

概述：

互动音乐播放器（以下简称播放器）是一套基于 Unity 原生音频系统，帮助游戏音频设计师和作曲实现复杂逻辑的互动音乐的插件。播放器结合了中间件 Wwise 中互动音乐的设计架构与中间件 Fabric 中对 Unity 里 Game Object 层级的使用，同时大幅度简化了上手难度。

播放器适用于各类中小型游戏项目，因为各种技术或商业上的原因不使用 FMOD、Wwise 等中间件，同时需要实现节拍对齐、智能转接、分层等功能的互动音乐。因 Unity Editor 不支持中文界面，播放器所有 UI 均为英文界面。

播放器由刘子奇制作，所有代码均为开源。如遇到 bug 请及时反馈至微信 victor_647

组件简介（页码）：

AudioMixer：一个基础的 Unity Audio Mixer，仅用于将默认音乐输出至其 Music Bus

Editor：存放所有关于 Inspector 中 UI 的脚本，无需用户加载使用

Library：播放器中用到的一些自定义库源码，无需用户加载使用

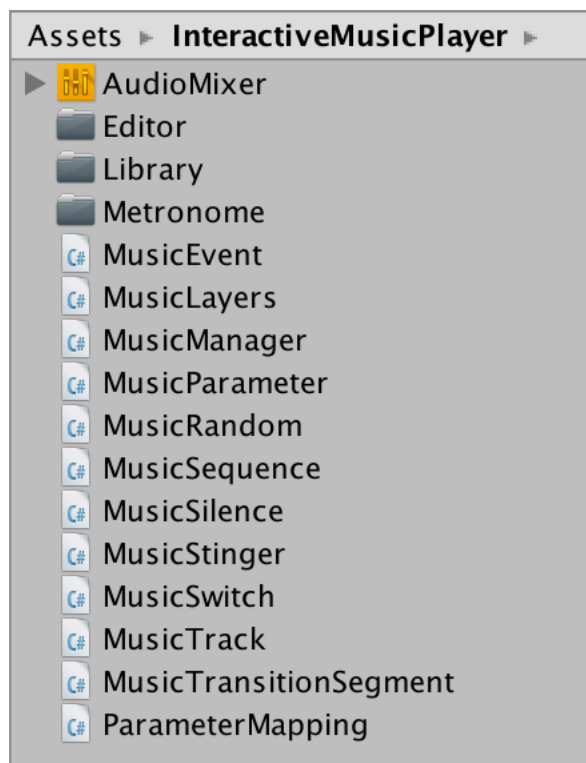
Music Manager (4)：整体音乐的声部和事件管理

Music Event (5)：触发播放/停止/转接等事件

Music Track (8)：存放一段音乐，音乐的最基本单位

Music Layers (11)：多个分轨音乐同时播放

Music Random (12)：随机选择一条音乐播放



Music Switch (14)：根据不同条件选择一条音乐播放

Music Sequence (16)：多条音乐按顺序播放

Music Silence (17)：在音乐间插入一段静音或延迟播放

Music Stinger (18)：在其他音乐播放同时触发短暂乐句或音符

Music Transition Segment (19)：在不同音乐之间转接的过渡段落

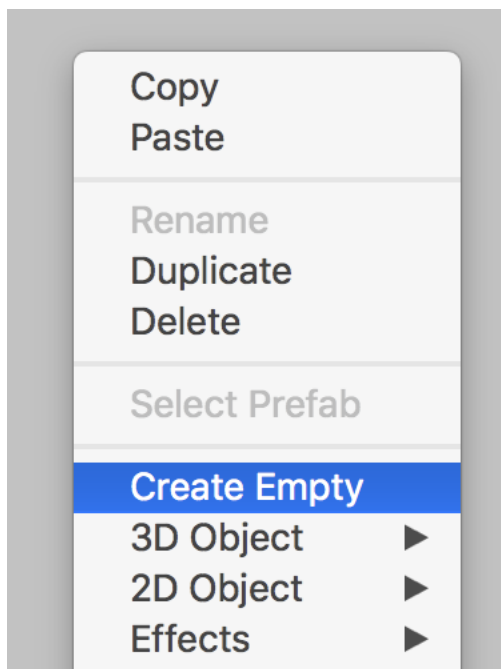
Music Parameter (21)：定义某个与游戏参数对应的音乐控制参数

Parameter Mapping (22)：通过音乐参数动态控制音轨的音量/声道/滤波器截止频率

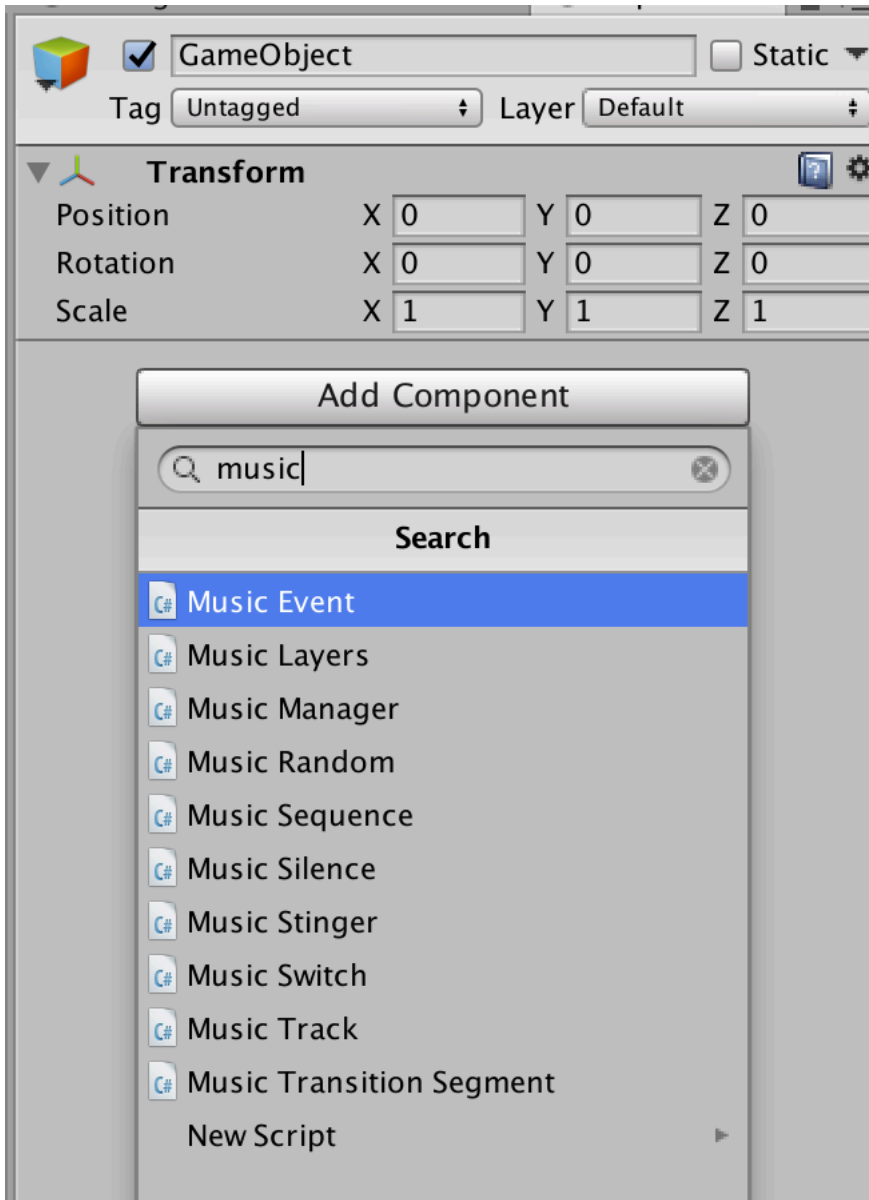
Metronome (24)：可挂载到任意音乐组件上的节拍器，用于测试音乐节拍是否对齐

