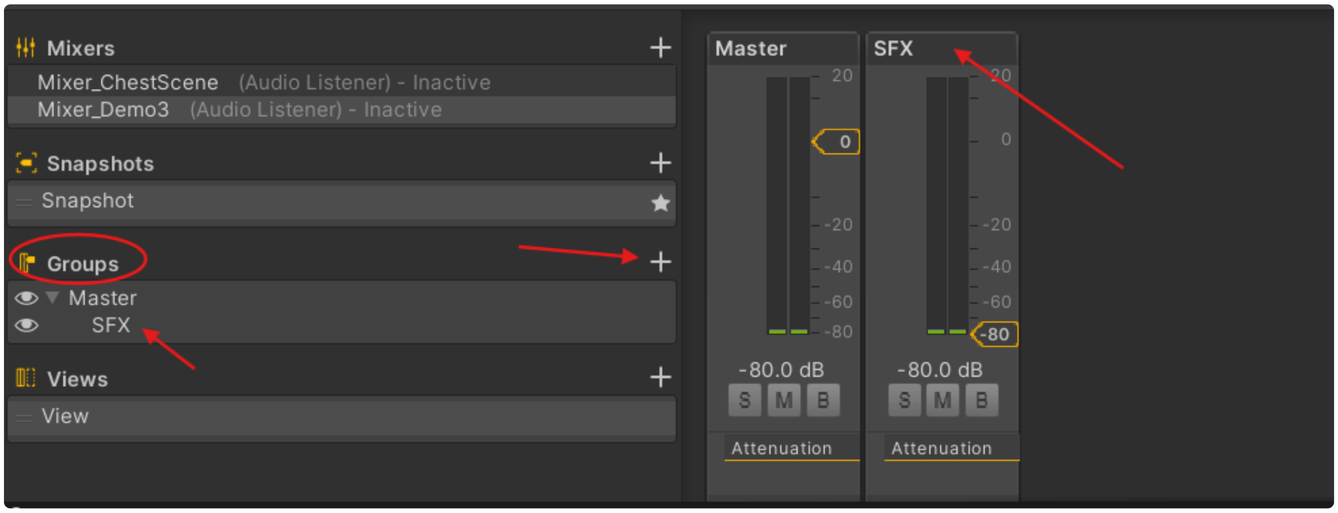
Dynamic Audio Mixing and Animation Events in Unity

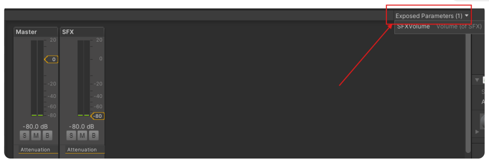
Part 1: Project Setup and Audio Mixer Creation

1. Create folder structure:
   - In the Project window, right-click and select `Create > Folder`
   - Name the folder "Audio"
   - Right-click on the Audio folder and select `Create > Folder`
   - Name this subfolder "Mixers"
2. Create an Audio Mixer:
   - Right-click on the Mixers folder
   - Select `Create > Audio Mixer`
   - Name it "GameAudioMixer"
3. Open the Audio Mixer:
   - Double-click on GameAudioMixer in the Project window
4. Create a group for your sound effect:
   - In the `Audio Mixer` window, find the `Groups` sections and
   - Click the `+` to create a new group
   - Name it "SFXGroup"



1. Expose the SFXGroup volume parameter:
   - In the `Audio Mixer` window, select the SFXGroup
   - In the `Inspector`, find the Volume slider
   - Right-click on "Volume" label next to the slider
   - Choose `Expose Parameter`
   - In the Exposed Parameters list (located at the top right of Audio Mixer window), double-click the newly created parameter

- Rename it to "SFXVolume" and hit `Enter` to confirm



> **Note**
>
> The Volume effect exists by default on each group in the Audio Mixer. We're just exposing it for script control.

