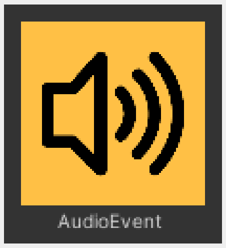
# Welcome!

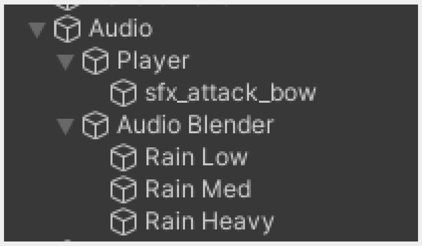
This is a documentation for my custom audio scripts for Unity's native audio system. If you have any questions, reach out to jamieleesounds@gmail.com. Download the scripts **here**.

## Overview

Audio Event and Audio Cue are designed for two common ways of handling audio in Unity. The first one is a **prefab-based** approach using Audio Event. A **prefab-based** approach allows audio events to organize under the game object hierarchy it's working with. This way, you can change and test values directly within the game object hierarchy. It also uses a monobehavior class which has useful methods for processing effects in real time. The downside is that Audio Event objects are only available within a scene and always have to live under the game object. This means the more complicated a scene is, the deeper you will have to keep track of where the Audio Event objects live.

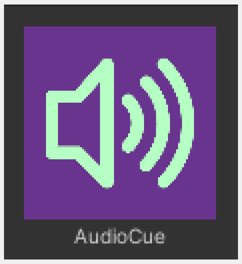


*Audio Event*

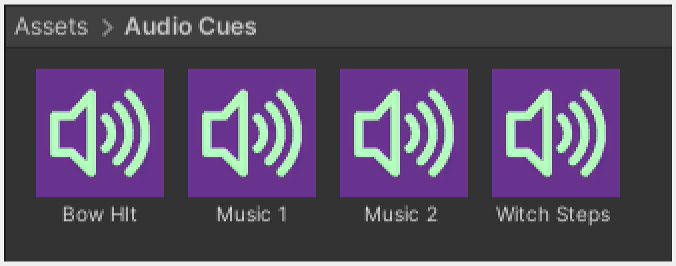


*Audio Blender using Audio Events*

The second way of handling audio is through Unity's **Scriptable Object** class. Scriptable objects are data/method containers that persist throughout the project. The biggest benefit of using scriptable objects is to have a better organization system and workflow for audio designers. Once you set up the script to play an **AudioCue** scriptable object, you can make design changes within the AudioCue asset instead of in the game object hierarchy. The down side is it's a bit more work to set up the AudioCue assets and audio files separately and sometimes it can be an overkill for smaller projects.



*Audio Cue*



*Audio Cues in the Assets folder*

## So which one should I use?

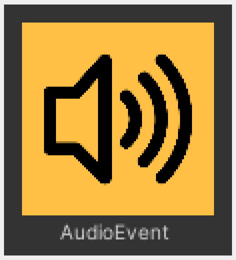
- Use **AudioEvent** if you're building smaller prototypes or have a feature that needs to be modified during runtime.
- Use **AudioCue** if you want dedicated organization for audio assets and easier editing throughout the project instead of scene-dependent.

---

## Table of Contents

---

# [Audio Event]

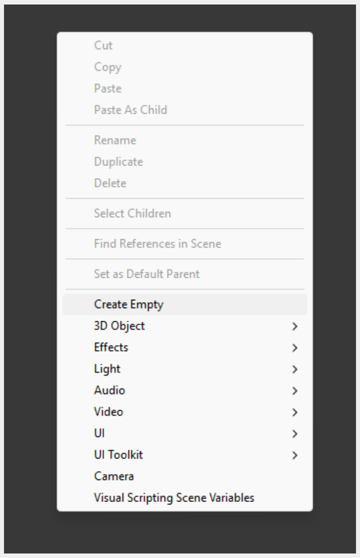


## Overview

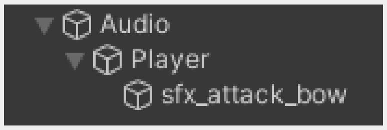
Audio Event is designed to build a prefab-based audio system. Load **AudioEvent.cs** script in your project and attach to corresponding game objects. It will populate [Unity's Audio Source]() and override Audio Source settings. It also has custom features, such as **different play types**, **randomizing start point**, **random pitch and volume, fade in**, and **low pass filter** effects. When not sure about any features, hover over the text and it will show help messages.

## How to use Audio Event

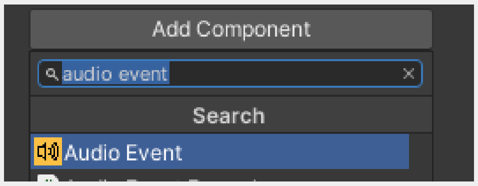
1. Right Click on Unity's Hierarchy panel and create an empty object.



