003”组件，可以在右侧属性窗口看到各种属性设置。你可以边调节属性边使用预览功能来达成满意的效果。

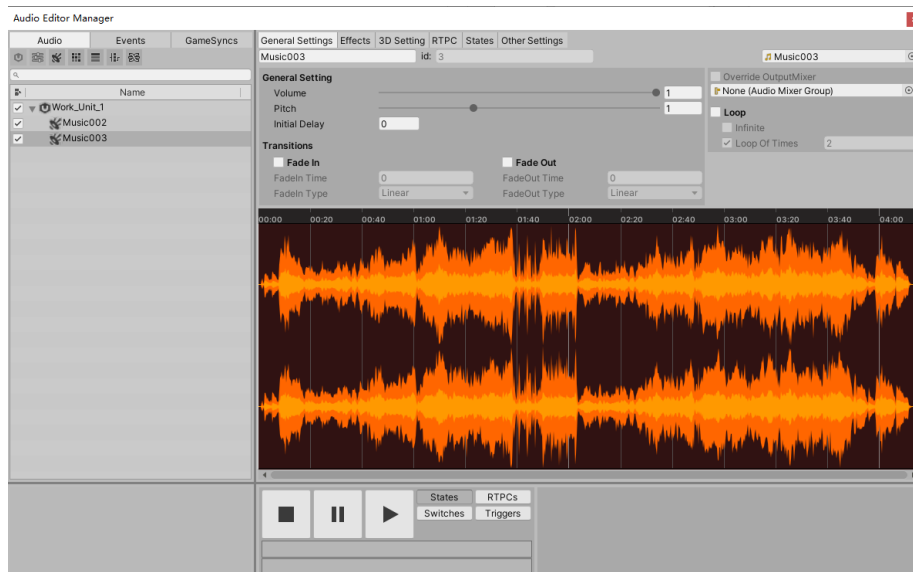


图 8 SoundSFX 组件的界面

例如你觉得音乐的音量有些过于嘈杂了，可以适当降低 **Volume** 属性，以及音乐应该是无限循环播放，则勾选 **Loop** 选项并且选择 **Infinite** 子选项。

或者你想让子弹声每次都带有些许音高变化，可以调整 **Pitch** 属性，或者在 **Pitch** 属性上右键，选择 **Random** 模式，这会在每次播放时在 **Random** 区间内随机取值播放（仅 **Volume** 和 **Pitch** 属性有此功能）。

或者音效应该是区域性的 3D 音效，可以在 **3D Setting** 标签内进行 3D 设置。

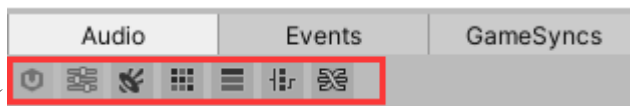
又或者想通过 UI 的滑动条来调整音乐的音量，可以在 **RTPC** 标签内设置对应的 **GameParameter** 进行映射。

详细的属性设置可以参见《主界面介绍》章节。

2.5 使用容器对复数音频进行有机组织

现在我们已经了解了如何对单个音频进行编辑并且控制其播放流程。这对于播放单个背景音乐或者音效是足够的，可是如果我们想更丰富游戏中的音效环境，例如将好几首音乐轮换作为背景音乐，又或者进入战斗状态后切换为战斗音乐而退出后切回原先的背景音乐，这就需要用到容器。

容器可以将复数音频有机结合起来实现整体功能。查看编辑器的 **Audio** 标签



页下的组件图标列表（），这里涵盖了所有容器类型，包括随机容器、顺序容器、选择容器等，容器的具体内容参见《主界面介绍》章节。

现在我们来实现让几首背景音乐随机轮换播放的功能，这就要使用到随机容器，在左侧列表图中选择 **WorkUnit_1**，然后点击组件列表中的 **RandomContainer** 图标（如果你不清楚哪个是 **RandomContainer** 的话，请在图标上悬停一会儿，它会出现注释提示），这会在 **WorkUnit_1** 层级下生成一个 **RandomContainer**，我们不妨将其改名为“**BGM**”，这样我们生成了一个 **RandomContainer** 容器。

光有容器还不够，目前容器的内容还是空的，我们需要为容器添加相应的子组件，这样容器才能对其进行操作。选择我们之前导入的“**Music002**”和“**Music003**”组件，然后一同拖入容器中。

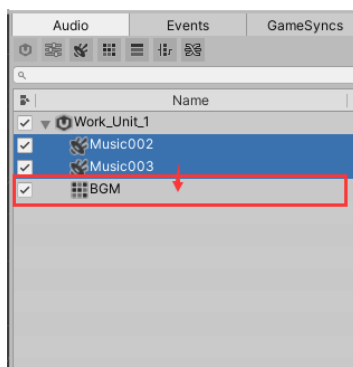


图9 将两个 SoundSFX 组件拖入容器中

这样容器的子组件就设置好了，容器会根据自身的设置对子组件进行相关的逻辑操作，接下来我们来对容器的属性进行设置。

首先我们将 **PlayType** 设置为 **Shuffle**，这样在所有的子组件轮换完之前不重复已播放的组件。我们也希望容器能连续地播放完一组音乐，所以 **PlayMode** 中选择 **Continuous**。最后我们希望容器能够无限循环直到我们触发停止事件，所以将 **Loop** 选项勾选上，并且勾选 **Infinite** 子选项，注意这里我们还需要将子组件中的 **Loop** 选项取消勾选，否则当容器随机播放到此子组件的时候会根据子组件的设置无限循环此子组件，这样就和我们想要的功能相违背了。现在我们已经设置好了容器的属性了，你可以先在预览窗口对容器预览播放查看是否符合我们想要的功能。

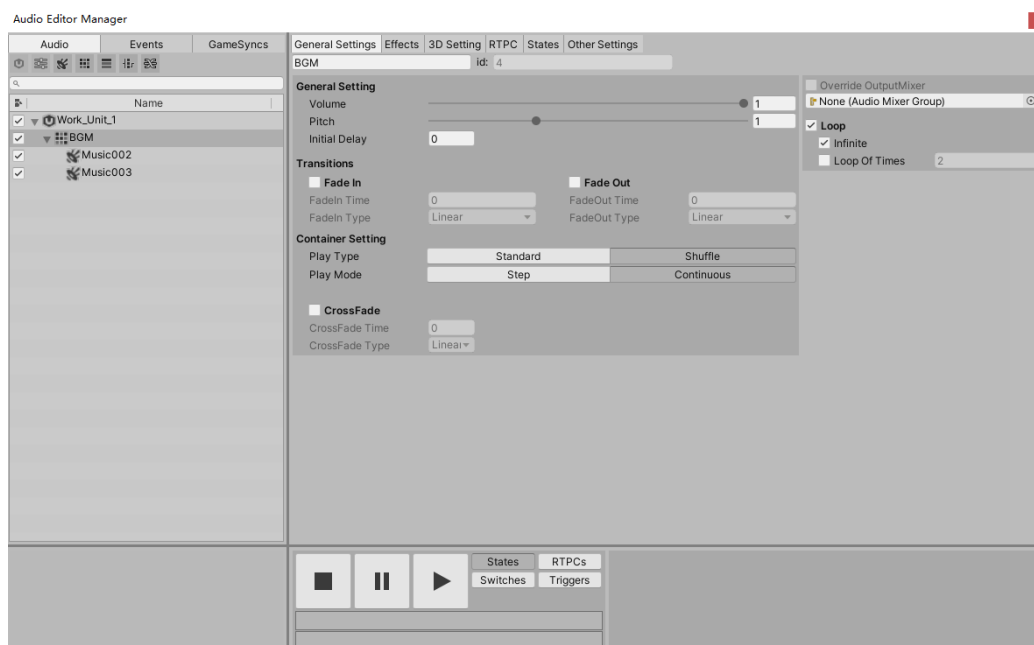


图10 将 RandomContainer 设置好后的界面

现在我们依然要为生成的容器创建事件，但我们也可以换种方式，利用原来

已经生成的事件。由于在功能上我们是用随机播放的随机容器组去替换单一播放的背景音乐，所以之前创建的关于“Music003”的相关事件可以转换复用。

切换到 Events 标签页，将之前创建的四个事件中的名称中的前缀“Music003”修改为“BGM”，以更好地匹配其相关功能。然后在所有事件的动作队列中，将 Target 从“Music003”修改为我们生成的容器“BGM”。

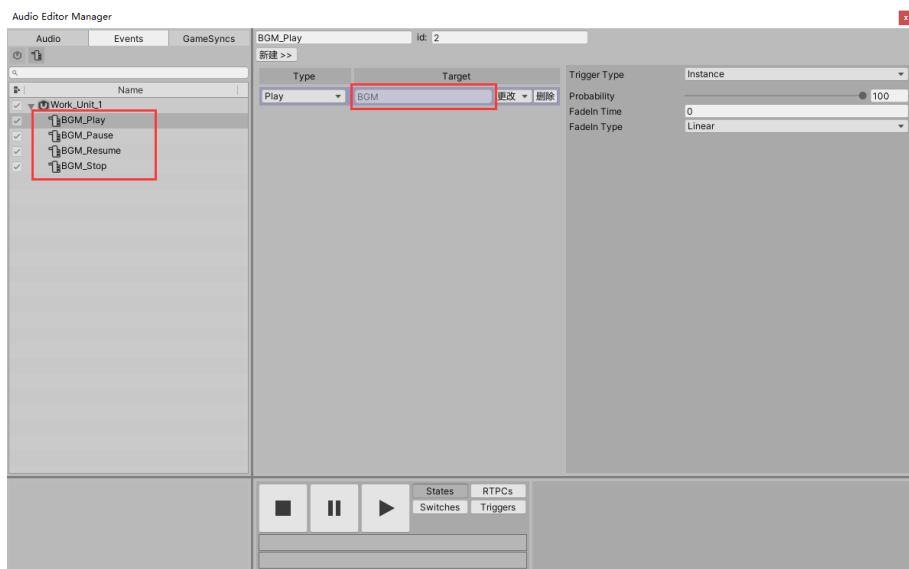


图 11 将 Event 修改设置好后的界面

而在触发端，我们不需要进行修改，因为 EventTrigger 所链接的是事件的 ID，我们仅对事件内容进行了修改而不是更换了事件，这样可以很方便地在不涉及程序端的情况修改我们想要的逻辑功能，这就是组件功能和事件功能分离的好处。现在查看一下之前创建的 EventTrigger 组件，应该已经显示修改后的事件名，试着触发事件感受修改后的效果。