Part 2: Creating the Audio Control Script

1. Create a Scripts folder:
   - In the Project window, right-click and select `Create > Folder`
   - Name the folder "Scripts"
2. Create a new C# script:
   - Right-click on the Scripts folder
   - Select `Create > C# Script`
   - Name it "AudioController"
3. Open the script:
   - Double-click on AudioController.cs to open it in your code editor
4. Replace the default code with the following:

```
using UnityEngine;
using UnityEngine.Audio;

public class AudioController : MonoBehaviour
{
    public AudioMixer audioMixer;
    public string volumeParameter = "SFXVolume";
    public float fadeDuration = 1.0f;

    private float startTime;
    private bool isFading = false;

    public void StartVolumeRise()
    {
        startTime = Time.time;
        isFading = true;
        Debug.Log("Starting volume rise");
    }

    void Update()
    {
        if (isFading)
        {
            float t = (Time.time - startTime) / fadeDuration;
            if (t > 1f)
            {
                t = 1f;
                isFading = false;
            }

            float currentVolume = Mathf.Lerp(-80f, 0f, t);
            audioMixer.SetFloat(volumeParameter, currentVolume);
            Debug.Log("Current volume: " + currentVolume);
        }
    }
}
```

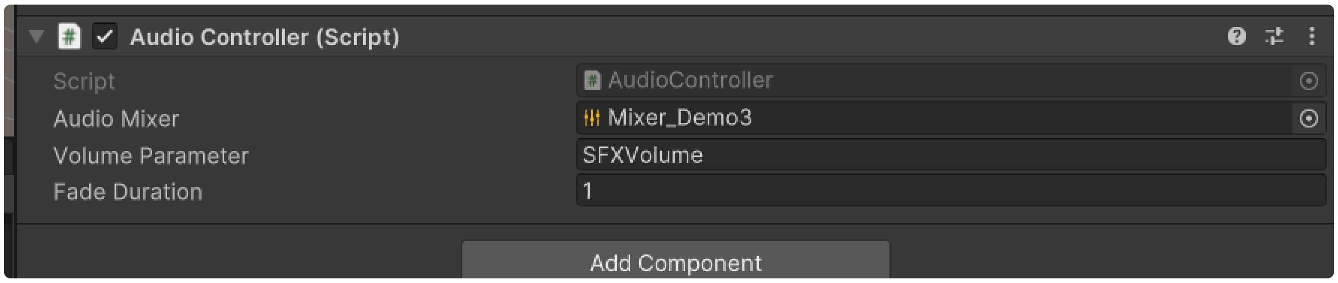
5. Save the script:
    - File > Save (or Ctrl+S / Cmd+S)

Tip

The Debug.Log statements will help you see when the volume rise starts and what the current volume is in the console window.

Part 3: Setting Up the Audio Manager GameObject

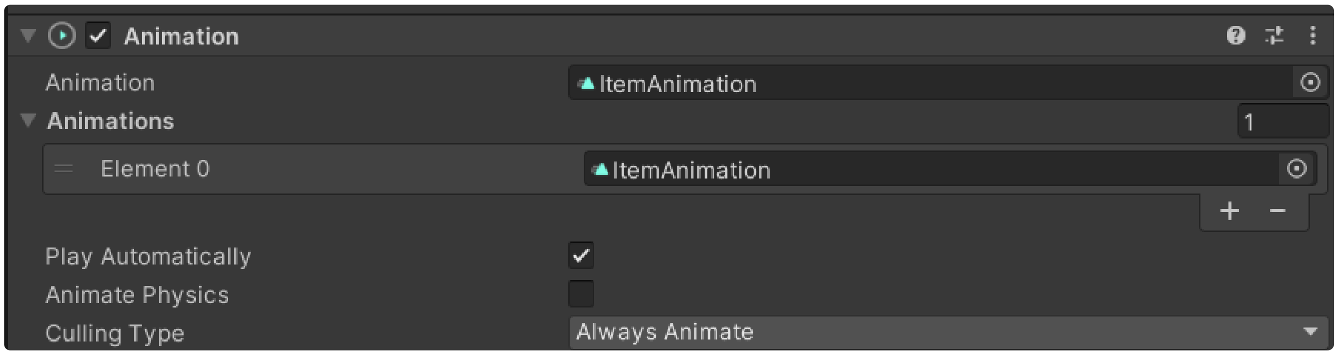
1. Create an empty GameObject:
    - In the Hierarchy window, right-click and select `Create Empty`
    - Name it "AudioManager"
2. Add the AudioController script:
    - Select the AudioManager GameObject in the Hierarchy
    - In the `Inspector`, click `Add Component`
    - Type "AudioController" and select it when it appears
3. Assign the Audio Mixer:
    - In the Project window, find your GameAudioMixer in the Audio/Mixers folder
    - Drag it into the Audio Mixer field in the AudioController component in the Inspector
4. Verify settings:
    - Ensure "Volume Parameter" is set to "SFXVolume"
    - You can adjust "Fade Duration" if desired (default is 1 second)