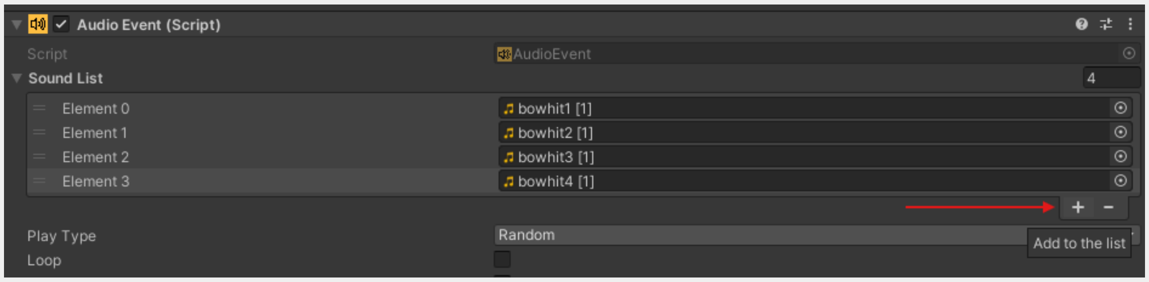
2. Use the empty objects to create a separate hierarchy just for audio or directly below the parent object it belongs to. Both ways technically work the same, it's just about personal preference for organizing!



3. Click on the empty object and add the Audio Event component in the inspector.



4. Load your audio files under Sound List by clicking on + sign.

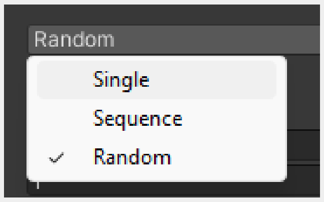

Audio Event (Script)

| | |
|---|---|
| Script | AudioEvent |
| Sound List | 4 |
| Element 0 | bowhit1 [1] |
| Element 1 | bowhit2 [1] |
| Element 2 | bowhit3 [1] |
| Element 3 | bowhit4 [1] |
| Play Type | Random |
| Loop | |

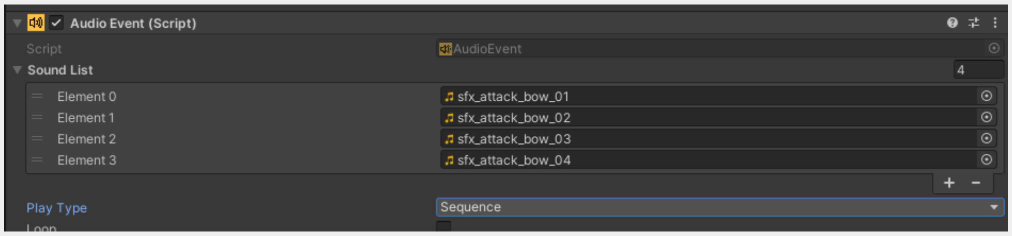
5. Configure parameters. See below for details.

## Play Types and when to use them

There are three different play types in Audio Event. Choose appropriate play types for different sound effects.



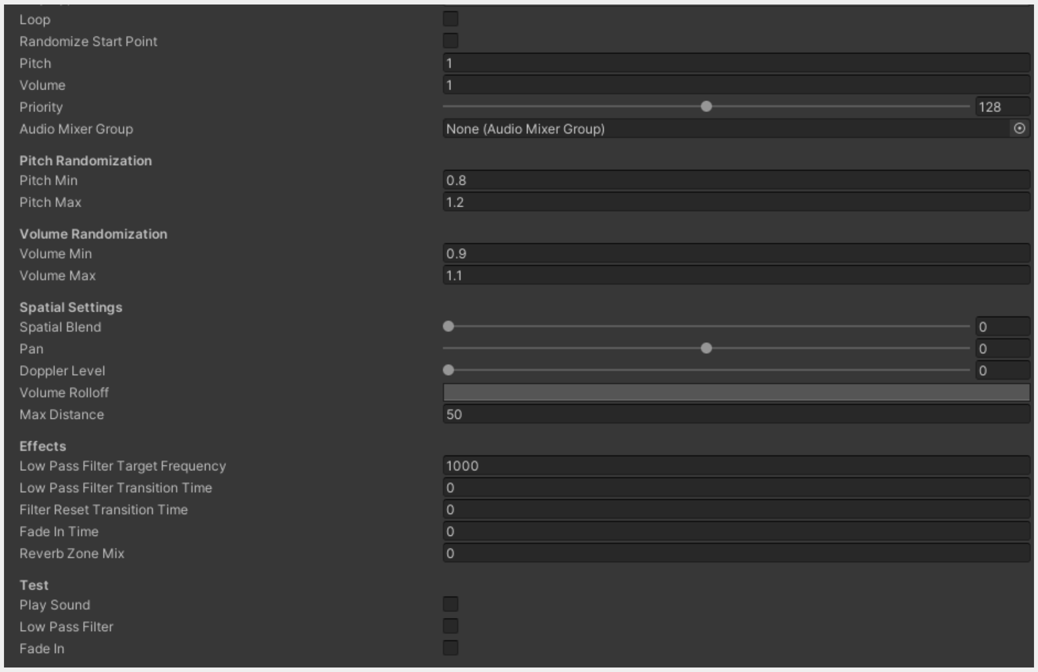
- **Single**: when selected, it will only play the file in the Element 0 spot. This play type is recommended for longer files such as ambience or music.
- **Sequence**: use this play type for a series of sounds that need to play in sequence. In this example, it will play sfx_attack_bow_01 then 02, 03, 04 in order; and loop back to the top of the list and repeat in order.