第三章 主界面介绍

编辑器界面可以分为 4 个部分，分别为 Project Explorer Window（项目浏览器窗口）、Property Editor Window（属性编辑器窗口）、Help Window（说明窗口）、Preview Window（预览窗口）。

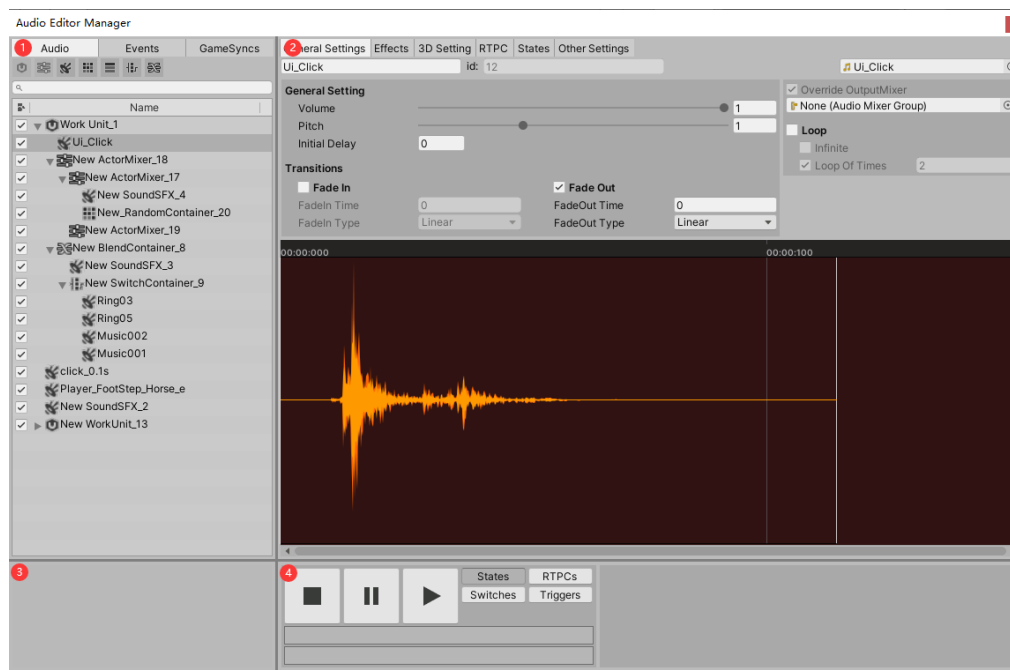


图 1 编辑器主界面

2.1 Project Explorer Window

项目浏览器窗口主要用于创建、删除和管理各类组件，通过树状列表可编辑组件之间的交互。



图 2 Project Explorer Window

- 1) 此标签分为 Audio、Event、GameSync 三个大类，分别实现不同的功能，每一个标签页下都有不同的组件。Audio 主要管理各种音频组件，Event 主要负责各类事件功能的编辑，GameSync 主要与游戏同步相关。

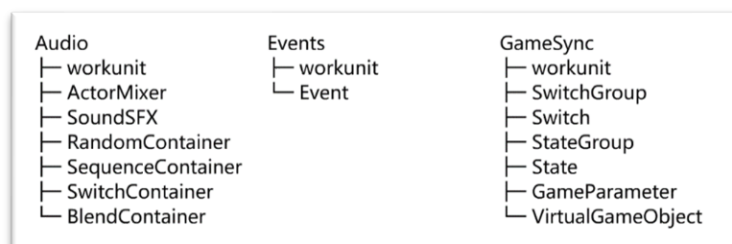


图 3 所有组件一览

- 2) 此行按钮会随着标签页改变，点击相应按钮即可在列表中当前选择的组件下创建一个子类组件。

WorkUnit 相当于文件夹，并且可以嵌套，用于对其他组件进行分类。虽然非 WorkUnit 组件可以创建在根目录下并且没有类型限制，但建议根据项目实际情况使用 WorkUnit 进行有结构的分类。多人协作时在不同 WorkUnit 下工作不会产生版本管理冲突。

SoundSFX 除了通过点击 SoundSFX 按钮进行创建之外，还提供了批量导入的功能，在 Unity 的 Project 窗口中选中若干 AudioClip，直接拖入到列表中的某一层级即可批量创建 SoundSFX，SoundSFX 会自动以 AudioClip 的名字命名。

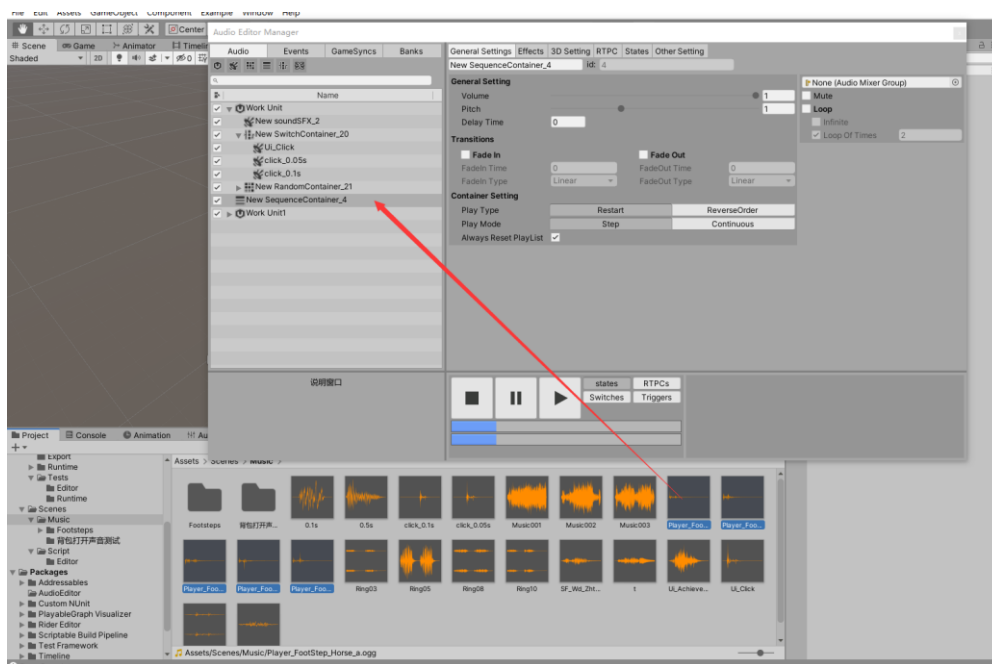


图 4 将若干 AudioClip 导入到编辑器中

- 3) 搜索框可用于快速搜索匹配对应名字的内容，不分大小写。还可通过“id:”前缀来对 id 进行搜索。
- 4) 树状列表显示了所有组件之间的结构，可以对每个组件进行拖动。容器的子类组件就是这个容器的输出内容，例如某个 RandomContainer 组件下有 3 个子 SoundSFX 组件，播放 RandomContainer 就会在 3 个 SoundSFX 中随机抽取播放。

左边的勾选框能“隐藏”某些组件，取消勾选后该组件不能进行编辑，并且不会成为其父级组件的子类，但依旧能进行事件触发来播放。

在列表中还可以通过双击左键来进行重命名，Audio 标签下可以选择一个或多个组件右键批量新建 Event 事件或者删除组件，也可以通过快捷键 Delete 来删除组件。

2.2 Property Editor Window

属性编辑窗口会显示左侧当前选中的组件的具体属性，对其进行编辑可实现不同的功能。如果不明白具体属性有何作用，可将鼠标悬停在其属性名字上，会浮现相关释义。接下来对不同组件的具体属性进行说明。