Audio Controller (Script)

Script: AudioController
Audio Mixer: Mixer_Demo3
Volume Parameter: SFXVolume
Fade Duration: 1

Part 4: Creating an Animated Object

1. Create a 3D object:
   - In the Hierarchy, right-click and select `3D Object > Cube`
   - Name it "AnimatedItem"
2. Add an Animator component:
   - With AnimatedItem selected, in the `Inspector`, click `Add Component`
   - Type "Animation" and select it
3. Create an Animations folder:
   In the Project window, right-click and select `Create > Folder`
   Name it "Animations"
4. Create a new Animation:
   - Right-click on the Animations folder
   - Select `Create > Animation`
   - Name it "ItemAnimation"
5. Assign the animation to the Animation:
   - Select AnimatedItem in the Hierarchy
   - In the Inspector, find the Animator component
   - Drag ItemAnimation from the Project window to the Animation field in the Animator component
6. Open the Animation window:
   - Go to `Window > Animation > Animation`
     - You can also access this window by double-clicking the ItemAnimation asset in the Assets > Animations folder


Animation

Animation: ItemAnimation
Animations: 1
Element 0: ItemAnimation
Play Automatically: ✓
Animate Physics:
Culling Type: Always Animate

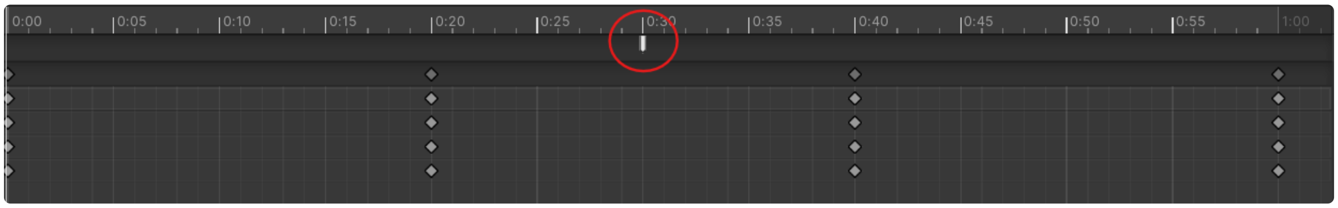
7. Ensure you're editing the right animation:
   - In the Animation window, make sure "ItemAnimation" is selected in the dropdown