Audio Event (Script)

| | |
|---|---|
| Script | AudioEvent |
| Sound List | 4 |
| Element 0 | sfx_attack_bow_01 |
| Element 1 | sfx_attack_bow_02 |
| Element 2 | sfx_attack_bow_03 |
| Element 3 | sfx_attack_bow_04 |
| Play Type | Sequence |
| Loop | |

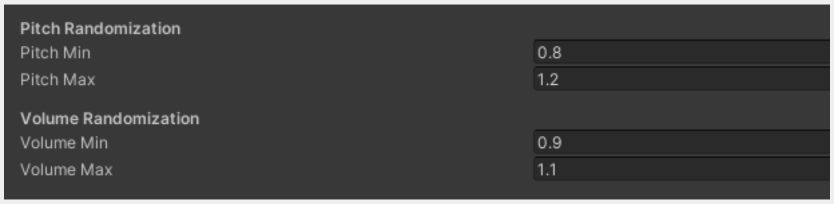
- **Random**: when selected, it will choose a clip in the Sound List. Next time it's triggered, it selects a different clip than the last played clip. Use this play type for short sounds that need to repeat a lot such as footsteps, physics impacts, attack sounds, and so on. Randomizing clips will help prevent ear fatigue and improve immersion.

## Other Features



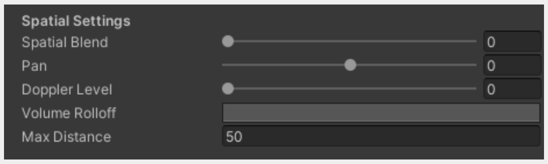
| | |
|---|---|
| Loop | ☐ |
| Randomize Start Point | ☐ |
| Pitch | 1 |
| Volume | 1 |
| Priority | 128 |
| Audio Mixer Group | None (Audio Mixer Group) |
| **Pitch Randomization** | |
| Pitch Min | 0.8 |
| Pitch Max | 1.2 |
| **Volume Randomization** | |
| Volume Min | 0.9 |
| Volume Max | 1.1 |
| **Spatial Settings** | |
| Spatial Blend | 0 |
| Pan | 0 |
| Doppler Level | 0 |
| Volume Rolloff | |
| Max Distance | 50 |
| **Effects** | |
| Low Pass Filter Target Frequency | 1000 |
| Low Pass Filter Transition Time | 0 |
| Filter Reset Transition Time | 0 |
| Fade In Time | 0 |
| Reverb Zone Mix | 0 |
| **Test** | |
| Play Sound | ☐ |
| Low Pass Filter | ☐ |
| Fade In | ☐ |

- **Loop**: check when the audio file needs to loop. Only available for Single Play Type.
- **Randomize Start Point**: selects a random start point in the audio clip. It works with short sounds as well but longer sounds would benefit more for this feature. e.g.) leaf rustling, rain ambience, etc.
- **Pitch**: sets base pitch.
- **Volume**: sets base volume.
- **Priority**: sets priority of the audio object. This feature can be used to build a dynamic mixing system or to implement a volume ducking feature. 0 is the highest priority and 256 is lowest.
- **Audio Mixer Group**: sets audio mixer group (which route to pass the audio through).
- **Randomization**: a useful feature to give more variations to the sounds. Experiment with different values for pitch and volume.
  - Set minimum and maximum value for each pitch and volume. When the audio is triggered, it will play the clip at a random pitch and volume value within the range you set.



| | |
|---|---|
| **Pitch Randomization** | |
| Pitch Min | 0.8 |
| Pitch Max | 1.2 |
| **Volume Randomization** | |
| Volume Min | 0.9 |
| Volume Max | 1.1 |

- **Spatial Settings**

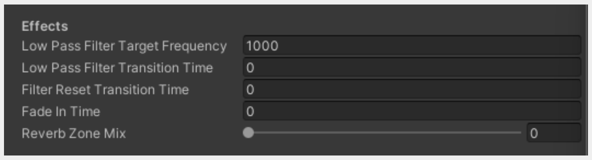


Spatial Settings

| | |
|---|---|
| Spatial Blend | 0 |
| Pan | 0 |
| Doppler Level | 0 |
| Volume Rolloff | |
| Max Distance | 50 |

  - **Spatial Blend**: sets 2D or 3D attenuation. 2D at 0 and 3D at 1.
  - **Pan**: sets which side to play the audio. Left at -1, right at 1, center at 0.
  - **Doppler Level**: enables doppler effect and the amount. The Doppler effect warps the pitch depending on distance. Think of a racing car passing by - hear the engine's pitch get higher as the car gets closer and lower as it gets farther away. Use this effect for fast moving objects such as bullets, pass-by cars, etc.
  - **Volume Rolloff**: sets volume attenuation depending on the distance to the listener. Use the animation curve to set your custom attenuation curve. When empty, it will use Unity's logarithmic curve.
  - **Max Distance**: sets max distance of the volume rolloff.

- **Effects**: Audio Event also has an effects section that can interact with other scripts for Real Time Parameter Change (RTPC). Currently there is a low pass filter, fade in, and reverb zone mix.

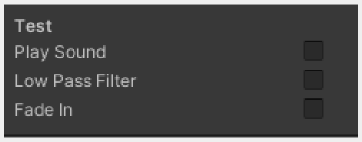


Effects

| | |
|---|---|
| Low Pass Filter Target Frequency | 1000 |
| Low Pass Filter Transition Time | 0 |
| Filter Reset Transition Time | 0 |
| Fade In Time | 0 |
| Reverb Zone Mix | 0 |

  - **Low Pass Filter Target Frequency**: sets target frequency for the low pass filter effect. The lower it is, the more muffled it will sound.
  - **Low Pass Filter Transition Time**: sets how long the low pass filter effect will take to be in effect.
  - **Filter Reset Transition Time**: transition time back to no filter effect.
  - **Fade In Time**: sets duration of fade in time in seconds. For example, fade in time is set to 5 seconds; it will take 5 seconds to reach the max volume.
  - **Reverb Zone Mix**: sets reverb zone mix level. 0 is dry and 1 is wet.