SoundSFX

SoundSFX 是最基础的组件,它能包裹一个音频文件并对其进行参数化调整,是音频输出的根本来源。

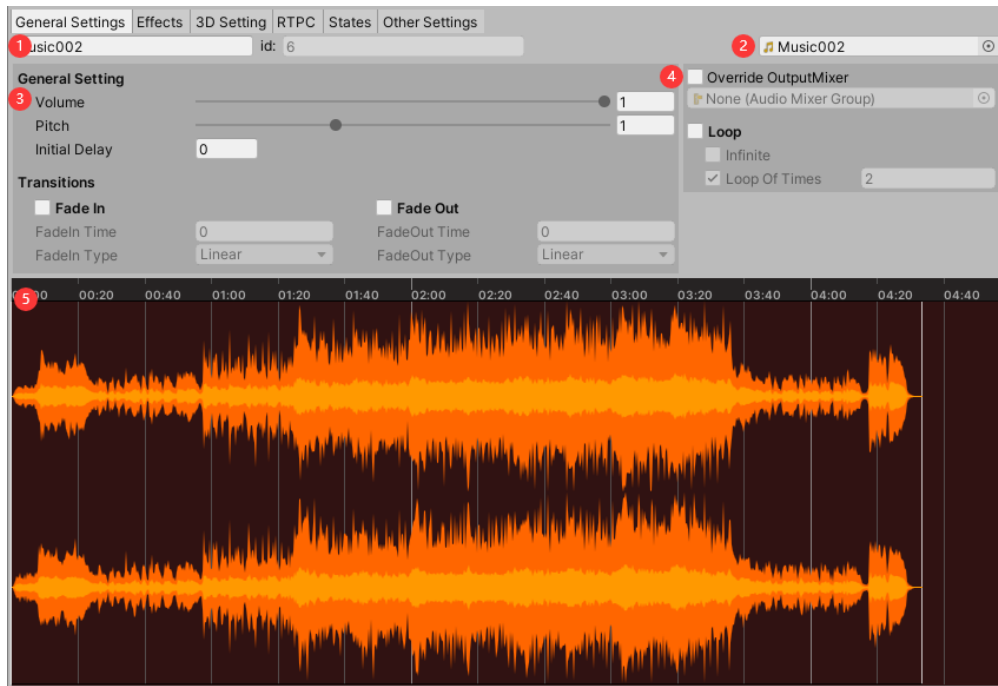


图 5 General Settings 中的设置

- 1) 组件的名字和 id, 名字可通过此栏进行修改。
- 2) SoundSFX 所链接的音频文件, 单击在文件的名字上可以在 Project 窗口中高亮显示, 点击右边的⊙按钮可更换音频文件。
- 3) 包括 Volume、Pitch、Delay, 淡入淡出等数值的设置。其中在 Volume 和 Pitch 轴上右键可以切换到随机的 Random 轴, 可进行音量上的随机输出。
- 4) AudioManagerGroup 是 Unity 本身的音频混音器, 可实现类似 Bus 的结构, 具有各种 DSP 效果以及闪避等功能。其界面可通过 Window=>Audio=>Audio Mixer 打开, 详情请见:

<https://docs.unity.cn/cn/2020.2/Manual/AudioMixerOverview.html>

请注意此选项不会在父级中生效。

- 5) 此时间轴会显示 AudioClip 的具体波形图, 以方便快速了解 clip 的波形结构。可以通过鼠标滚轮放大或缩小时间轴。右键菜单可选择“Resize”重置波形图, 或者“Show”选项选择两种不同的显示模式。快捷键 shift+鼠标左键进行拖动可以设置 Delay Time。

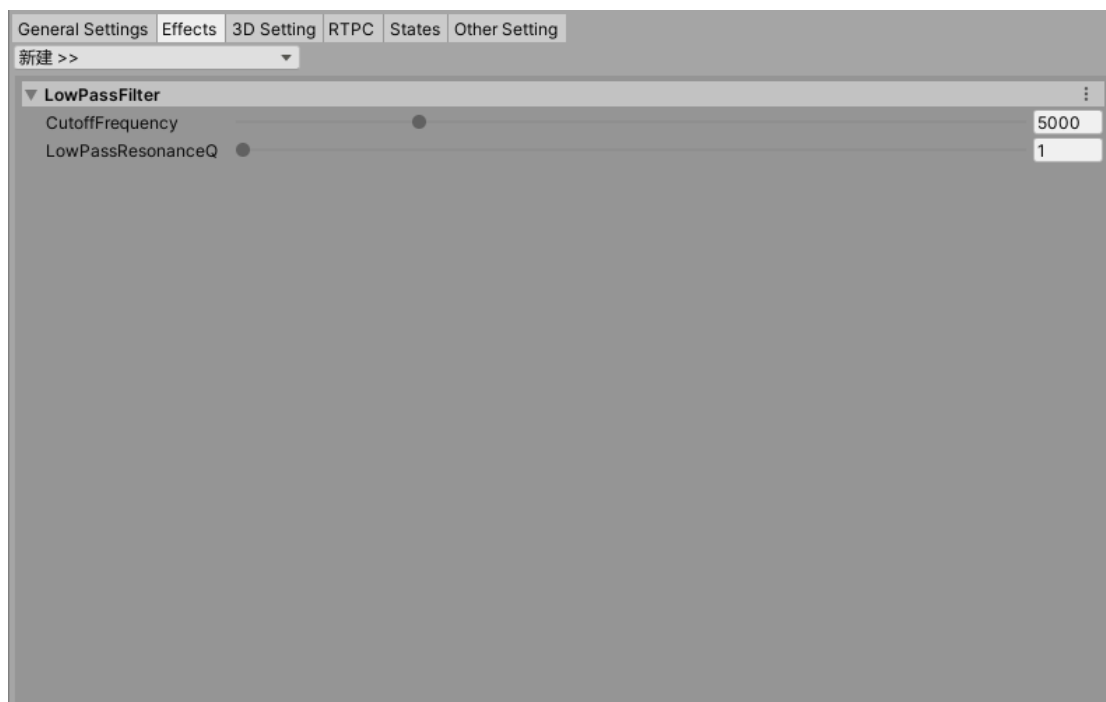


图 6 Effects 中的设置

- 1) 特效目前提供了低通滤波器 (Audio Low Pass Filter), 高通滤波器 (Audio High Pass Filter), 回声滤波器 (Audio Echo Filter), 失真滤波器 (Audio Distortion Filter), 混响滤波器 (Audio Reverb Filter) , 混响区域(Audio Reverb Zone), 合声滤波器 (Audio Chorus Filter)七种, 具体的参数设置及效果请参考:

<https://docs.unity.cn/cn/2020.2/Manual/class-AudioEffect.html>

请注意, 目前在子级中设置的特效在父级中不会生效, 例如在 randomContainer 的一个子级 SoundSFX 中设置低通滤波器, 播放 randomContainer 时随机到此 SoundSFX, 其低通滤波器不会生效。

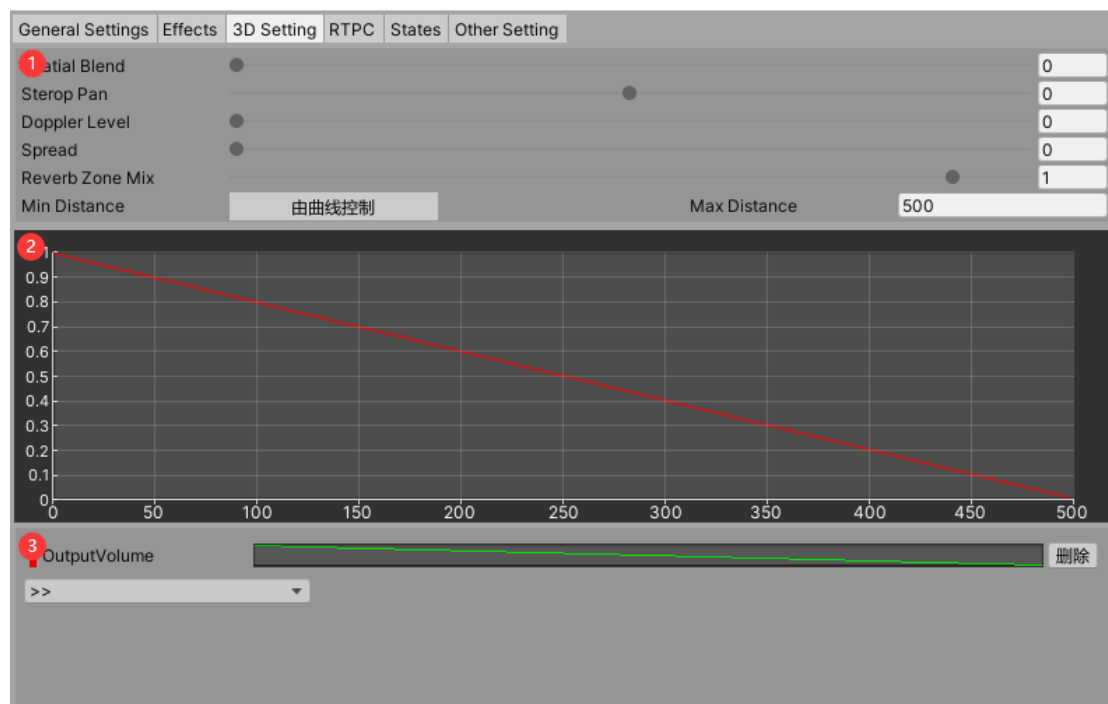


图 7 3D Setting 中的设置

1) Spatial Blend: 设置组件的 2D 和 3D 分量。

Stereo Pan: 设置左右声道的偏移。

Doppler Level: 设置多普勒效应的程度。

Spread: 在发声空间中将扩散角度设置为 3D 立体声或多声道。

Reverb Zone Mix: 设置输出到混响区的信号量。

Distance: 设置组件的影响范围。可以在任意 GameObject 上添加 AudioSource 组件, 调节其中的 Min Distance 和 Max Distance 来查看在游戏场景中的实际范围大小。

2) 此图表显示了其下边所有属性随距离衰减的曲线情况。

3) 可以在此设置一些属性关于距离的变化曲线, 支持的属性有: Volume、Spatial Blend、Spread、Low Pass Filter、Reverb Zone Mix、Priority。点击右侧带有绿线的框会打开曲线设置窗口, 可以设置具体的曲线。

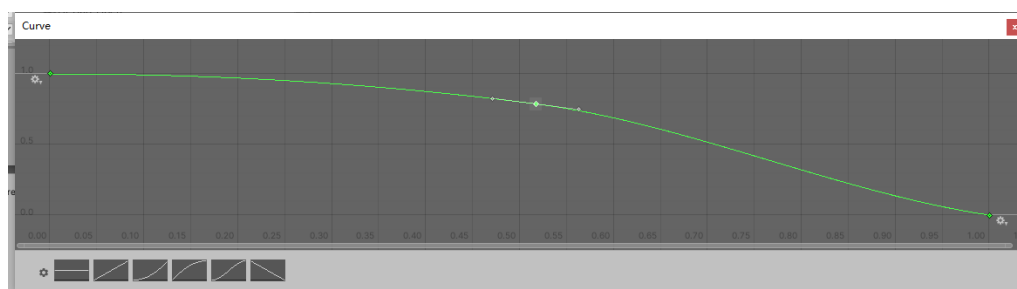


图 8 曲线设置窗口

在曲线设置窗口中，双击左键或快捷键 **Delete** 可以快速新建和删除节点，在节点上右键可以编辑节点的具体数值，以及其两个杠杆点(Tangent)的有无和协作方式。窗口下方可以进行预设的管理。