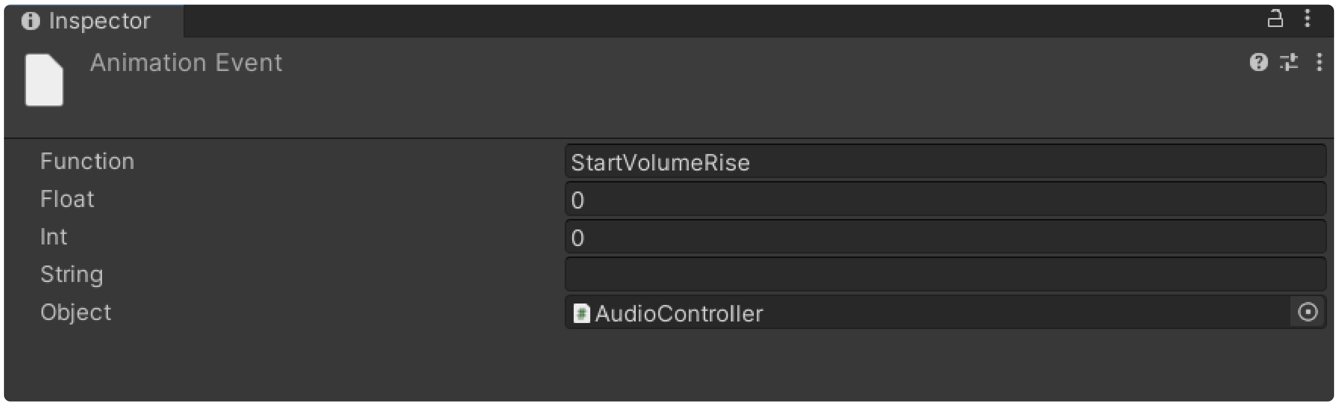
8. Create a simple animation:
   - Ensure AnimatedItem is selected in the Hierarchy
   - You may need to create a new `Clip`. When prompted click the `Create New Clip..`. button. Search for your ItemAnimation.anim asset and select it, when prompted, Overwrite it.
   - In the Animation window, click Add Property (This is only selectable when the Animated Item object is selected)
   - Expand Transform and select Position by clicking the `+`
   - Create keyframes by adjusting the X, Y, and/or Z positions at different points in the timeline

> **Note**
>
> This will create a simple up-and-down animation. Feel free to make it more complex if desired.

Part 5: Adding an Animation Event

1. Open the Animation window with your ItemAnimation
2. Add an event:
   - In the Animation timeline, right-click at the point where you want the sound to start
   - Select `Add Animation Event`
3. Configure the event:
   - In the Inspector, you'll see the Animation Event properties
   - For `Object`, search for and select the AudioController script
   - For Function, type "StartVolumeRise"





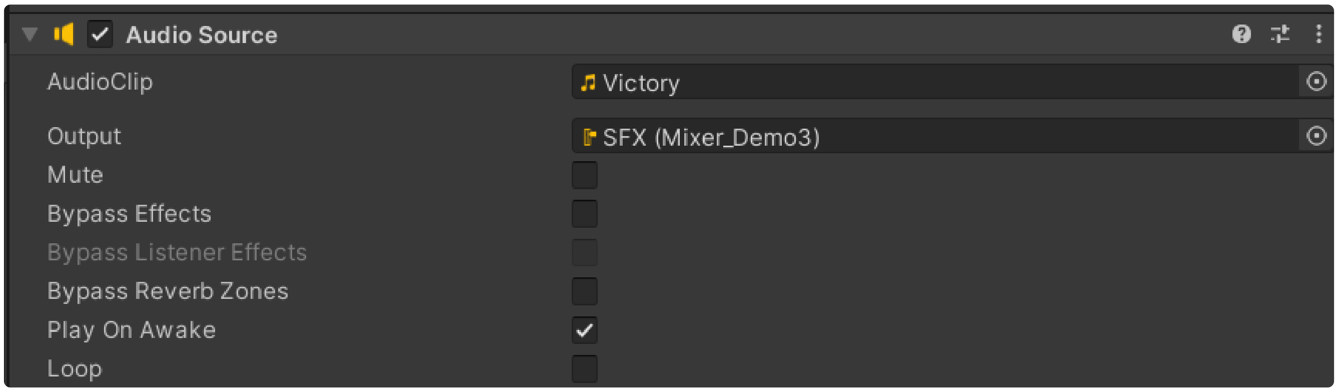
Function	StartVolumeRise
Float	0
Int	0
String	
Object	AudioController

> **Warning**

The `Function` parameter name is the method of the same name in the `AudioController.cs` script. When using Function calls in animation events spelling and punctuation matters; the names need to match exactly.

Part 6: Testing Your Setup

1. Ensure an audio source is set up:
   - Add an Audio Source component to AnimatedItem
   - Assign a sound file to the Audio Source
   - In the Audio Source component, set Output to SFXGroup



Audio Source

AudioClip	Victory
Output	SFX (Mixer_Demo3)
Mute	☐
Bypass Effects	☐
Bypass Listener Effects	☐
Bypass Reverb Zones	☐
Play On Awake	✓
Loop	☐

2. Set initial volume:
   - In the Audio Mixer, set SFXGroup volume to -80 dB
3. Play the scene:
   - Click the Play button at the top of the Unity editor
4. Observe the result:
   - Watch the AnimatedItem move
   - When the animation event triggers, you should hear the volume increase
   - Check the Console window for debug messages showing the volume change

Tip

If you don't hear anything, check that your sound file is playing and that the Audio Source is properly assigned to SFXGroup. Also, be sure that the sound selected for playback is longer than the fade amount set on the script (which defaults to 1 second).