基本操作：

1. 新建一个空的 Game Object

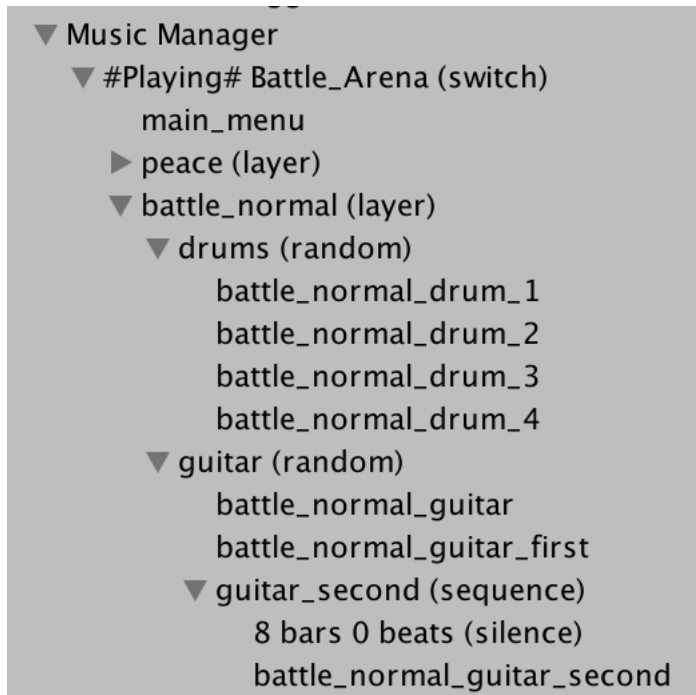


2. 在新建的 Game Object 上添加对应的音乐组件



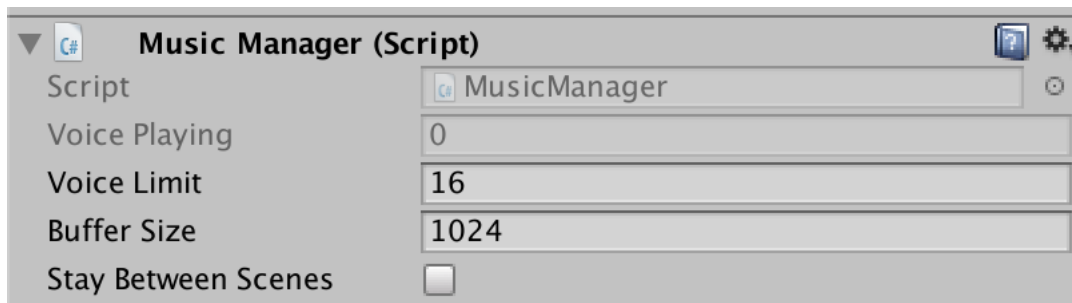
3. 最上层的 Game Object 应当为 Music Manager，以方便管理，但 Music Manager 在特殊情况下也可挂在在其他 Game Object 上（如 Game Manager）
4. 在下属的 Game Object 中，分别添加对应的组件。Game Object 的嵌套方式应当与音乐的封装方式一致

示例使用方式（见下页）：



组件使用方法：

Music Manager



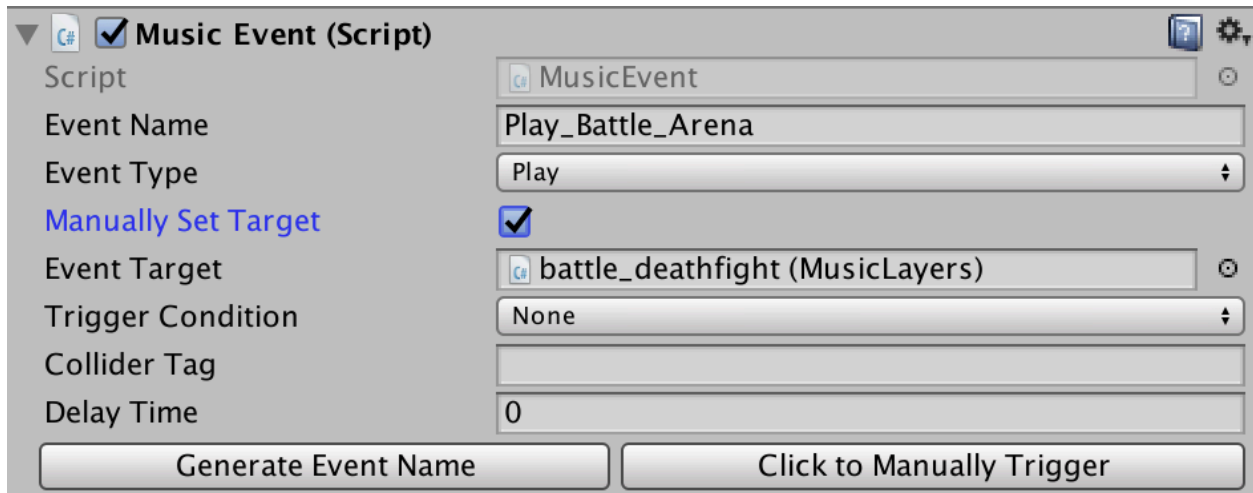
Voice Playing：记录当前有多少轨音乐正在同时播放

Voice Limit：最多同时播放的音轨（包括音效）数量，超出则将音量最小的放入虚拟声部

Buffer Size：音频载入的缓冲采样数，**必须为 2 的 x 次方**，采样数越高则 CPU 消耗越少，但音乐（包括音效）触发延迟会相应增加（在 44100 采样率下，1024 采样对应约 23 毫秒播放延迟）

Stay Between Scenes：在切换 scene 的时候，是否保留 Music Manager

Music Event:



▼ ☒ Music Event (Script)

Script: MusicEvent

Event Name: Play_Battle_Arena

Event Type: Play

Manually Set Target: ☒

Event Target: battle_deathfight (MusicLayers)