请注意，目前不要将曲线设置出 y 轴的[0,1]区间外，否则可能会产生不可预测的结果。

4) 3D 设置的更详细说明，请参考：

<https://docs.unity.cn/cn/2020.2/Manual/class-AudioSource.html>

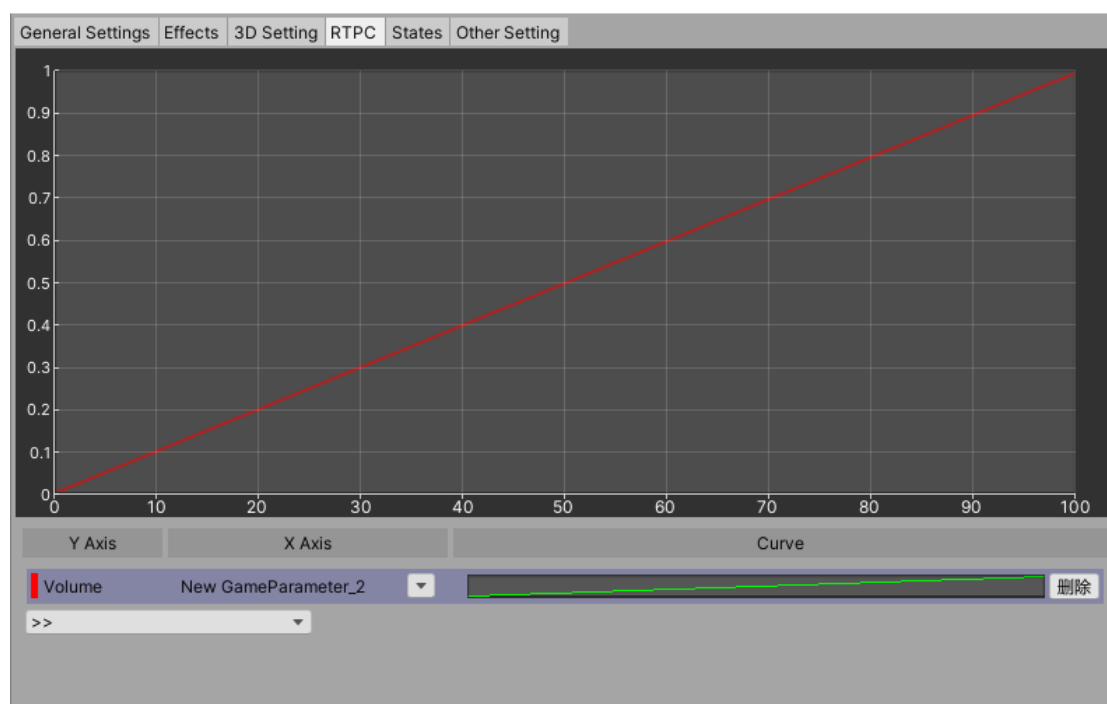


图 9 RTPC 中的设置

1) 想要使用 RTPC 功能请先在 GameSync 中创建 GameParameter。Y 轴可选择属性有 Volume、Pitch、Priority，其曲线的设置方式与 3D Setting 相同；

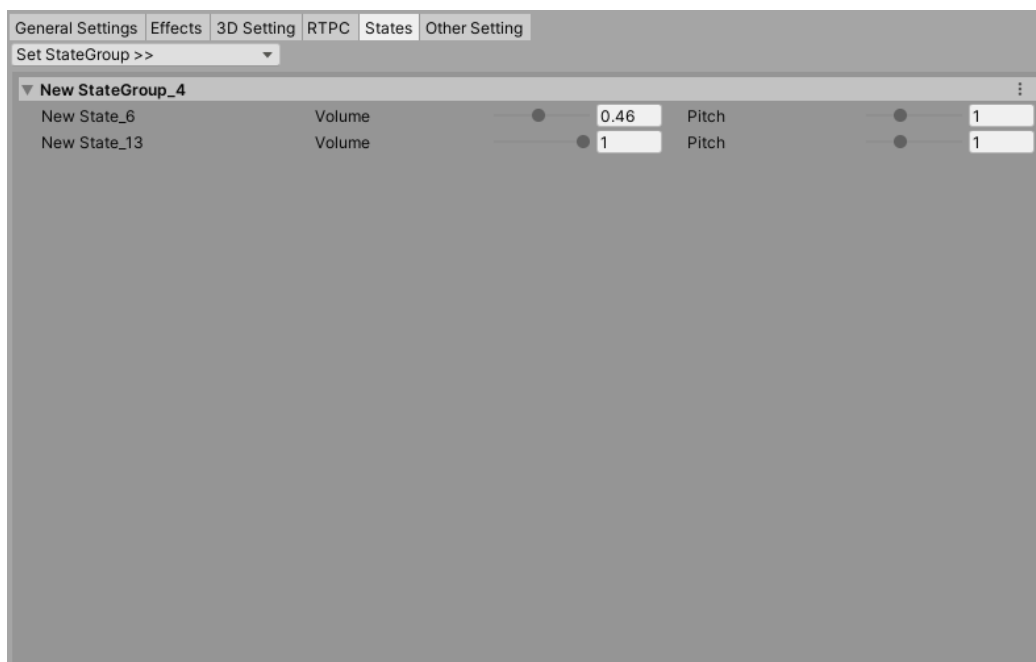


图 10 States 中的设置

- 1) 想要使用 State 功能请先在 GameSync 中创建 StateGroup。选择左上角的“Set StateGroup”选项可以在此组件中链接一个 StateGroup，可以设置在不同 State 下的属性，目前只开放 Volume 和 Pitch 两种属性。

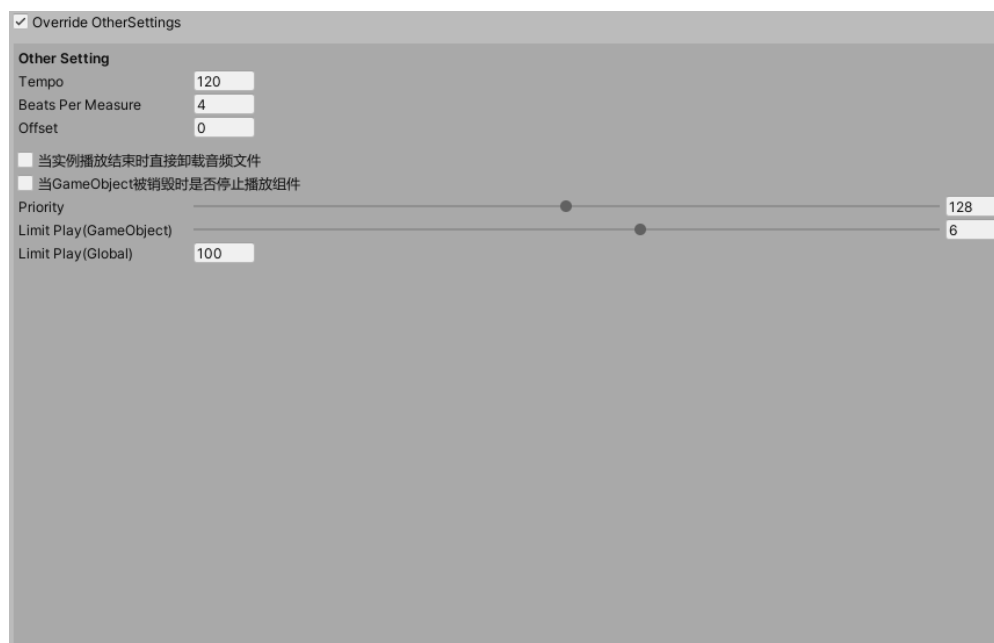


图 11 OtherSettings 中的设置

- 1) **Other Setting:** 此框中的内容为当前组件的其他设置，包括：
 - 设置 BPM 以及节拍信息，用于多音频实例间的节拍对齐功能。
 - 实例播放结束时是否直接从内存中卸载相应的音频文件。

- 触发实例的 **GameObject** 被销毁时是否直接停止播放当前组件的实例。
- 优先级（数值越小优先级越高）。
- 限制此组件的最大同时播放数量，分为 **GameObject** 和 **Global**。

ActorMixer

ActorMixer 的界面与 **SoundSFX** 十分类似，其功能是为其下的子组件提供统一化的功能设置。子组件可以通过 **Override** 选项进行特异性设置。