- **Test**



Test

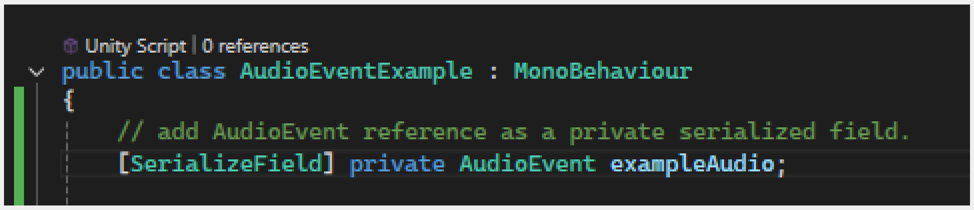
| | |
|---|---|
| Play Sound | ☐ |
| Low Pass Filter | ☐ |
| Fade In | ☐ |

  - **Play Sound**: click to play sound during runtime. Useful to do quick tests without switching.
  - **Low Pass Filter**: low pass filter test button. Also interactable with other scripts.
  - **Fade In**: fade in test button. Also interactable with other scripts.

## How to reference Audio Event in scripts

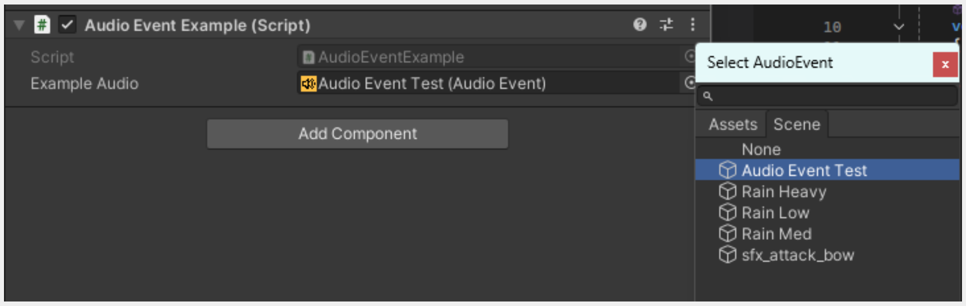
Audio Event is a public class that can be referenced in other scripts. Follow these steps to add Audio Event to your game object. Step 2 and 3 can be in opposite orders.

1. Set a private variable from the Audio Event class. Specify the audio name.



```
Unity Script | 0 references
public class AudioEventExample : MonoBehaviour
{
    // add AudioEvent reference as a private serialized field.
    [SerializeField] private AudioEvent exampleAudio;
```

2. Click on the game object and select the corresponding Audio Event object.

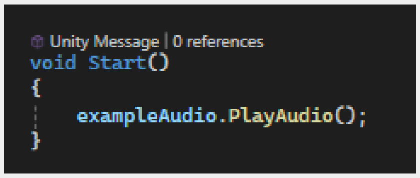


Audio Event Example (Script)

Script: AudioEventExample

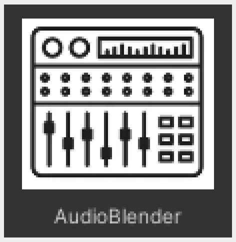
Example Audio: Audio Event Test (Audio Event)

3. Add ***[audio name].PlayAudio();*** where the audio clip should play in response of.



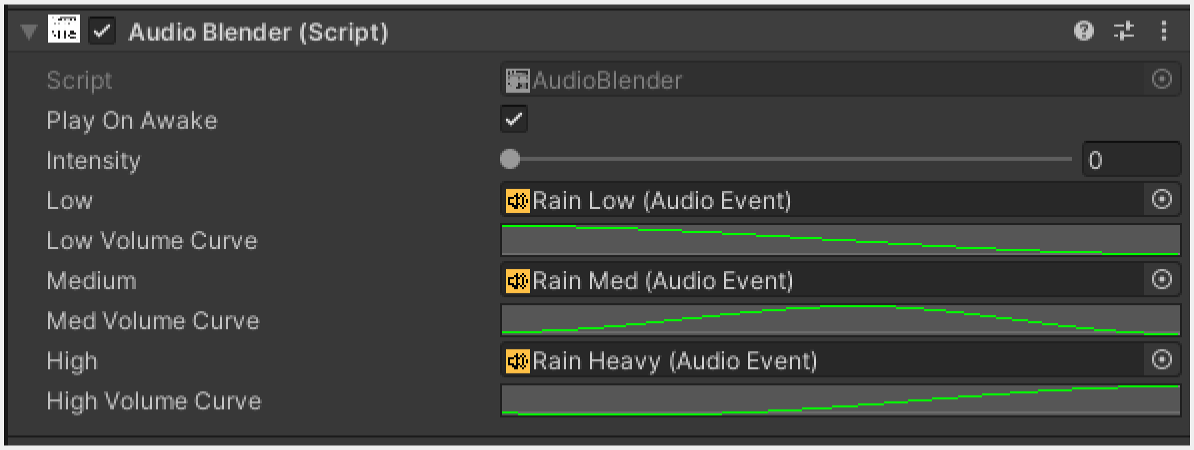
```
Unity Message | 0 references
void Start()
{
    exampleAudio.PlayAudio();
}
```

# [Audio Blender]



## Overview

Audio Blender is a custom script that blends three audio clips depending on the intensity level. It's designed to control **volumes** of the three separate Audio Event objects. Each Audio Event object has an animation curve attached that can be configured individually.