Trigger Condition: None

Collider Tag:

Delay Time: 0

Generate Event Name Click to Manually Trigger

Event Name：事件的名称。可手动输入，或点击下方按钮，根据播放的音乐名称和播放方式自动生成。

Event Type：事件的内容，可从下列几项选择



▼ ☒ Music Event (Script)

Script: MusicEvent

Event Name: Play_Battle_Arena

Event Type: Play

Manually Set Target:

Event Target:

Trigger Condition:

Collider Tag:

Delay Time:

Generate Event Name Click to Manually Trigger

- ✓ Play
- Stop
- Stop On Grid
- Transition To
- Change Parameter Value
- Set Switch
- Trigger Stinger

Play：播放音乐

Stop：停止播放音乐（立刻停止，无视转接判定）

Stop On Grid：在下一次转接判定时停止播放音乐

Transition To：在下一次转接判定时转接至音乐

Change Parameter Value：更改音乐上 Parameter Mapping 组件中的参数

Set Switch：更改音乐上的 Switch（音乐必须是 Music Switch）

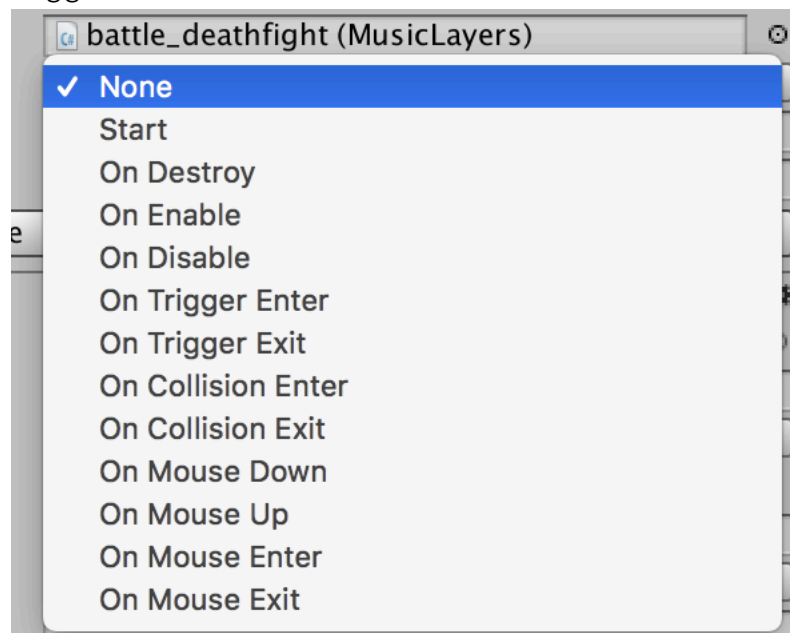
Trigger Stinger：触发 Stinger，必须在 Stinger 对应的音乐正在播放时才有效

注：当 Event Type 设为 Transition To 时，选项会出现一条 Transition From，需要指定从哪条音乐转接；设为 Change Parameter Value 时，会出现 Parameter Name 和 Parameter Value，填写参数的名字和数值；设为 Set Switch 时，会出现 Switch Name 填写事件切换至 Switch 的名字。

Manually Set Target：如未勾选，会自动获取同一 Game Object 上的音乐

Event Target：响应事件的目标音乐。若勾选 Manually Set Target，需要手动拖拽

Trigger Condition：事件的触发方式，可从下列几项选择



None：手动触发或通过游戏内脚本触发

Start：游戏开始时或挂载的 Game Object 生成时

On Destroy：挂载的 Game Object 从场景中删去时

On Enable：该 Music Event 组件或挂载的 Game Object 开关被打开时

On Disable：该 Music Event 组件或挂载的 Game Object 开关被关闭时

On Trigger Enter：挂载的 Game Object 与其他 Game Object 进入物理重叠状态时

On Trigger Exit：挂载的 Game Object 与其他 Game Object 离开物理重叠状态时

On Collision Enter：挂载的 Game Object 与其他 Game Object 发生物理碰撞时

On Collision Exit：挂载的 Game Object 与其他 Game Object 结束物理碰撞时

On Mouse Down：挂载的 Game Object 被鼠标按下点击键时

On Mouse Up：挂载的 Game Object 被鼠标释放点击键时

On Mouse Enter：鼠标移动至挂载的 Game Object 区域内时

On Mouse Exit：鼠标离开挂载的 Game Object 区域内时

Collider Tag：如触发方式为 On Trigger 或 On Collision 相关的四种，需要检测带有特定 Tag 标识的 Game Object 进入判定范围，则需要在此填写标识内容，留空则默认检测所有 Game Object

Delay Time：若事件需要延迟触发，在此填写延迟时间

Generate Event Name 按钮：通过

游戏中所有音乐相关的事件都需要添加一个 Music Event 来注册。Event 的位置既可放在对应的音乐组件上，也可挂载在场景物件上，取决于事件的触发方式。如需要使用 Trigger Enter 等基于物理的触发方式，应当挂载在对应的场景物件上。

在游戏开始后，Music Manager 会记录每个 Music Event，通过名称识别。其他游戏脚本中触发 Music Event 需要执行以下语句：

```
MusicManager.Instance.PostEvent(string eventName);
```

若当前脚本中含有该 Music Event 的 reference，也可使用：

```
MusicManager.Instance.PostEvent(MusicEvent event);
```

Music Track:

音乐最基本的单位，每条 Music Track 对应一个音乐文件（wav 或 ogg 格式）

强烈建议在从 DAW 导出音乐时，保留尾音，通过 Post Exit 的设置可以保证无缝完美循环

（以下设置为所有可播放的组件共有）

Override Parent Rhythm：是否覆盖上层音乐组件的节拍设置，勾选则以下三项可以编辑

Tempo：音乐的速度，每分钟街拍数

Beats Per Bar：每小节拍子数

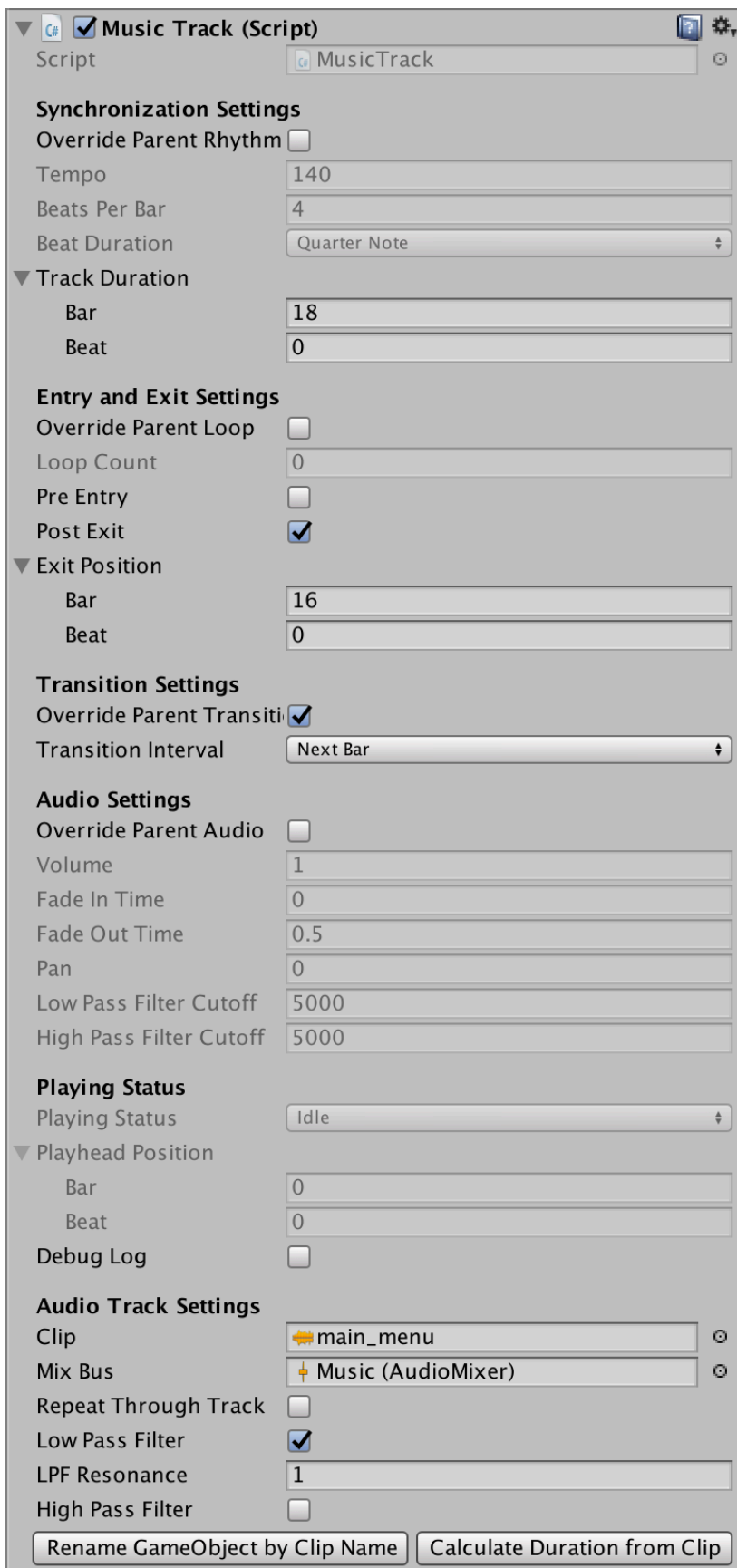
Beat Duration：节拍时长，可选四分音符、八分音符或十六分音符

Track Duration：整段音乐的总长度，包括尾音预留的空拍，以小节数/余下拍子数格式填写

Override Parent Loop：是否覆盖上层音乐组件的循环次数

Loop Count：音乐循环次数，0 为无限循环，1 为单次播放，其余为有限循环次数

Pre Entry：音乐循环或主体部分开始前是否拥有前奏，如不完全小节中的音符



Music Track (Script)

Script: MusicTrack

Synchronization Settings

Override Parent Rhythm ☐

Tempo: 140

Beats Per Bar: 4

Beat Duration: Quarter Note

Track Duration

Bar: 18

Beat: 0

Entry and Exit Settings

Override Parent Loop ☐

Loop Count: 0

Pre Entry ☐

Post Exit ☒

Exit Position

Bar: 16

Beat: 0

Transition Settings

Override Parent Transition ☒

Transition Interval: Next Bar

Audio Settings

Override Parent Audio ☐

Volume: 1

Fade In Time: 0

Fade Out Time: 0.5

Pan: 0

Low Pass Filter Cutoff: 5000

High Pass Filter Cutoff: 5000

Playing Status

Playing Status: Idle

Playhead Position

Bar: 0

Beat: 0

Debug Log ☐

Audio Track Settings

Clip: main_menu

Mix Bus: Music (AudioMixer)

Repeat Through Track ☐

Low Pass Filter ☒

LPF Resonance: 1

High Pass Filter ☐

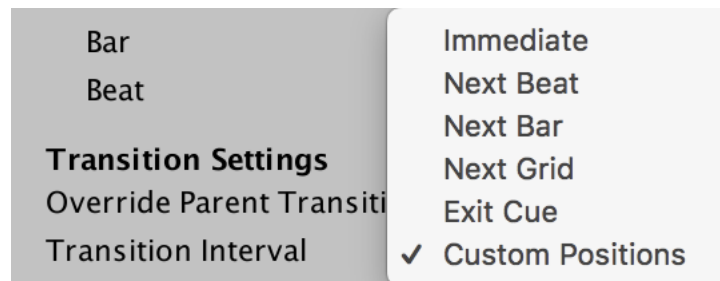
Rename GameObject by Clip Name Calculate Duration from Clip

Post Exit：音乐循环或主体部分结束时是否带有尾音，如混响产生的湿声

Entry/Exit Position：音乐主体进入/结束时的节拍位置

Override Parent Transition：是否覆盖上层音乐组件中的转接设置

Transition Interval：转接检测间隔，有下列几项选择



Immediate：立刻进入下一段音乐

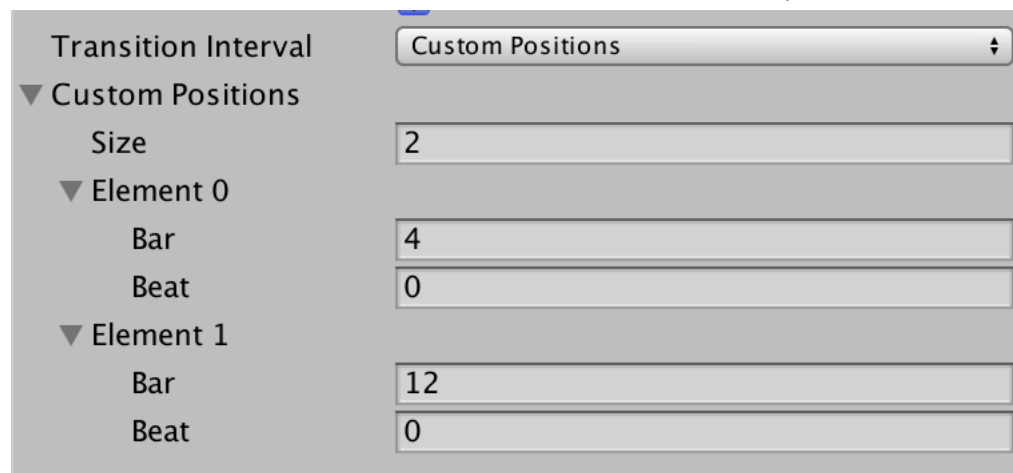
Next Beat：下一拍转接至下一段音乐

Next Bar：下一小节

Next Grid：下一段落，段落长度可在出现的 Grid Length 中手动填写

Exit Cue：当前音乐结束时才进行转接

Custom Positions：手动输入可以转接的节拍位置列表（会自动包含 Exit Cue）



Override Parent Audio：是否覆盖上层音乐组件中音轨的参数设置

Volume：音量，范围为 0-1，默认为 1（最大音量）

Fade In/Out Time：淡入/淡出时间。默认淡出为 0.5 秒以实现更自然的转接

Pan：左右声道控制，范围为-1 到 1，默认为 0（中央）

Low/High Pass Filter Cutoff：低通/高通滤波器的截止频率，默认为 5000Hz

Playing Status：当前音乐的播放状态，共有 Idle (停止)，PreEntry (前奏)，Playing (主体部分)，Post-Exit (尾奏) 四种状态