Part 7: Troubleshooting a Common Animation Event "Gotcha"

It's not working...

No sound plays and I get an error!

After following the previous steps, you may encounter this error:

```
'Animated Item' AnimationEvent 'StartVolumeRise' has no receiver! Are you missing a
component?
```

Learning Opportunity

I've intentionally led you into a ***very common*** mistake regarding animation events and scripting.

Understanding the Problem

1. The issue:
   - Animation Events can only call methods on components attached to the ***same GameObject*** as the Animator/Animation(s).
   - Our AudioController is on a separate GameObject (AudioManager), not on AnimatedItem.

Solving the Problem

We'll solve this by creating a "bridge" script that connects our AnimatedItem to the AudioController.

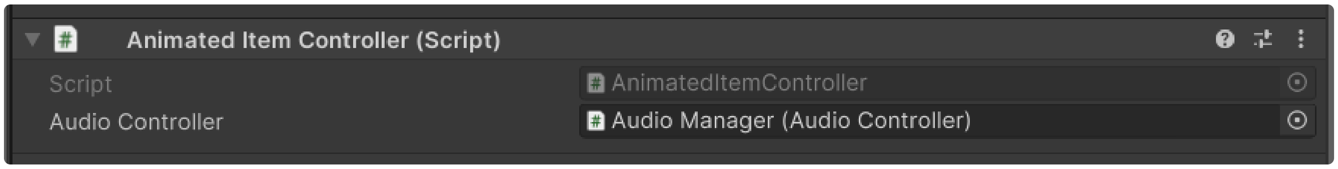
1. Create a new script:
   a. In the Project window, navigate to your Scripts folder
   b. Right-click, choose `Create > C# Script`
   c. Name it "AnimatedItemController"
2. Edit the AnimatedItemController script:

```
using UnityEngine;

public class AnimatedItemController : MonoBehaviour
{
    public AudioController audioController;

    public void TriggerVolumeRise()
    {
        if (audioController != null)
        {
            audioController.StartVolumeRise();
            Debug.Log("Volume rise triggered from AnimatedItem");
        }
        else
        {
            Debug.LogError("AudioController reference is missing!");
        }
    }
}
```

3. Set up the AnimatedItemController:
    - Select AnimatedItem in the Hierarchy
    - Add the AnimatedItemController component
    - Drag the AudioManager object to the Audio Controller field in the Inspector
4. Update the Animation Event:
    - Open the Animation window
    - Select the existing animation event
    - In the `Inspector`, change the Function to `AnimatedItemController > Methods > TriggerVolumeRise`


Animated Item Controller (Script)
Script: AnimatedItemController
Audio Controller: Audio Manager (Audio Controller)

Why This Solution Works

- The AnimatedItemController is on the same GameObject as the Animator, satisfying Unity's requirement.
- It acts as a "bridge," connecting the Animation Event to the AudioController on another GameObject.
- This pattern is commonly used to maintain a clean separation of concerns in larger projects.

Real-World Application

In game development, you'll often need to connect systems that aren't directly compatible. Creating "bridge" or "manager" scripts is a common and useful technique.

Lessons Learned

1. Always consider which GameObject owns an Event or Message.
2. Plan your game architecture to handle communication between different systems.
3. Use intermediate scripts to bridge gaps between components on different GameObjects.

4. Debug logs can be invaluable for tracking the flow of method calls in your game.

Test Again (now that it should all be connected)

3. Play the scene:
    - Click the Play button at the top of the Unity editor
4. Observe the result:
    - Watch the AnimatedItem move
    - When the animation event triggers, you should hear the volume increase
    - Check the Console window for debug messages showing the volume change