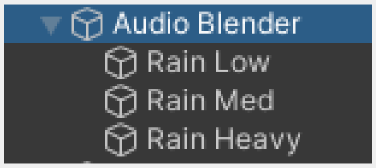
Audio Blender (Script)

| | |
|---|---|
| Script | AudioBlender |
| Play On Awake | ✓ |
| Intensity | 0 |
| Low | Rain Low (Audio Event) |
| Low Volume Curve | |
| Medium | Rain Med (Audio Event) |
| Med Volume Curve | |
| High | Rain Heavy (Audio Event) |
| High Volume Curve | |

**Demo**: https://youtu.be/roxMW1i4DdI

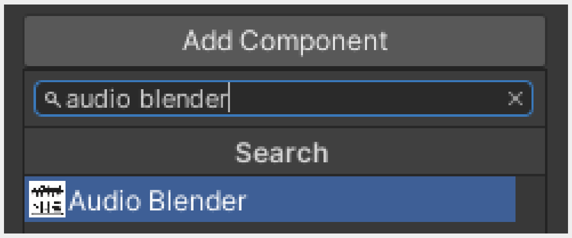
## How to set up Audio Blender

1. Create an empty object that nests three different intensities.

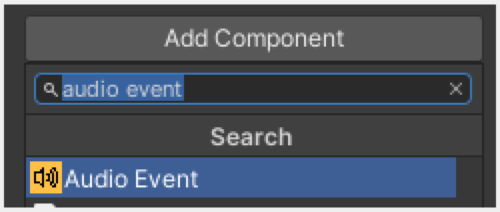


```
Audio Blender
    Rain Low
    Rain Med
    Rain Heavy
```

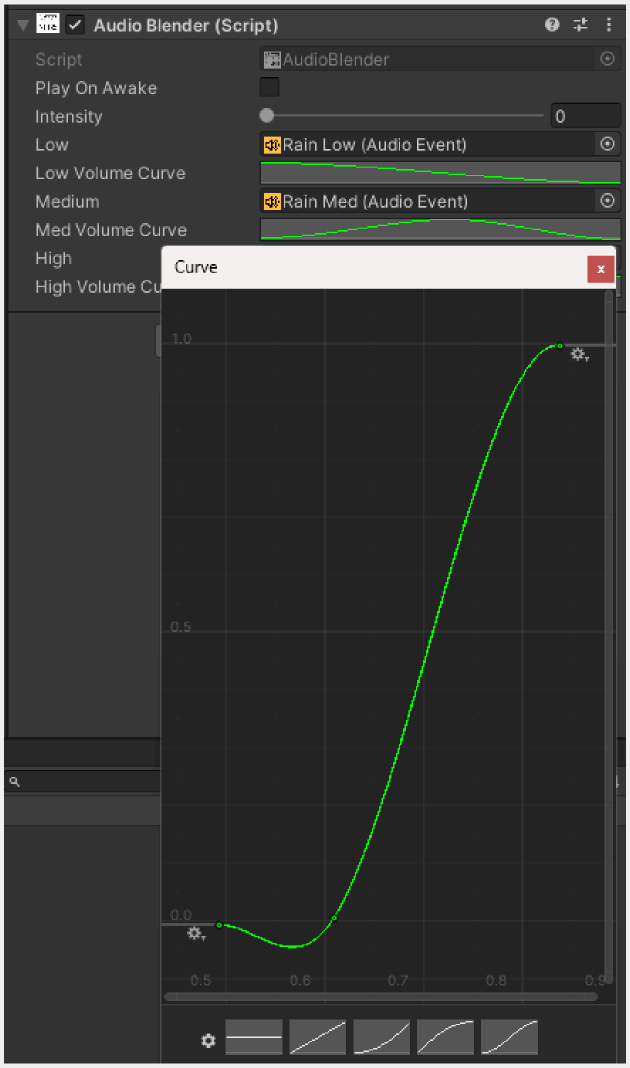
2. Add AudioBlender.cs component in the parent object.



3. Add AudioEvent.cs components in the children objects.



4. Assign Corresponding Audio Events in the Audio Blender and set custom curves.

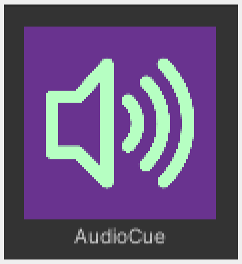


Script: AudioBlender
Play On Awake
Intensity: 0
Low: Rain Low (Audio Event)
Low Volume Curve
Medium: Rain Med (Audio Event)
Med Volume Curve
High
High Volume Cu

**Potential Use Cases:**

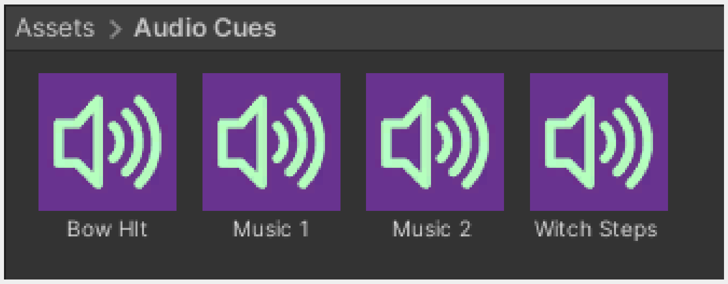
- Weather system
- Car engine
- Dynamic Music System