Playhead Position：当前音乐播放的节拍位置

Debug Log：是否在 Console 里显示播放事件

(以下设置为 Music Track 及类似的 Music Stinger、Music Transition Segment 独有)

Clip：播放的音乐文件

Mix Bus：音频信号输出的通道，默认为 Music

Repeat Through Track：为了节省资源空间，如在一段较长的音乐中有小段 loop 不断重复的（如 16 小节的循环中，有 2 小节的鼓点需要重复 8 次），点选该选项会自动填充至整段音乐。**在从 DAW 导出小段 loop 时，不得留有尾音，选中该项则会自动删去 Pre-Entry 和 Post-Exit**

Low/High Pass Filter：是否打开低通/高通滤波器

LPF/HPF Resonance：滤波器共振值，越高则在截止频率附近声音越尖锐，默认为 1

Rename GameObject by Clip Name 按钮：点击可自动重命名 Game Object 为音频文件名

Calculate Duration from Clip 按钮：点击可根据当前节拍速度自动得到整段音频以节拍为单位的长度

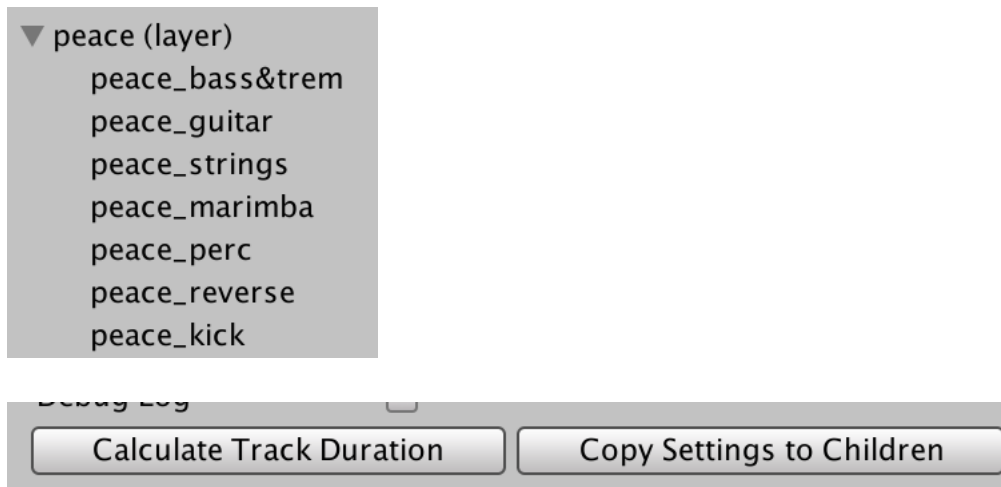
Music Track 是播放器最底层的组件，也是最基本的音乐单位。其对应的 Game Object 应为最底层，不应当拥有任何更下层的音乐组件。

一条 Music Track 仅能播放单个音频文件，如需要多层音乐同时播放，请使用 Music Layers 组件，并将每一条 Music Track 作为其下层的 Game Object

Music Layers:

用于同时播放多条音轨，以实现分层控制音乐。

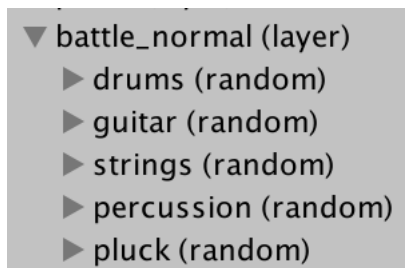
Music Layers 的界面除下方两个按钮外没有任何独特内容，只需要将所有需要同时播放的 Music Track 放在其下层即可。



Calculate Track Duration 按钮：将音乐长度设置为下层中最长的音轨的长度

Copy Settings to Children 按钮：将本 Music Layers 中的所有设置一键复制至下层音轨

除了 Music Track 之外，Music Layers 还可以嵌套其他组件同时使用，如 Music Random



需要注意的是，不管拥有多少下层音乐组件，所有音轨的速度以及转接方式应当相同。若忘记设置音乐长度，则会自动与从下层中最长的一条保持一致。下层不同音乐的节拍、循环次数和长度可以保持不同，以实现不同乐器之间的异步循环。

Music Random:

用于随机选取一条音乐播放。

Random Settings

Random Mode: Random

Avoid Repeat: ☒

Random On Loop: ☒

▼ Random List

Size: 4

▼ Element 0

Music: battle_normal_drum_1 (MusicTrack)

Percentage: 25

▼ Element 1

Music: battle_normal_drum_2 (MusicTrack)

Percentage: 25

▼ Element 2

Music: battle_normal_drum_3 (MusicTrack)

Percentage: 25

▼ Element 3

Music: battle_normal_drum_4 (MusicTrack)

Percentage: 25

Fill with Children Tracks Copy Settings to Children

Random Mode：随机选择方式

Random，完全随机

Shuffle，轮流随机，直到所有选择都被播放过一遍才可重复

Avoid Repeat：不重复上一条选择的音乐（只在 Random 模式下可用）

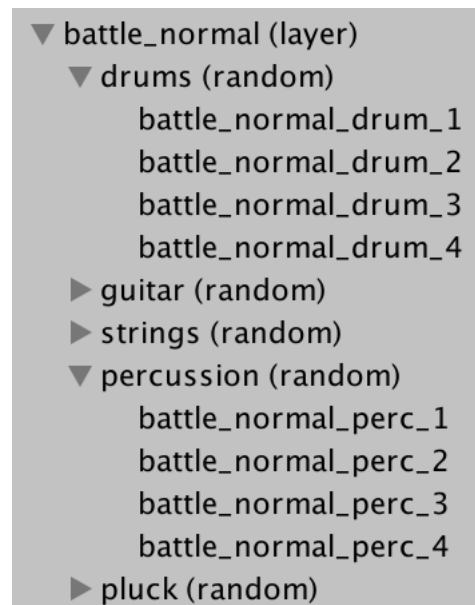
Random On Loop：若音乐循环，每次循环会进行一次随机选择

Random List：可被随机选中的音乐列表。若手动添加，需要在 Size 栏输入选项总数，并拖拽带有音乐组件的 Game Object 进每项的 Music 栏。在 Percentage 中输入每项的触发几率。若输入的所有项数之和不为 100，游戏开始时会自动重新加权计算为百分制。