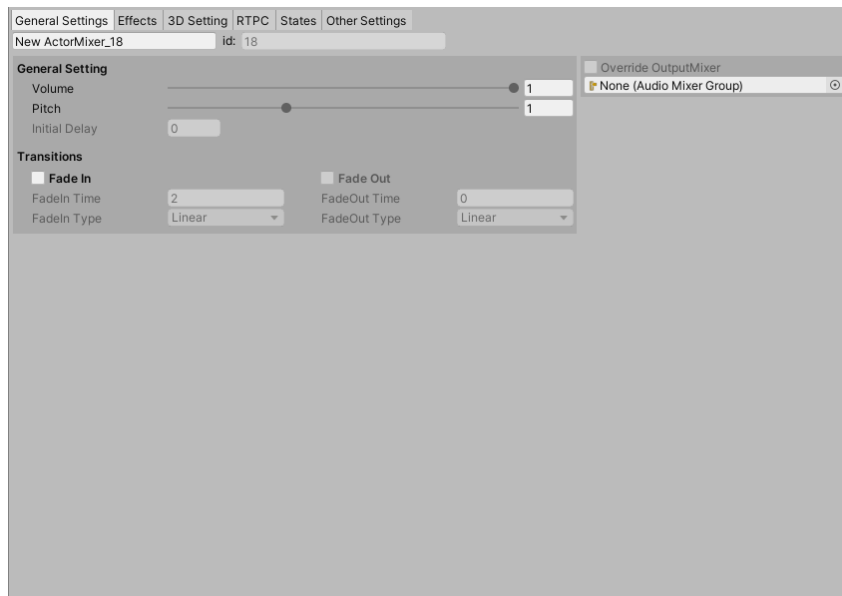


图 12 ActorMixer 的界面

Random Container

Container 用于对子组件按照功能进行对应逻辑播放，其大多数的属性设置与上述 **SoundSFX** 相同，现只说明其不同的部分。

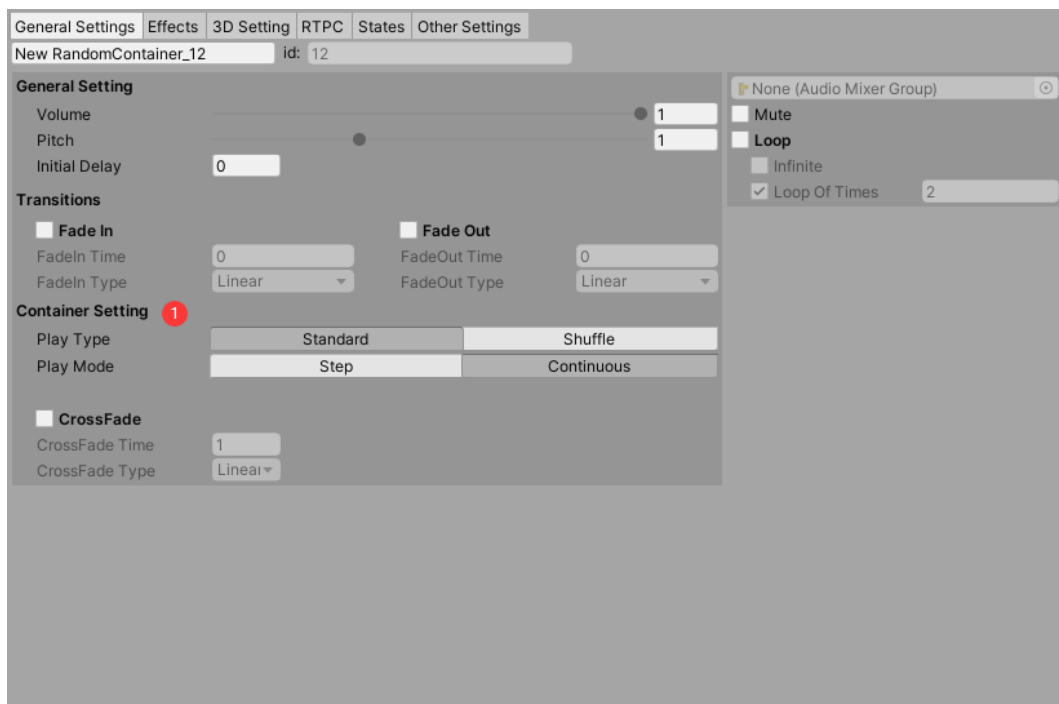


图 13 RandomContainer 的 ContainerSetting

1) RandomContainer 具有专属的 Container Setting 选项，包括：

- **Play Type:** Standard 为标准乱序模式。Shuffle 为洗牌模式，在播放完全部子组件前任意子组件不会重复播放 2 次。
- **Play Mode:** Step 为步进播放，每次 play 都只播放 1 个子组件。Continuous 为连续播放，每次 play 会根据 Play Type 播放完所有子组件。
- **Cross Fade:** 可设置时间和类型，类型分为延迟和交叉淡变两种，仅在 PlayType 设置为 Continuous 时生效。交叉淡变在子组件只有一首和每次循环的结尾这两种情况下不会生效。

在 Loop 循环下搭配 Container Setting 可以创造非常丰富的随机循环效果。假设在 Loop 中设置循环 n 次，Step 模式相当于从容器中抽取 n 个子组件进行乱序播放（选择 Shuffle 选项可以确保不会连续播放相同的子组件）。Continuous 模式会对整个容器进行循环（Shuffle 选项可以确保两次循环前后的首尾不会相同）。

Sequence Container

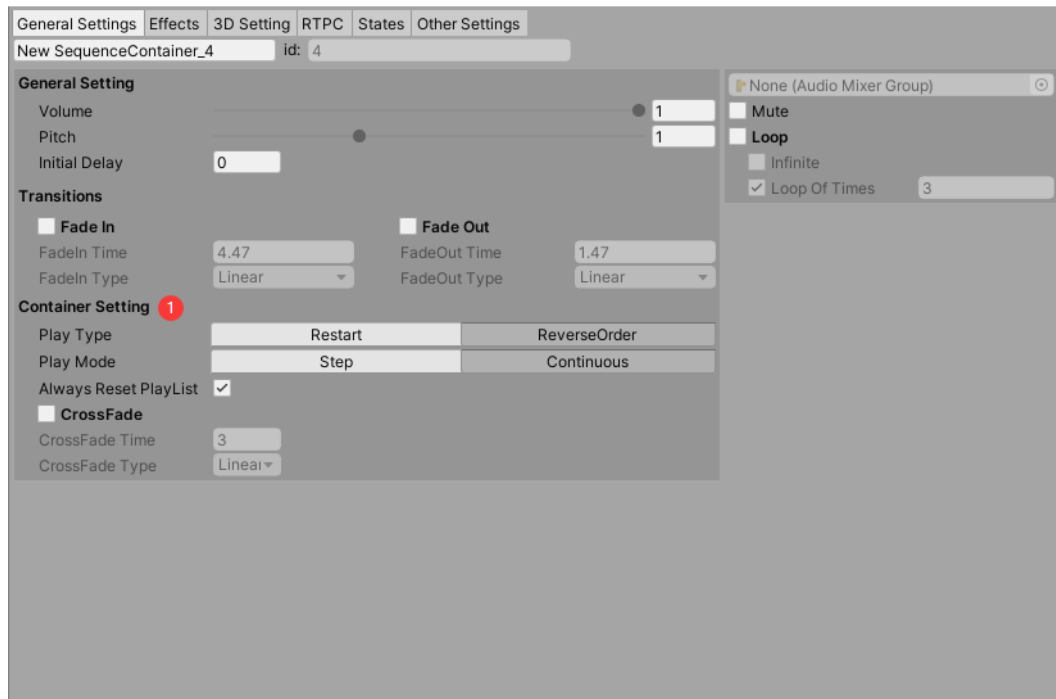


图 14 SequenceContainer 的 ContainerSetting

1) SequenceContainer 的 ContainerSetting 有 3 个选项，包括：

- Play Type: 当进行循环计数相关时，Restart 为重头开始，Reverse 为逆序开始。
- Play Mode: Step 为步进式播放，Continuous 为连续播放。
- Always Reset PlayList: 仅 continuous 时有效，勾选后每次 Play 都将从首尾开始播放。
- CrossFade: 功能同 RandomContainer。

假设容器中有 3 个子类，对容器循环播放 3 次，当勾选 Always Reset PlayList 时，其播放顺序为 123、123、123（Restart）或 123、321、123（Reverse），取消勾选时，其顺序为 123、231、312（restart）或 123、212、321（Reverse）。

Switch Container

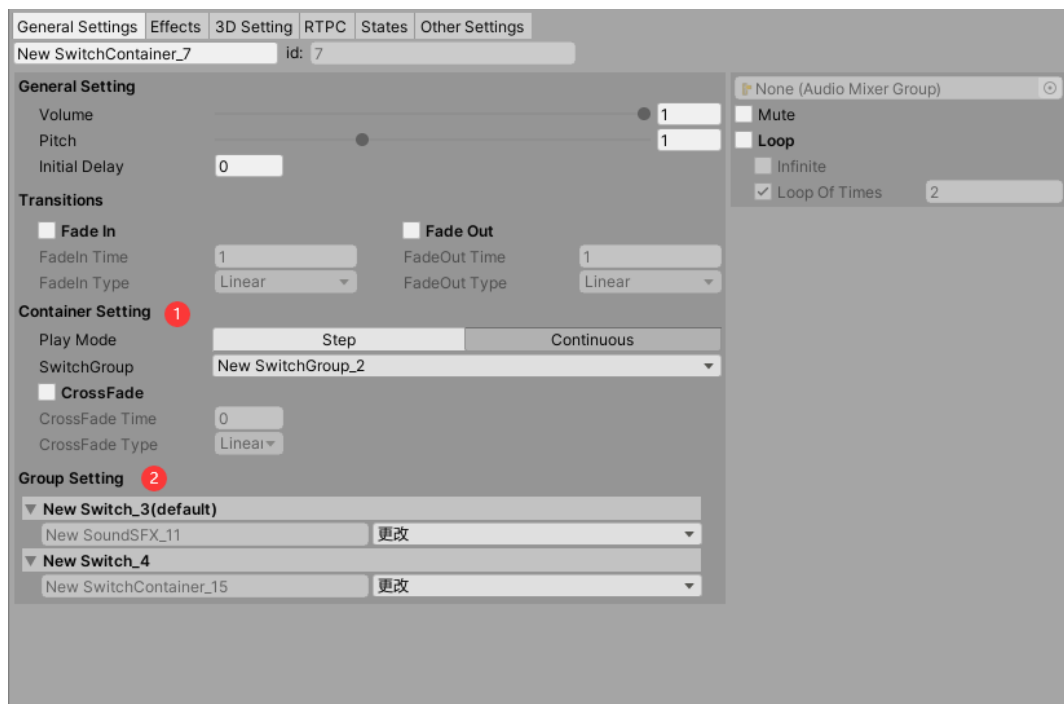


图 15 SwitchContainer 的 ContainerSetting

1) 想要实现 SwitchContainer 的功能，请先在 GameSync 中创建 SwitchGroup。

SwitchContainer 的设置包括以下 2 项：

- **Play Mode:** Step 模式下播放结束后会即时销毁实例，如需再次播放则需重新触发 Play 事件。Continuous 模式下播放结束后不会进行销毁，需触发 Stop 事件才会进行销毁，每当 Switch 变更时，会直接播放对应的子组件。播放的次数由 Loop 决定。
- **SwitchGroup:** 在此可选择已设定好的 SwitchGroup。
- **CrossFade:** 仅在 Continuous 时可用，在 Switch 切换时生效。