# [Audio Cue]



## Overview

Audio Cue uses Unity's Scriptable Object class to store audio data as **assets,** independent of game objects. Scriptable objects are great for storing and referencing unchanging data - and luckily most audio data don't need to change during runtime. The data stored in Scriptable Objects also persist throughout the project as opposed to game objects being dependent on the scenes. Optimization wise, there isn't a significant difference between using monobehaviour vs. scriptable objects for your audio system - especially when the project is small. However, it's nice to be able to store audio data in an asset format which improves organization and workflow in Unity.

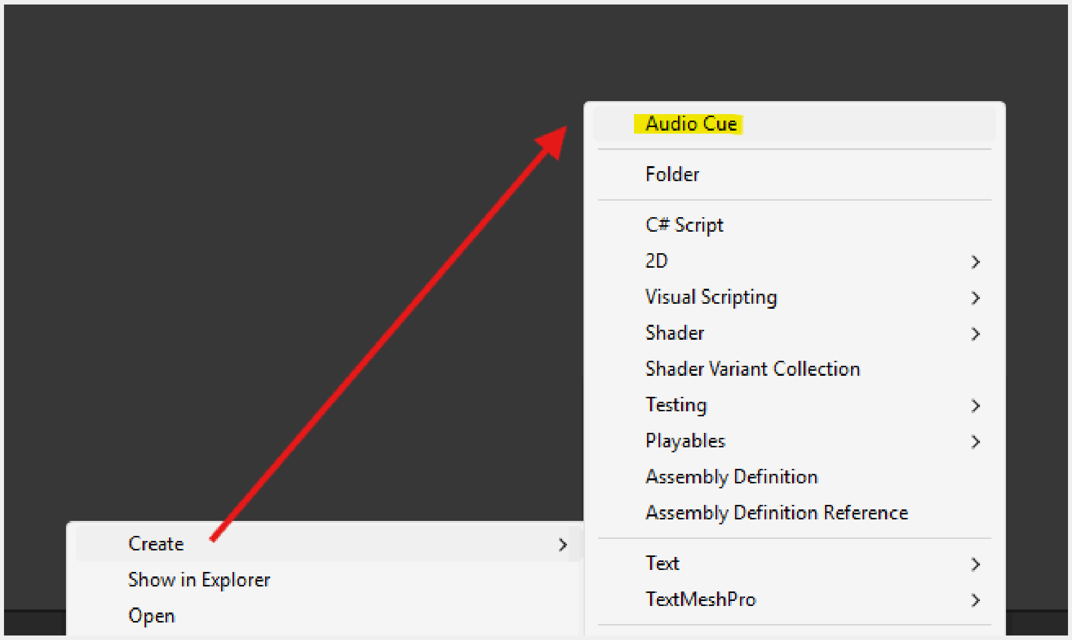


*Audio Cue Assets*

Audio Cue also features **different play types**, **randomizing start point**, and **random pitch and volume.**

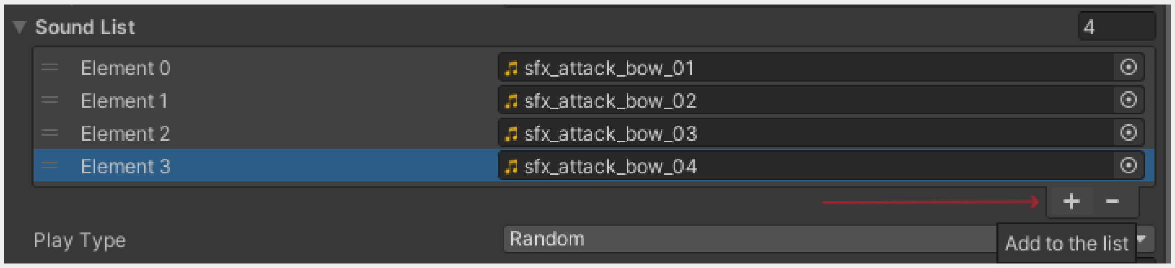
## How to use Audio Cue

1. Once you load AudioCue.cs into your project, you will be able to create an Audio Cue asset by right clicking an empty space in the asset folder - Create - Audio Cue.



2. Name your Audio Cue. I recommend using the same naming convention as the audio clips it will contain.

3. Add audio clips by clicking on the “+” sign.



Sound List 4

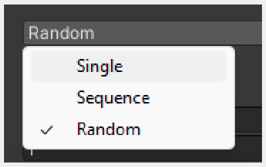
| | |
|---|---|
| Element 0 | sfx_attack_bow_01 |
| Element 1 | sfx_attack_bow_02 |
| Element 2 | sfx_attack_bow_03 |
| Element 3 | sfx_attack_bow_04 |

Play Type Random

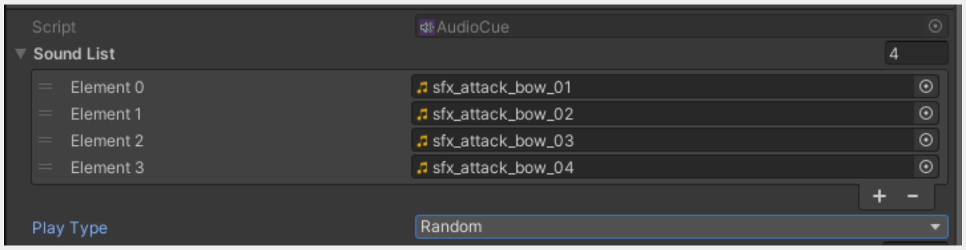
4. Configure parameters. See below for details.