Fill with Children Tracks 按钮：自动寻找下层的音乐，按照相等触发几率填充至随机列表

Copy Settings to Children 按钮：同其他音乐组件

Random Music 示例：在 Music Layer 下，每次循环为鼓乐器随机选择一个节奏型播放



Music Random 下层至少需要有 2 条音乐可供选择，和其他任何音乐组件皆可互相嵌套（Music Transition Segment 与 Music Stinger 除外），取决于需要实现的设计需求。

Music Random 直属下层的所有音乐，其转接间隔会被自动设置成 Exit Cue，同时 **Loop Count** 会被自动设为与 Music Random 一致。

Music Switch:

用于通过外部变量选取一条音乐播放。

同 Music Random，需要填充下层音乐作为可选对象，或点击 Fill with Children Tracks 按钮自动填充。

Switch Name 为每个选项从外部触发时填写的对应名称，默认与下层音乐同名。

Switch Settings

Switch To Same Position ☐

▼ Switch List

Size

▼ main_menu

Switch Name

Music

▼ peace

Switch Name

Music

▼ battle_normal

Switch Name

Music

▼ battle_deathfight

Switch Name

Music

▼ battle_losing

Switch Name

Music

▼ battle_winning

Switch Name

Music

▼ invisible

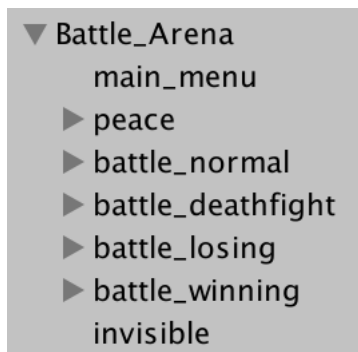
Switch Name

Music

Current Switch

Switch To Same Position：若勾选，则切换音乐时，会切换至不同音乐的同一播放位置继续播放，而非从头开始播放，适用于同一首音乐不同表现形式之间的切换，也可使用 Music Layers + Parameter Mapping 连接音量间接实现。**勾选此项时，所有下层音乐必须与 Music Switch 的长度及循环设置相同，否则可能出现同步错误**

Music Switch 主要在游戏场景等状态切换，需要无缝衔接至另一段音乐时使用。**若 scene 有变更，必须勾选 Music Manager 上的 Stay Between Scenes。**



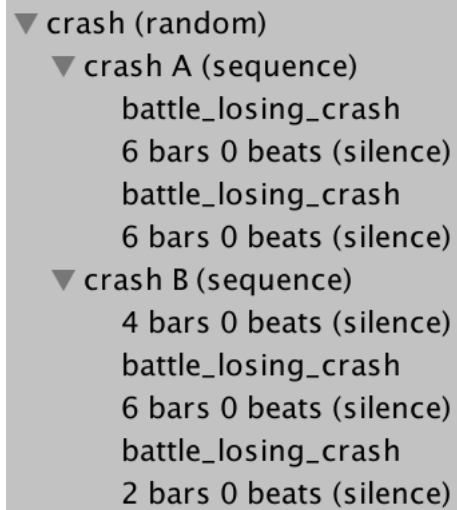
在播放对应的音乐时，Music Switch 的播放进度会与被选中播放的音乐保持同步。播放中途收到改变 Switch 的事件时，当前音乐会根据设定的转接间隔，自动转接到新 Switch 对应的音乐上。

在需要设置同一个 Music Switch 的不同 Switch 时，不需要为每一个 Switch 创建一个 Music Event，只需创建一个，再通过程序直接设置不同的 Switch，方法如下：

```
MusicManager.Instance.SetSwitch(string eventName, string switchName);
```

Music Sequence:

用于建立一个按顺序播放的列表，列表中的音乐按 Game Object 排列顺序从上至下播放。



▼ crash (random)
 ▼ crash A (sequence)
 battle_losing_crash
 6 bars 0 beats (silence)
 battle_losing_crash
 6 bars 0 beats (silence)
 ▼ crash B (sequence)
 4 bars 0 beats (silence)
 battle_losing_crash
 6 bars 0 beats (silence)
 battle_losing_crash
 2 bars 0 beats (silence)

Music Sequence 在 Music Track 的基础上，仅带有 Calculate Track Duration 按钮。Music Sequence 的时长会由所有下层音轨之和自动计算。下层音乐若循环超过 1 次，计时则会累加循环次数。若下层中有任何一段无限循环的音乐，则 **Music Sequence** 的总长度会在播放该音乐时与该音乐保持一致。

Music Sequence 直属下层的所有音乐，其转接间隔会被自动设置成 Exit Cue。

Music Silence:

用于填充一段空白时长的音轨，或延迟播放一段音乐。

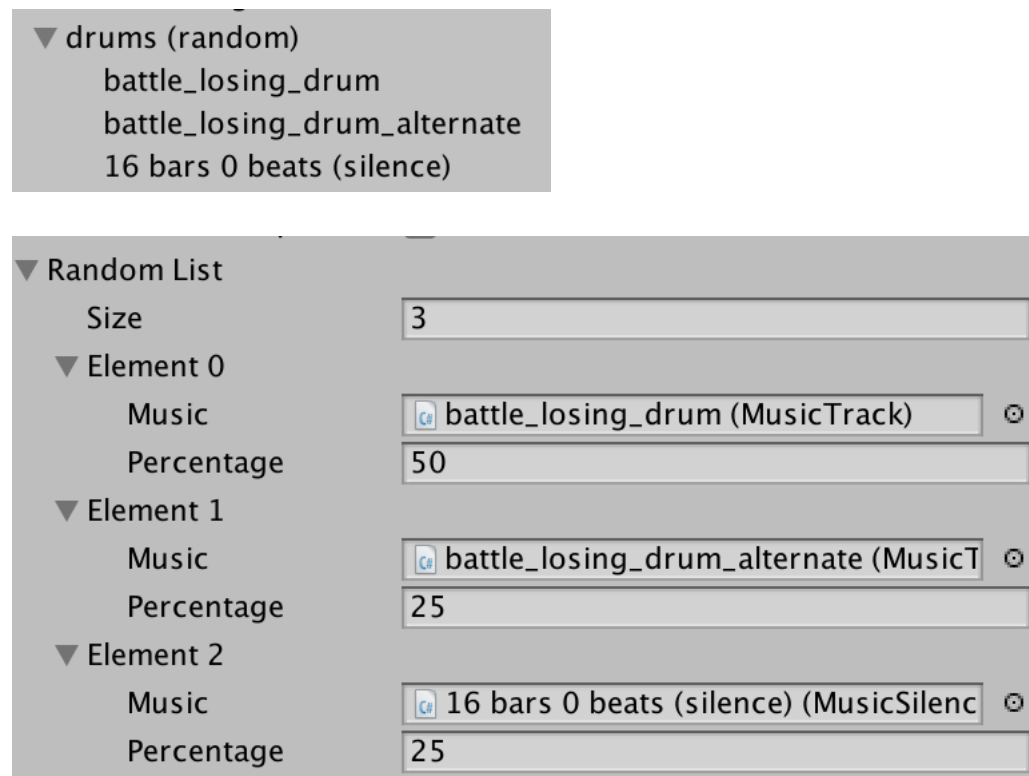
Music Silence 的界面与 Music Track 基本相同，但没有音频文件相关的选项。