2) 在 Group Setting 中会显示相关 SwitchGroup 下所有的 Switch 选项，可以设置相关 Switch 所对应的子组件。当 SwitchGroup 切换到某个 Switch 时，相关的 SwitchContainer 将播放对应的子组件。在 Switch 标签上右键可以设置容器默认播放的子组件，右侧会以（default）显示。

Blend Container

BlendContainer 可以将复数的子组件同时进行播放，并通过游戏参数 GameParameter 来控制每个部分的输出权重。

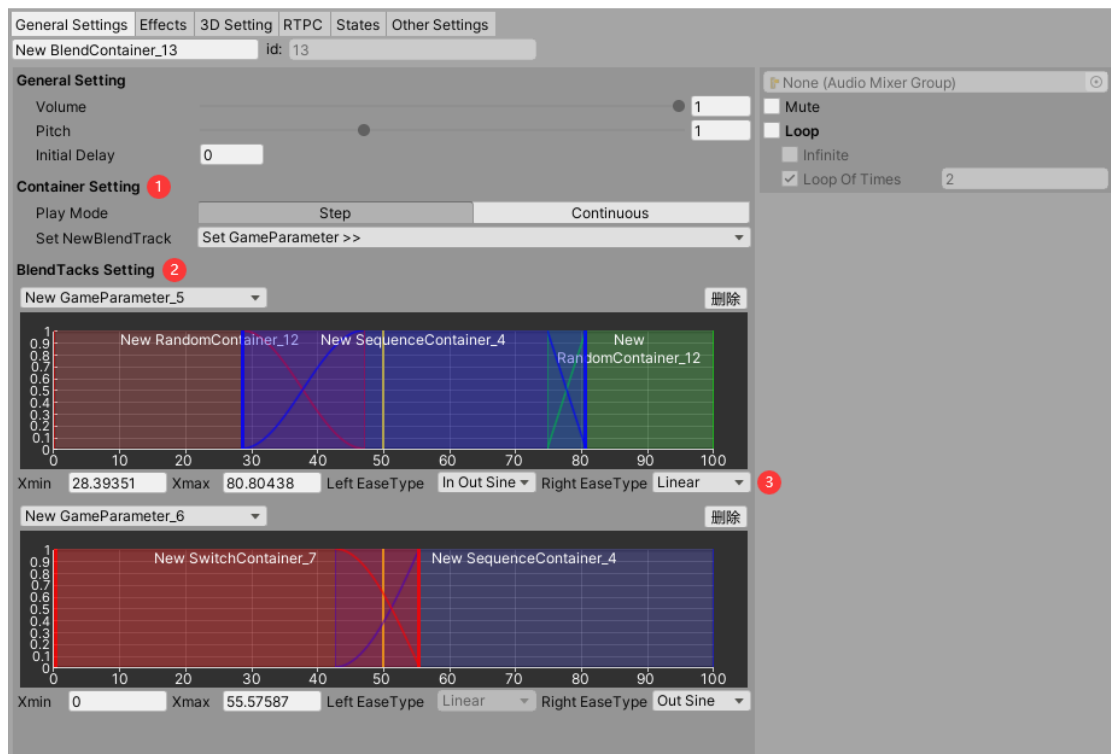


图 16 BlendContainer 的 ContainerSetting

1) BlendContainer 的专属选项包括以下：

- **PlayMode**: Step 模式下 BlendContainer 会同时播放所有子组件，且 GameParameter 的数值仅控制各部分的输出分量，所有子组件播放结束后销毁。Continuous 模式下当 GameParameter 进入一个新的区域时，对应的子组件会重新开始播放，确保能获得即时的音频响应，组件会持续监听 GameParameter 的变化，不会自动销毁，需主动调用 Stop 事件。
- **Set GameParameter**: 可以在此选择一个 GameParameter 来创建一个 BlendTrack，其内容会在 BlendTracks Setting 中显示。

2) 每一个 BlendTrack 包含了目标 GameParameter 以及一个用于显示各部分子组件区域的图表。在图表的空白处右键会显示 BlendContainer 的子组件菜单，选择其中一项即可创建相应的子组件区域，该区域以颜色进行区分，选中后会高亮显示，并且可以划动左右边界来达到调整边界范围的目的。黄色的竖线代表 GameParameter 当前的数值。

3) 当选中某一个子组件区域时，会显示其相关信息，包括左边界和右边界的数值，以及左右的交叉过渡类型。

Event

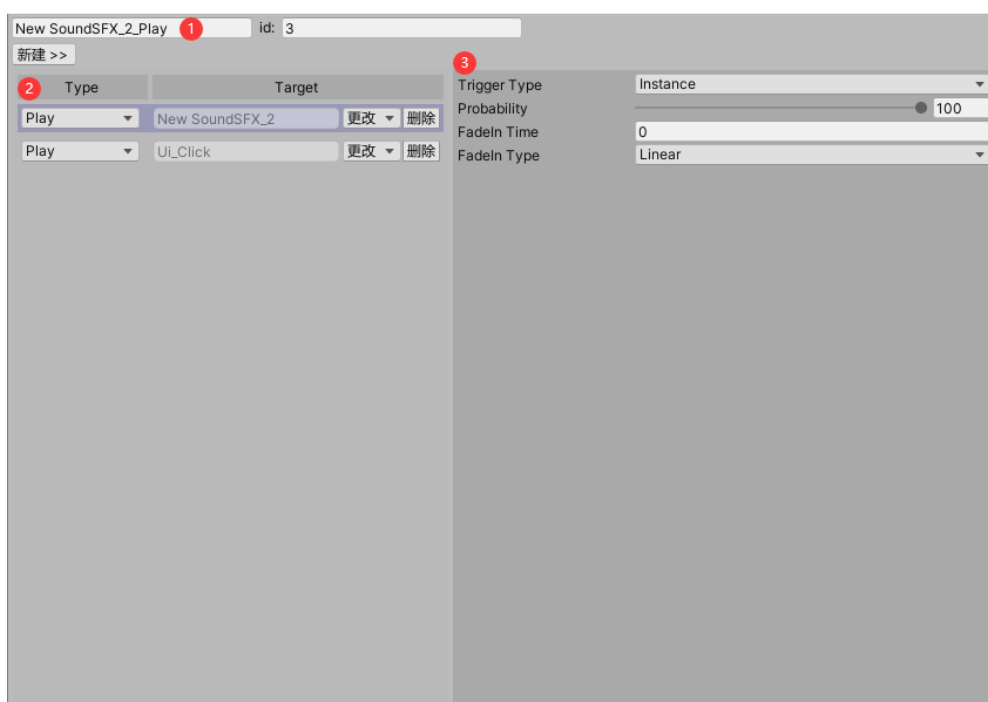


图 17 Event 的设置

- 1) Event 的 Id 可以进行修改，但必须是大于 0 的正整数并且与其他 Event 的 Id 都不相同。这是为了避免在通过填表方式进行事件触发的项目中，某个 Event 事件的误删或者需要修改等情况时，可以直接修改 id 以对应原来的表，而不再需要找策划修改触发表。
- 2) 每一个 Event 事件下都可以包含若干个 Action（动作），通过“新建>>”按钮即可创建一个动作，动作的类型有 Play, Pause, PauseAll, Resume, ResumeAll, Stop, StopAll, SetSwitch, SetState 几种。可在“更改”按钮下选择该动作的作用对象，在弹出来的选择窗口中双击某一组件或者单击“确定”按钮可以选择某个对象。可以单击“删除”按钮或者在鼠标选中某个动作时使用 Delete 快捷键删除动作。
- 3) 当选择不同的动作时，此框会显示当前动作的详细设置，目前有 TriggerType（触发类型）、Probability（触发概率）、Scope（作用范围）、FadeIn Out（淡入淡出）等功能。
 - Play: 生成一个对象组件的实例，进行播放。
 - Pause: 对对象组件的实例进行暂停。Scope 为 GameObject 时会暂停相同组件 Id 且同 GameObject 下的相同实例，设置为 Global 时会暂停所有相

同组件 Id 的实例。

- **PauseAll:** Scope 为 GameObject 时会暂停触发事件的 GameObject 下的所有实例，为 Global 时会暂停所有的实例。
- **Resume:** 对对象组件的实例进行恢复播放。
- **ResumeAll:** 在作用范围内恢复组件播放，Scope 设置同 PauseAll。
- **Stop:** 停止播放某个组件的实例，并进行销毁。Scope 设置同 Pause。
- **StopAll:** 在作用范围内停止并销毁所有实例。Scope 设置同 PauseAll。
- **SetSwitch:** 设置对象 SwitchGroup 切换到对应 Switch。
- **SetState:** 设置对象 StateGroup 切换到对应 State。