### Play Types and when to use them

There are three different play types in Audio Cue. Choose appropriate play types for different sound effects.

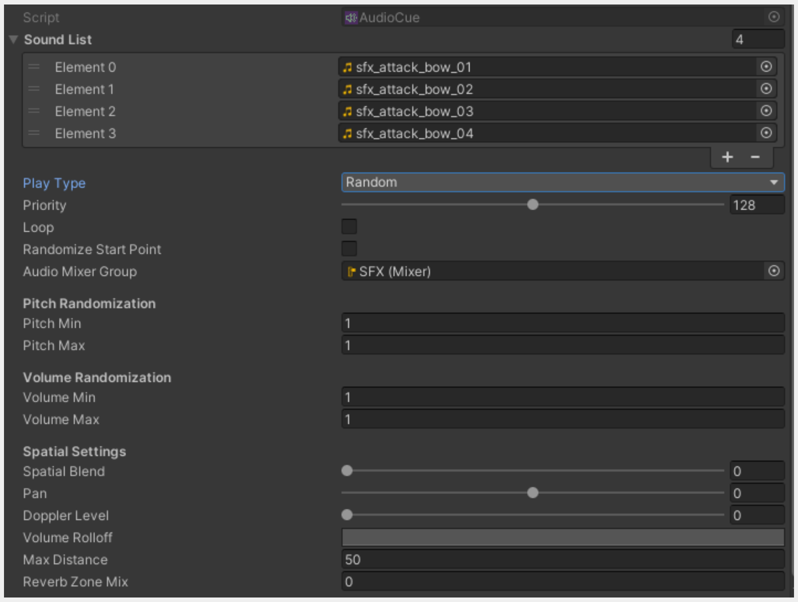


- **Single**: when selected, it will only play the file in the Element 0 spot. This play type is recommended for longer files such as ambience or music.
- **Sequence**: use this play type for a series of sounds that need to play in sequence. In this example, it will play sfx_attack_bow_01 then 02, 03, 04 in order; and loop back to the top of the list and repeat in order.
- **Random**: when selected, it will choose a clip in the Sound List. Next time it's triggered, it selects a different clip than the last played clip. Use this play type for short sounds that need to repeat a lot such as footsteps, physics impacts, attack sounds, and so on. Randomizing clips will help prevent ear fatigue and improve immersion.



| | |
|---|---|
| Script | AudioCue |
| Sound List | 4 |
| Element 0 | sfx_attack_bow_01 |
| Element 1 | sfx_attack_bow_02 |
| Element 2 | sfx_attack_bow_03 |
| Element 3 | sfx_attack_bow_04 |
| Play Type | Random |

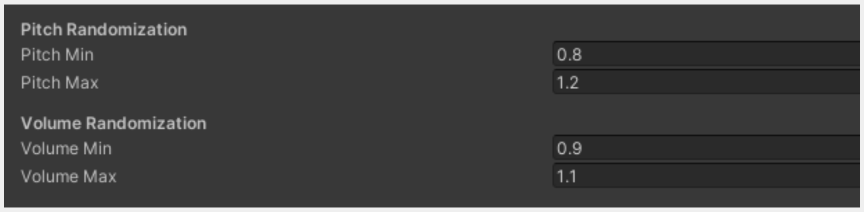
## Other Features



| | |
|---|---|
| Script | AudioCue |
| Sound List | 4 |
| Element 0 | sfx_attack_bow_01 |
| Element 1 | sfx_attack_bow_02 |
| Element 2 | sfx_attack_bow_03 |
| Element 3 | sfx_attack_bow_04 |
| Play Type | Random |
| Priority | 128 |
| Loop | |
| Randomize Start Point | |
| Audio Mixer Group | SFX (Mixer) |
| **Pitch Randomization** | |
| Pitch Min | 1 |
| Pitch Max | 1 |
| **Volume Randomization** | |
| Volume Min | 1 |
| Volume Max | 1 |
| **Spatial Settings** | |
| Spatial Blend | 0 |
| Pan | 0 |
| Doppler Level | 0 |
| Volume Rolloff | |
| Max Distance | 50 |
| Reverb Zone Mix | 0 |

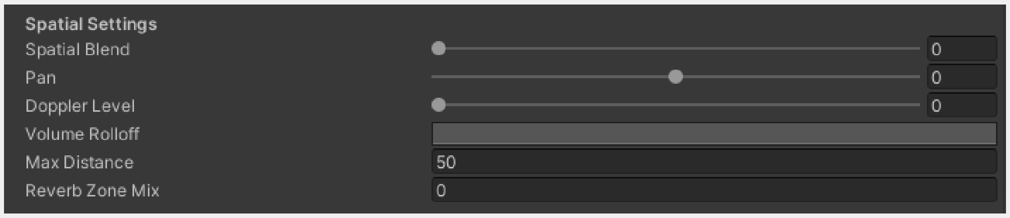
- **Loop**: check when the audio file needs to loop. Only available for Single Play Type.