Music Silence 会自动根据时长重命名 Game Object。

Music Silence 不可以拥有 Pre-Entry 和 Post-Exit。

Music Silence 同为最底层的音乐组件，下层不可嵌套任何其他组件。

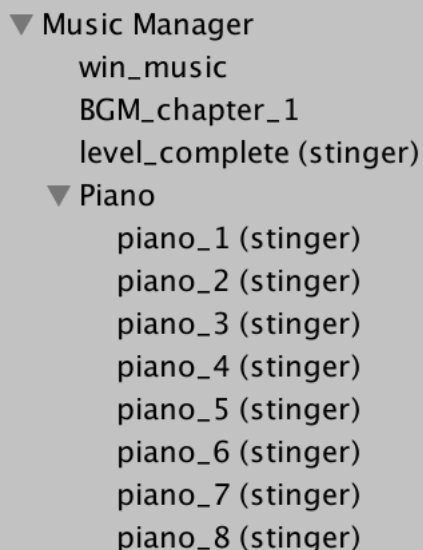
使用示例：



在 Music Random 下，添加一条随机选项为 Music Silence，设置 Percentage 为 25，则该段音乐有四分之一的几率不播放任何鼓点。

Music Stinger:

用于在当前播放音乐之上，在节拍上触发音符或乐句。



```
▼ Music Manager
  win_music
  BGM_chapter_1
  level_complete (stinger)
  ▼ Piano
    piano_1 (stinger)
    piano_2 (stinger)
    piano_3 (stinger)
    piano_4 (stinger)
    piano_5 (stinger)
    piano_6 (stinger)
    piano_7 (stinger)
    piano_8 (stinger)
```

Music Stinger 的界面与 Music Track 基本一样，除了新增一项 Host Music，决定了当什么音乐正在播放时可以触发本条音乐。需要将可播放的**最上级**的音乐组件对应的 Game Object 拖拽上去。

Music Stinger 中的 Transition Interval 指的是 Stinger 触发的最短间隔。如设为 Next Bar，则音乐会等到下一小节的第一拍才会播放。

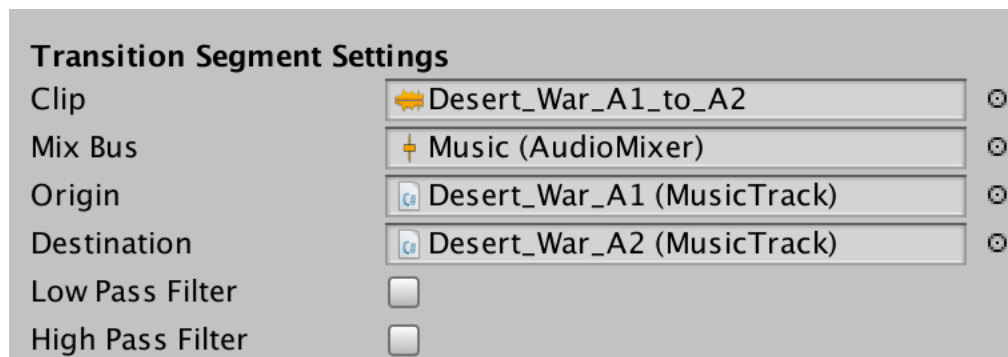
如在本案例中，玩家按键盘的键可以触发钢琴音符，则 Host Music 选取 BGM_chapter_1，触发间隔选取 Immediate（立刻）

如 Stinger 中带有 Pre-Entry，则会比对当前触发间隔剩余时间与 Pre-Entry 长度。若**剩余时间短于 Stinger 的 Pre-Entry 长度，则会根据触发间隔决定触发延迟**。若触发间隔为 Custom Positions 或 Exit Cue，则会立刻触发；其他情况下，会等待下一次触发间隔时触发。

音乐之间转接的时间判定，也遵循以上逻辑。若转接目标带有 Pre-Entry，为保证转接合拍，请在预计转接时间比转接目标的 Pre-Entry 长度更早之前触发转接事件。

Music Transition Segment:

用于两段音乐转接之间播放的过渡段落。



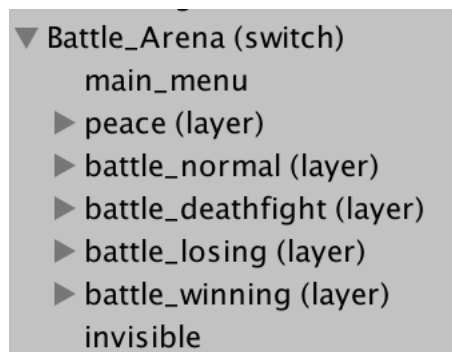
Origin：触发 Music Transition Segment 的音乐，转接之前播放的音乐

Destination：播放完 Music Transition Segment 后转接至的音乐

其他设置与 Music Track 相同。




Origin 和 Destination 其中一项可留空，则从任意音乐转接或转接至任意音乐时都使用本条 Music Transition Segment。

使用示例：





在 Music Switch 下实现 Battle 和 Peace 两种状态下音乐互相切换的转接段落，则需要以下两段 Music Transition Segment：

Transition Segment Settings	
Clip	 to_battle
Mix Bus	 Music (AudioMixer)
Origin	 peace (MusicLayers)
Destination	None (Music Component)

Transition Segment Settings	
Clip	 to_peace
Mix Bus	 Music (AudioMixer)
Origin	None (Music Component)
Destination	 peace (MusicLayers)

需要排除 Invisible 状态和 Peace 状态互相进入的情况，则需要添加两条额外的 Music Transition Segment，指定 Origin 和 Destination，**但 Clip 需要留空**：