Switch Group

SwtichGroup 用于创建和管理 Swtich 结构，以实现 SwitchContainer 的功能。

State Group

SwtichGroup 用于创建和管理 State 结构，并且可以设置不同 State 之间的过渡转换。

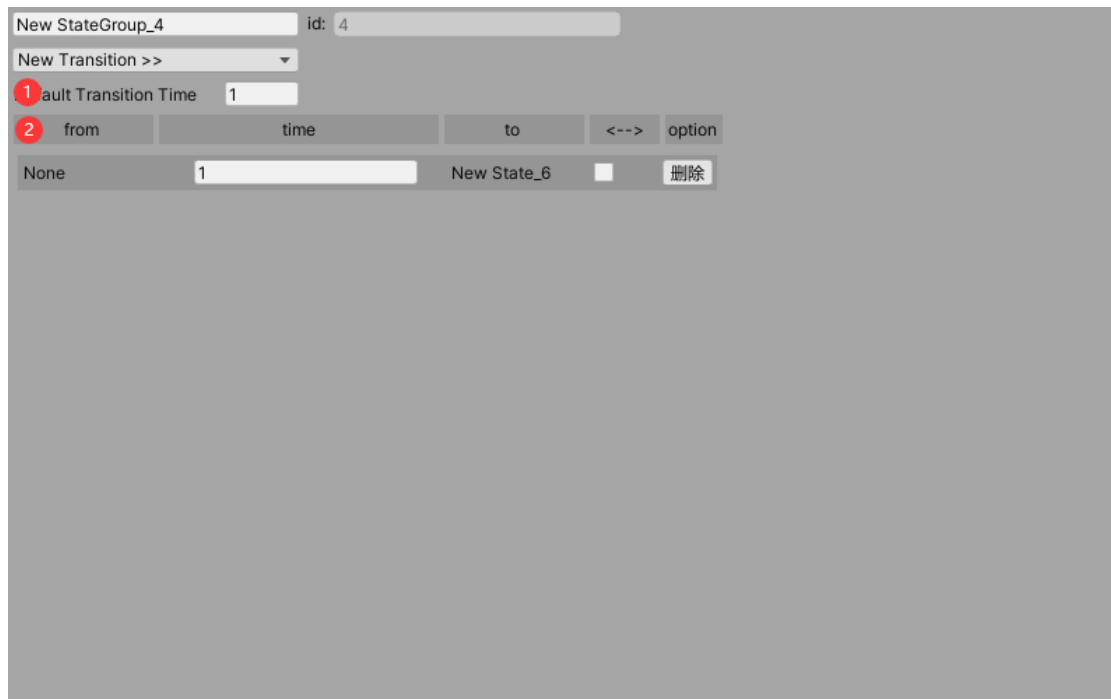


图 18 StateGroup 中的设置

- 1) **Default Transition Time:** 默认的过渡时间，以秒为单位。

- 2) 单击“New Transition>>”按钮可创建一个从某个状态至另一个状态的过渡设置，中间的时间即为过渡时间，右侧的勾选框为是否为可逆操作。点击 **Option** 中的“删除”即可删除此设置。

Game Parameter

GameParameter 用于链接游戏中的各种数据，通过 GameParameter 控制组件的属性，以完成 RTPC（Real Time Parameter Control）的功能。

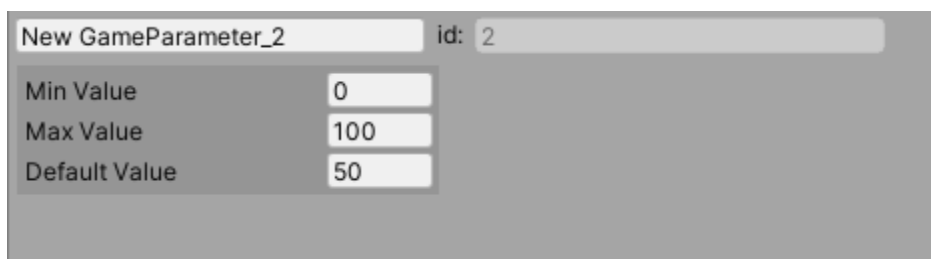


图 19 GameParameter 中的设置

- 1) GameParameter 可以设置最小值和最大值区间，以及初始化时默认的数字。

2.3 Help Window

帮助窗口会在用户进行操作时显示相关内容的说明，以帮助用户更好地了解相应功能。未完成。

2.4 Preview Window

预览窗口可以对音频组件进行播放预览（仅当项目浏览器窗口标签选择为 **Audio** 且在列表中选中的具体的 **Audio** 组件时），并且可以快速的进行相关的选项测试。在播放预览时切换 **Audio** 组件会直接停止播放当前正在预览的实例。



图 20 Preview Window

- 1) 预览提供对组件的 Play、Pause、Resume、Stop 功能，多次按下 Play 按钮可生成多个实例进行播放。pause 将暂停所有的实例，在 Pause 状态下点击 Play

按钮即可恢复播放。Stop 按钮将停止并销毁所有的预览实例。

- 2) 可在此设置组件的相关选项，包括切换不同的 State、Switch、RTPC、Trigger（未完成）的功能。
- 3) 显示总音频输出量谱。

2.5 其他说明

使用 Override 功能

AudioEditor 扩展了 Override 功能，在 AudioComponent 组件的若干标签页中可以看到此选项。具有此选项的页面数据在父级组件中时不可编辑且不会生效，例如 A 是 B 的子组件，当播放 A 组件时，优先使用 A 的 otherSetting 数据而非 B 的数据，即使此时正在播放 B 组件的音频。当勾选 Override 选项时，页面数据变为可编辑，当播放 B 组件时将读取 B 自身的数据，但依旧遵循上述规则。

第四章 EventLog Window

在 Unity 的菜单中选择 Window=>Audio Editor=>EventLog 可打开 EventLog 窗口，分为 EventLog 和 Instance 两个页面。



图 1 EventLog 窗口

EventLog 页面用于记录和显示内部事件的触发情况。点击左上角的 Clear 按钮可以清空内容（在每次 Unity 进入 Play Mode 时也会自动清空内容）。右上的搜索框可以进行快速搜索寻找想要的内容。

一次完整的事件触发会分成好几行表明其进程。一般由各种途径触发事件后，便会执行事件下的所有动作，若为 Play，则会生成一个对象组件的实例进行播放。当实例播放结束符合销毁条件或者执行相应的 Stop 动作等，则会提示销毁实例。点击对应对象组件可进行跳转查看，点击事件行会高亮相关联的动作。

Instance 页面用于显示当前编辑器系统的各项数据内容，包括 AudioComponent、Parameter、Switch 的信息等。

第五章 ProjectSettings 设置

AudioEditor 还提供了一些关于项目的其他设置。在 Unity 菜单 Editor=>Project Settings=>AudioEditor 中。

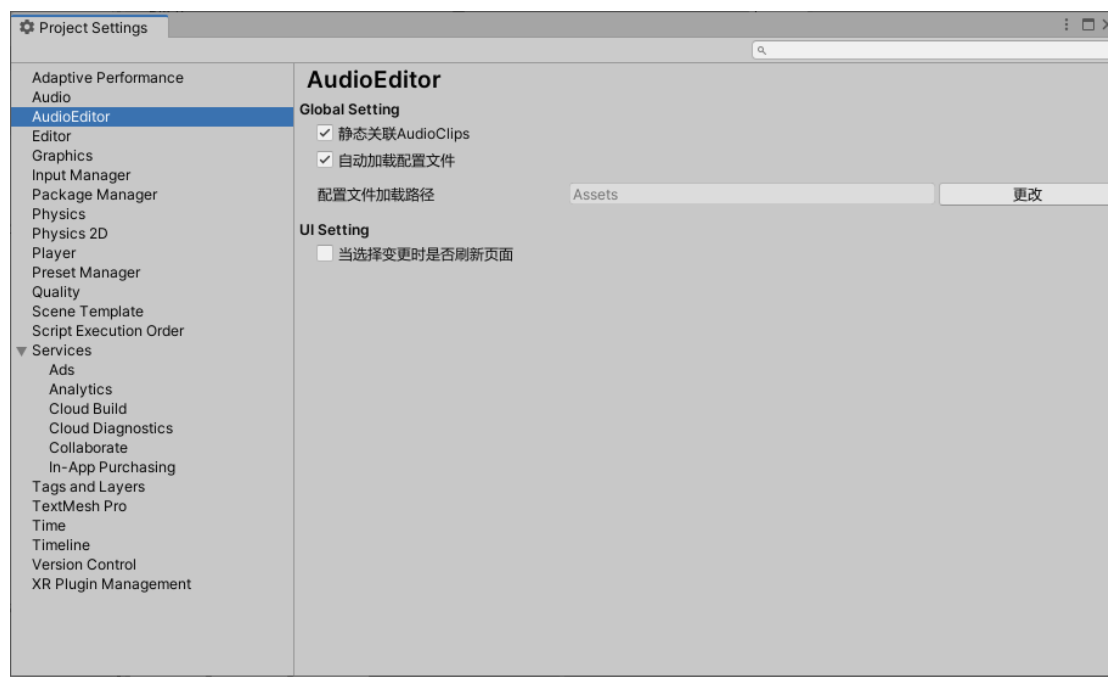


图 1 ProjectSetting 中的选项设置

- 1) Global Setting: 此框中的内容为系统全局设置，与单个组件无关，包括：
 - 是否静态关联音频文件，如果因热更新等需求取消勾选此选项，请实现 `AudioEditorManager.LoadAudioClipFunc` 方法，详情请见脚本 API 章节。
 - `RealVoicesNumber`，游戏中可同时听到的最大声音数。
 - 储存相关配置文件的加载路径。
- 2) UI Setting: 此框中的内容为编辑器主界面的选项设置，与单个组件无关，包括：
 - 当左侧列表的组件选择变更时，是否刷新属性窗口标签至 `General Settings`。

第六章 如何触发事件

4.1 通过 EventTrigger 触发

在场景中选择任意想要触发事件的 GameObject，在其 Inspector 面板中点击 Add Component=>AudioEditor=>EventTrigger，即可挂载 EventTrigger 组件。