- **Randomize Start Point**: selects a random start point in the audio clip. It works with short sounds as well but longer sounds would benefit more for this feature. e.g.) leaf rustling, rain ambience, etc.
- **Pitch**: sets base pitch.
- **Volume**: sets base volume.
- **Priority**: sets priority of the audio object. This feature can be used to build a dynamic mixing system or to implement a volume ducking feature. 0 is the highest priority and 256 is lowest.
- **Audio Mixer Group**: sets audio mixer group (which route to pass the audio through).
- **Randomization**: a useful feature to give more variations to the sounds. Experiment with different values for pitch and volume.
    - Set minimum and maximum value for each pitch and volume. When the audio is triggered, it will play the clip at a random pitch and volume value within the range you set.



Pitch Randomization
Pitch Min 0.8
Pitch Max 1.2
Volume Randomization
Volume Min 0.9
Volume Max 1.1

- **Spatial Settings**

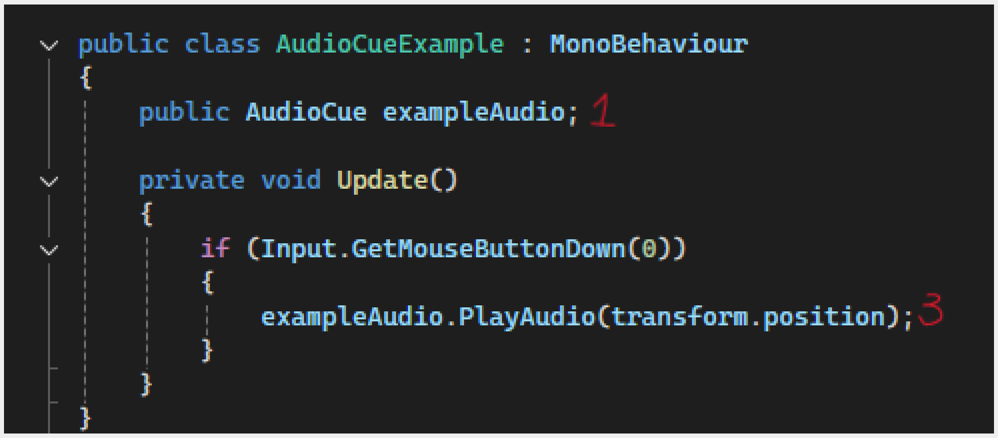


Spatial Settings
Spatial Blend 0
Pan 0
Doppler Level 0
Volume Rolloff
Max Distance 50
Reverb Zone Mix 0

    - **Spatial Blend**: sets 2D or 3D attenuation. 2D at 0 and 3D at 1.
    - **Pan**: sets which side to play the audio. Left at -1, right at 1, center at 0.
    - **Doppler Level**: enables doppler effect and the amount. The Doppler effect warps the pitch depending on distance. Think of a racing car passing by - hear the engine's pitch get higher as the car gets closer and lower as it gets farther away. Use this effect for fast moving objects such as bullets, pass-by cars, etc.
    - **Volume Rolloff**: sets volume attenuation depending on the distance to the listener. Use the animation curve to set your custom attenuation curve. When empty, it will use Unity's logarithmic curve.
    - **Max Distance**: sets max distance of the volume rolloff.
    - **Reverb Zone Mix**: sets wet level for the reverb zone.

## How to reference Audio Cue in scripts

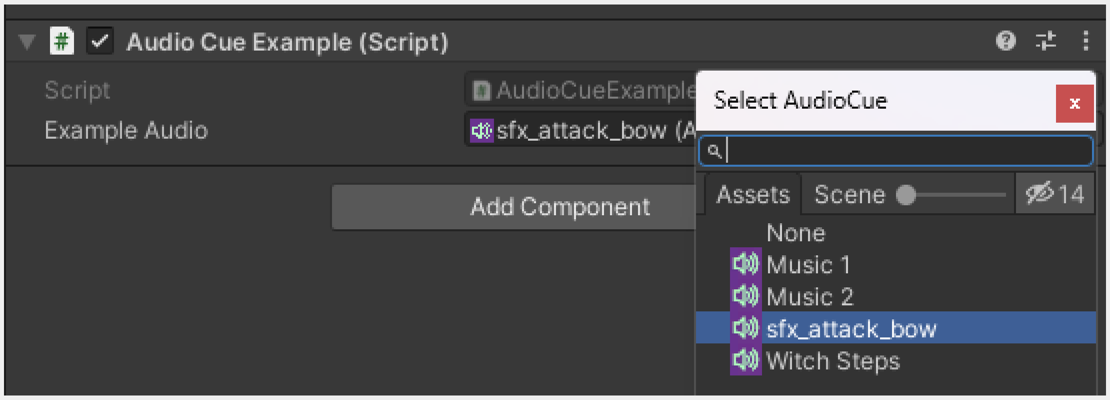
Audio Cue is a public class that can be referenced in other scripts. Follow these steps to activate Audio Cue in your script. Step 2 and 3 can be in opposite orders.



```
public class AudioCueExample : MonoBehaviour
{
    public AudioCue exampleAudio; 1

    private void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            exampleAudio.PlayAudio(transform.position);3
        }
    }
}
```

1. Set a private variable from the Audio Cue class. Specify the audio name for clarity.
2. Click on the game object and select the corresponding Audio Cue.



3. Add ***[audio name].PlayAudio(transform.position);*** where the audio clip should play in response of.