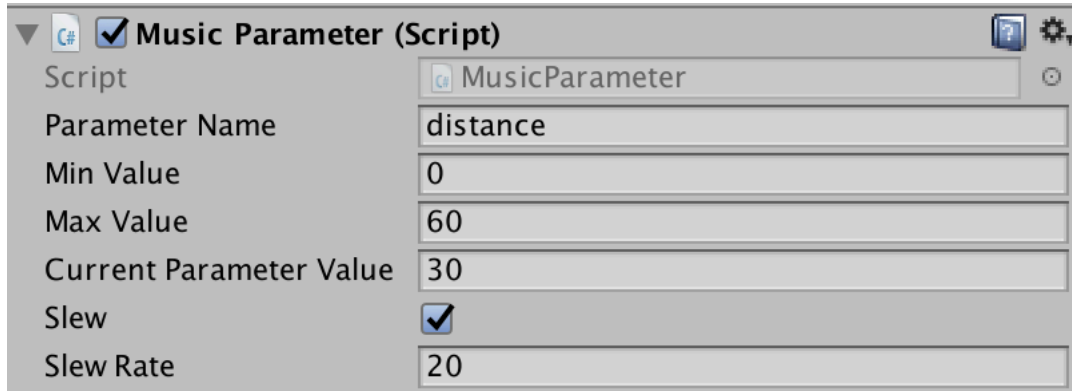
Transition Segment Settings	
Clip	None (Audio Clip)
Mix Bus	 Music (AudioMixer)
Origin	 invisible (MusicTrack)
Destination	 peace (layer) (MusicLayers)

Transition Segment Settings	
Clip	None (Audio Clip)
Mix Bus	 Music (AudioMixer)
Origin	 peace (layer) (MusicLayers)
Destination	 invisible (MusicTrack)

若 Music Transition Segment 的 Origin 和 Destination 相同，在普通循环状态下不会触发过渡段落，**但在 Music Random 重复选中同一段音乐、Music Sequence 下同一段音乐出现多次、以及 Music Switch 下重复切换至同一 Switch 时**，会触发对应的过渡段落。

Music Parameter:

用于定义一条可控制音乐的参数



Parameter Name：参数的名字。名字为唯一识别，不可有两个同名参数

Min/Max Value：参数最小/最大值

Current Parameter Value：参数初始值，游戏中参数变化后会更新为参数当前值

Slew：当参数突然变化时，是否渐变（类似淡入淡出）

Slew Rate：参数渐变速率，单位/秒

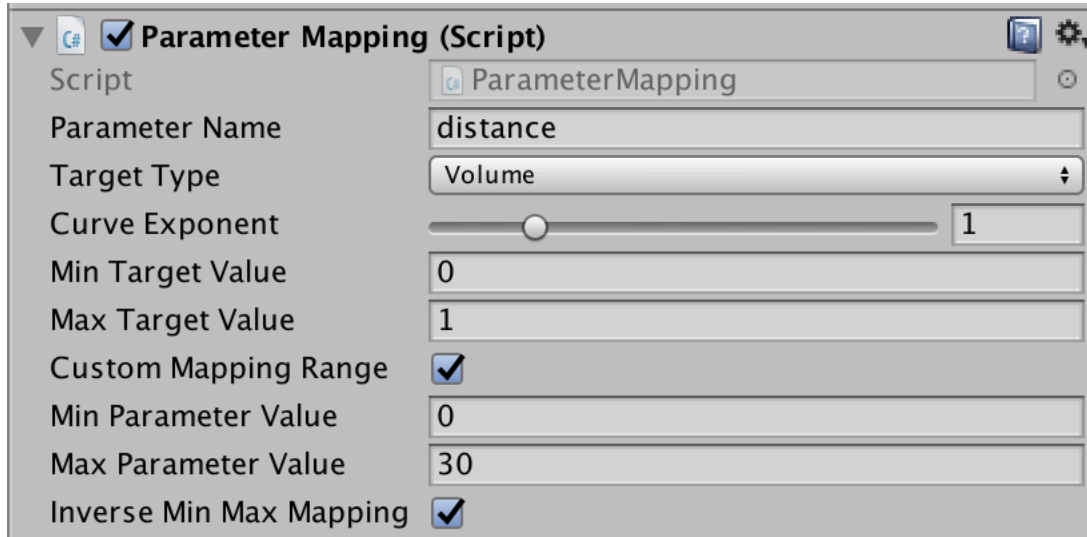
Music Parameter 可以挂载在任何不会消失的 Game Object 上，但建议挂在 Music Manager 上以便管理。

当 Music Parameter 需要和游戏中某个变量同步时，需要在含有游戏变量的脚本中使用以下脚本，并将 value 替换成游戏变量即可：

```
MusicManager.Instance.SetParameterValue(string parameterName, float value);
```

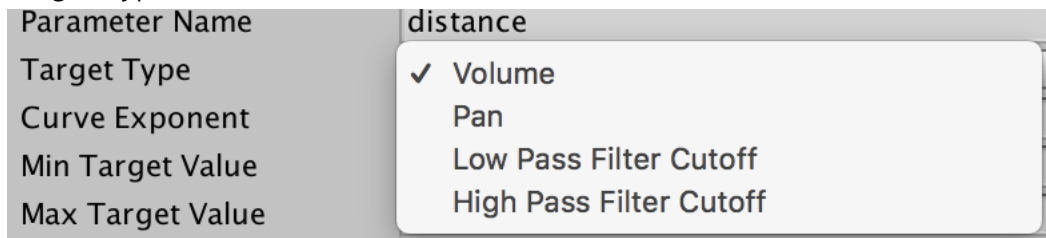
Parameter Mapping:

用于将音乐参数对应到音乐的变化中。



Parameter Name：使用的参数名称，需要与 Music Parameter 上的名称一致

Target Type：参数控制音乐的内容：



Volume：音量

Pan：左右声道

Low/High Pass Filter Cutoff：低通/高通滤波器截止频率

Curve Exponent：控制曲线的曲率。如当前参数位于正中央(50%)时，Curve Exponent 为 1 则音量也为 50%；Curve Exponent 为 2 时，音量为 25%；Curve Exponent 为 0.5 时，音量为 71%，以此类推。**Curve Exponent 越小，在接近于参数最小值时，曲线斜率越大**

Min/Max Target Value：控制目标范围的最小/最大值。如控制 Pan 时，完全左右声道宽度则应分别设置为 -1 和 1

Custom Mapping Range：曲线两端是否只用参数中间的一段，若勾选否，则囊括整个参数定义范围。如上图情况中，距离的定义为 0-60，而这里只用了 0-30，则 30 以上的距离都使用 30 时对应的音量。

Inverse Min Max Mapping：是否将参数最大最小值互换。如上图情况中，距离越近（数值越小），音量应当越大而非越小，则应勾选此选项。

Parameter Mapping 可以挂载在任何一个带有音乐组件的 Game Object 上，影响该 Game Object 上的音乐组件及其下层。若下层有任何音乐需要覆盖或无视 **Parameter Mapping**，应当在下层各个组件中分别挂载 **Parameter Mapping**，而非在上层总组件上挂载。

Metronome:

用于测试音乐拍子是否对齐的节拍器。



Click Down Beat：每小节第一拍的音效文件

Click Normal：每小节其余拍子的音效文件

Volume：节拍器音量

播放器中自带一套节拍器音效文件，也可替换成自己的文件。

Metronome 只需挂载在需要对应节拍的音乐组件的同一 Game Object 上即可，当音乐开始播放时会自动播放，音乐停止时也会自动停止。