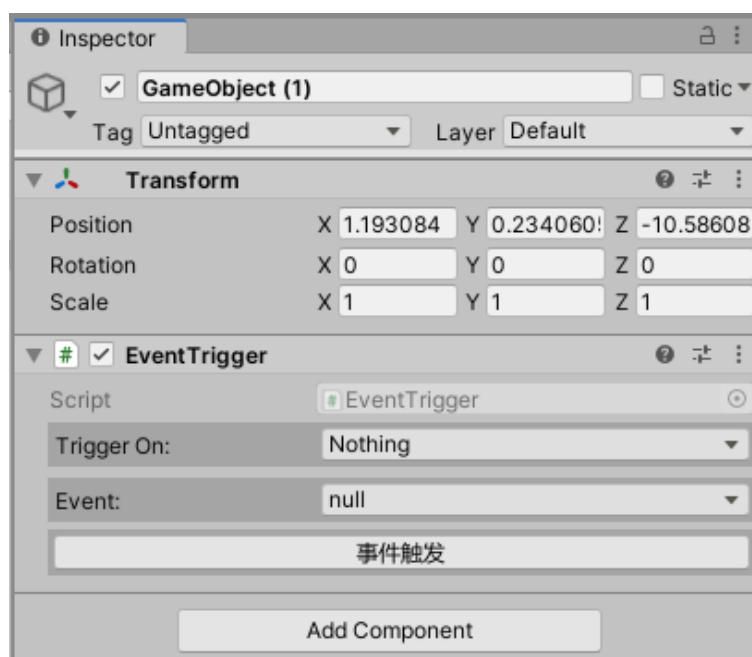


图 1 EventTrigger 一览

- Trigger On 选项提供了事件的触发方式，目前有 Start, TriggerEnter, TriggerExit, Disable, Destroy 几个选项。
- Event 一栏中可以设置要触发的事件，点击右侧按钮进行设置。
- 点击“事件触发”按钮可以不通过 Trigger On 来触发事件，通常用来预览和检查事件内容是否正确。

4.2 通过在脚本中自定义触发

在 unity 的 Project Window 中新建一个自定义脚本，然后将脚本拖动至某一想要触发的 GameObject 上，同时在脚本中声明 EventReference 变量，如下内容：

```
0 个引用
3 public class EventReferenceTest : MonoBehaviour
{
    public EventReference eventA = new EventReference();
    // Start is called before the first frame update
    0 个引用
    3 void Start()
    {
        eventA.PostEvent(gameObject);
    }

    // Update is called once per frame
    0 个引用
    3 void Update()
    {
    }
}
```

图 2 脚本的内容

保存后切换回 Unity，可以看到挂载的脚本中多了一个 Event 的选项，同样也可在右边的按钮中选择要触发的事件，也可直接在脚本中通过对 `eventA.targetEventID` 赋值想要触发的事件 ID。

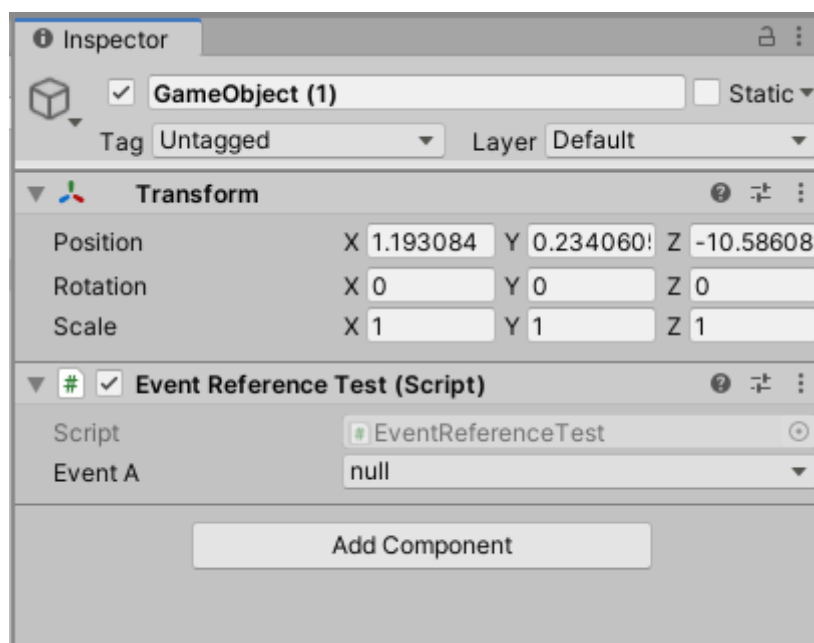


图 3 声明 EventA 变量后 unity 中的面板

链接完事件之后可通过调用 `eventA.PostEvent()` 方法触发事件，该方法需传递一个规定于作用域的 `GameObject`，一般为当前脚本的 `GameObject`，更多 `EventReference` 的功能请参见《脚本 API》章节。

第七章 脚本 API

5.1 AudioEditorManager

描述

该类为插件的总管理器，实现加载配置，监听事件，管理实例的创建和销毁等。一个场景中只能有一个 AudioEditorManager。

静态变量

Public List<AudioClip> GlobalStaticClips;

储存不会被卸载的音频文件，向此列表中添加 AudioClip 可避免被 Manager 卸载。

静态函数

public static void UnloadAllAudioClip();

从内存中卸载在 Manager 中管理的所有的音频文件(目前正在播放的组件所占用的音频文件以及 Global Static Clips 中储存的音频文件除外)，一般在切换场景等过程中使用。

Events

Public static event System.Func<string, AudioEditorData> LoadAudioEditorDataFunc;

当 AudioEditorManager 无法加载目录中的配置文件时引发的事件。请通过此事件确保 AudioEditorManager 能获取到配置文件。例如：

```

0 个引用
public class AudioModule : MonoBehaviour
{
    0 个引用
    void Start()
    {
        AudioEditorManager.LoadAudioEditorDataFunc -= LoadAudioEditorMangerData;
        AudioEditorManager.LoadAudioEditorDataFunc += LoadAudioEditorMangerData;
    }

    2 个引用
    private static AudioEditorData LoadAudioEditorMangerData(string dataPath)
    {
        //传入的参数dataPath为文件目录, 请在此返回AudioEditorData
        return ResourceModule.LoadResourceSync(dataPath) as AudioEditorData;
    }
}

```

图 1 LoadAduioEditorDataFunc 使用示例

public static event System.Func<string, AudioClip> LoadAudioClipFunc;

当 AudioEditorManager 无法获取到 AudioClip 时会引发此事件。如果在编辑器内的 Global Setting 中取消勾选“静态关联 AudioClips”，AudioEditorManager 在需要加载 AudioClip 时都会引发此事件。请通过此事件确保 Manager 能够获取到 AudioClip。传入的参数为对应 AudioClip 的路径。

5.2 EventReference

描述

在脚本中声明的可实现事件触发功能的工具类。

变量

public int targetEventID;

想要触发的事件 ID。

函数

public void PostEvent(GameObject gameObj , Action completeCallBack = null);

向 AudioEditorManager 发送一次事件触发申请。参数 gameObj 为想要触发事件所链接的 GameObject。CompleteCallBack 为事件结束后的回调。

public void OverrideEventTypeOnce(AEEventType whichType);

将暂时覆盖触发的事件中的动作类型，用于下一次 PostEvent()，参数 whichType 为要设置的新的动作类型，目前只有 Play、Pause、Resume、Stop 可以被覆盖。这只是 EventReference 中的设置，并不会改变事件本身的设置。

public void OverrideEventType(AEEventType whichType);

永久覆盖触发的事件中的动作类型。这只是 EventReference 中的设置，并不会改变事件本身的设置。

public static void PostEvent(GameObject gameObject, int eventId ,AEEventType? overrideEventType = null, Action completeCallBack = null);

通过静态方法触发对应的事件，参数为 Event ID，overrideEventType 为覆盖动作，completeCallBack 为事件结束后的回调。

public static void PostEvent(GameObject gameObject,string eventName) ;

通过静态方法触发对应的事件，参数为 Event Name，其他同上。

5.3 RTPC

描述

用于链接相关的 GameParameter 并进行数据的更新。

变量

public int targetGameParameterID;

设置对应于 GameSync 中 GameParameter 的 ID。

函数

public void SetGlobalVlue(float newValue);

为对应 ID 的 GameParameter 传输新的值。

public static void PostGameParameterValue(int gameParameterId,float newValue);

通过 GameparameterId 关联 RTPC 中的属性。

public static void PostGameParameterValue(int gameParameterName,float newValue);

通过 GameparameterName 关联 RTPC 中的属性。

第八章 快捷键

AudioEditor 编辑器的快捷键目前暂时不能自定义，现列出快捷键提供相关参考。快捷键使用需要编辑器在焦点状态才会响应，使用时请多加注意。



图 1 焦点状态



图 2 非焦点状态

| 快捷键 | 其他条件 | 功能 |
|------------|-------------------------------|---------------|
| 主界面通用 | | |
| Ctrl+S | | 保存操作内容 |
| Ctrl+Z | | 撤销操作 |
| Space | 选中 Audio 或 Event 组件时 | 预览播放/停止/恢复 |
| 列表图 | | |
| Ctrl+C | 选中若干组件时（选中组件显示为蓝色） | 复制组件 |
| Ctrl+V | 选中组件时 | 粘贴组件至所选父级组件 |
| Delete | 选中若干组件时（选中组件显示为蓝色），且鼠标在列表图视窗内 | 删除所选组件 |
| 其他 | | |
| Shift+鼠标左键 | 在 SoundSFX 组件的波形图内滑动时 | 快速设置 Delay 数值 |
| Delete | 选中某些属性时 | 删除属性/动作等 |

