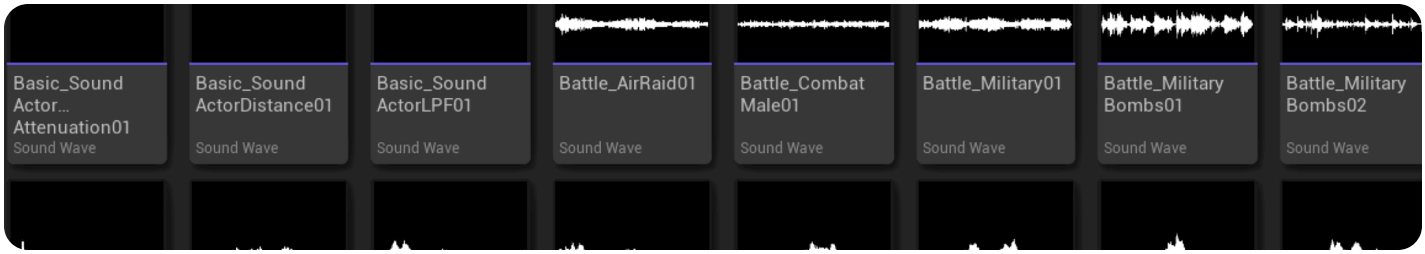
Audio Effects

How to play audio effects within a Niagara simulation



An audio effect is when you want to play some sound within a Niagara simulation. For example, when a particle simulation collides with something, then a sound plays. There are several ways to accomplish this within the Niagara system.

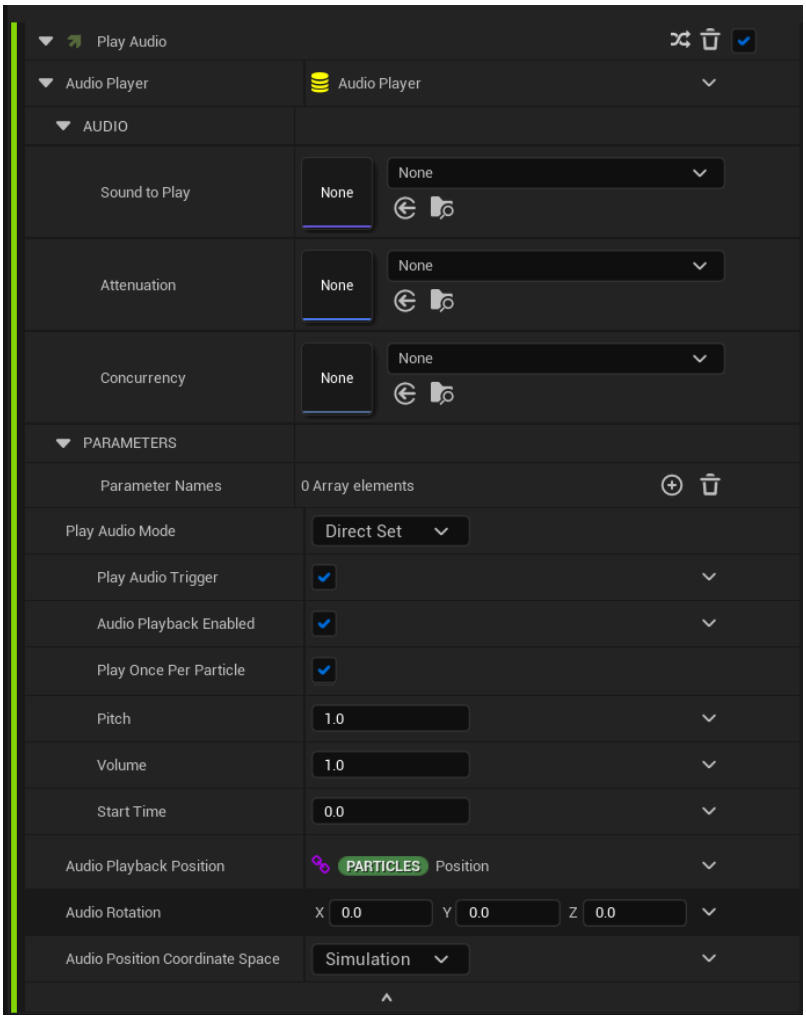
Each method has its pros and cons. Following is a brief overview of each.

> You can find samples of these methods in the Content Examples project on the **Epic Games Launcher: Unreal Engine > Learn > Engine Feature Samples > Content Examples**.

Using the Play Audio Module

For a one-off sound effect, the **Play Audio** module is the simplest way to play sound effects from Niagara. It is useful for event-driven effects, such as a reaction to a particle collision. It is also the cheapest solution in terms of memory usage.

1. Add the **Play Audio** module to your emitter. You can add this to the Particle Update group so that this module is evaluated as the particles age.



Click image for full size.

2. From **Sound to Play**, select a sound from the dropdown. If you have the Content Examples project open, there are already some sounds added to the project.
3. Set up the **Play Audio** condition.

   This fires a one-time effect using the configured values for pitch, volume, and more. Once an effect is triggered, it cannot be changed or stopped, and continues to play even if the particle simulation is stopped.

Play Audio Module

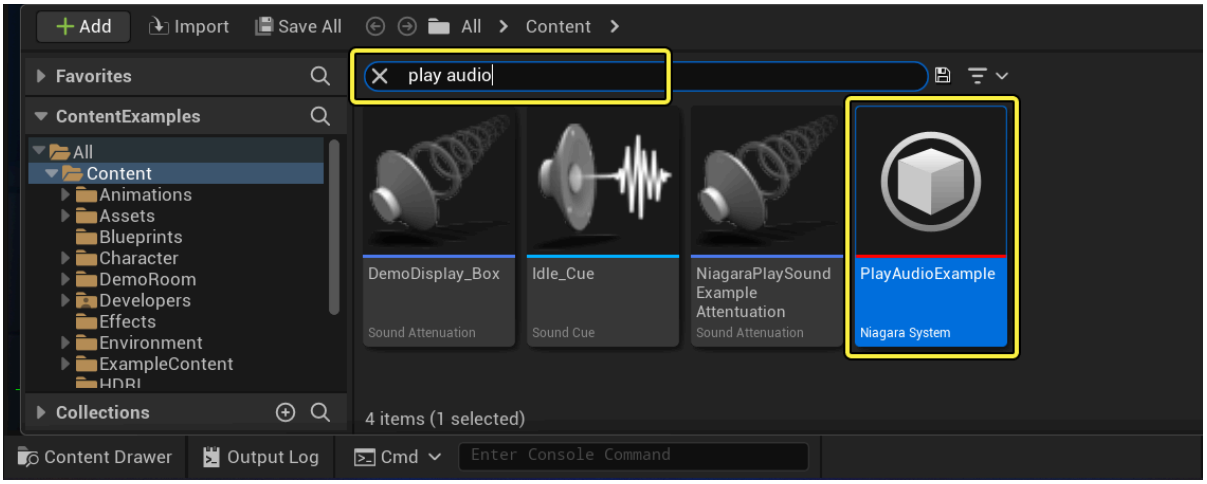
Pros	Cons
* Most performant solution (for both memory and CPU usage)	* Sound properties like volume or pitch are fixed once the sound starts playing

Play Audio Module

* Easiest to set up	* Sounds cannot update their positions over time as a particle moves

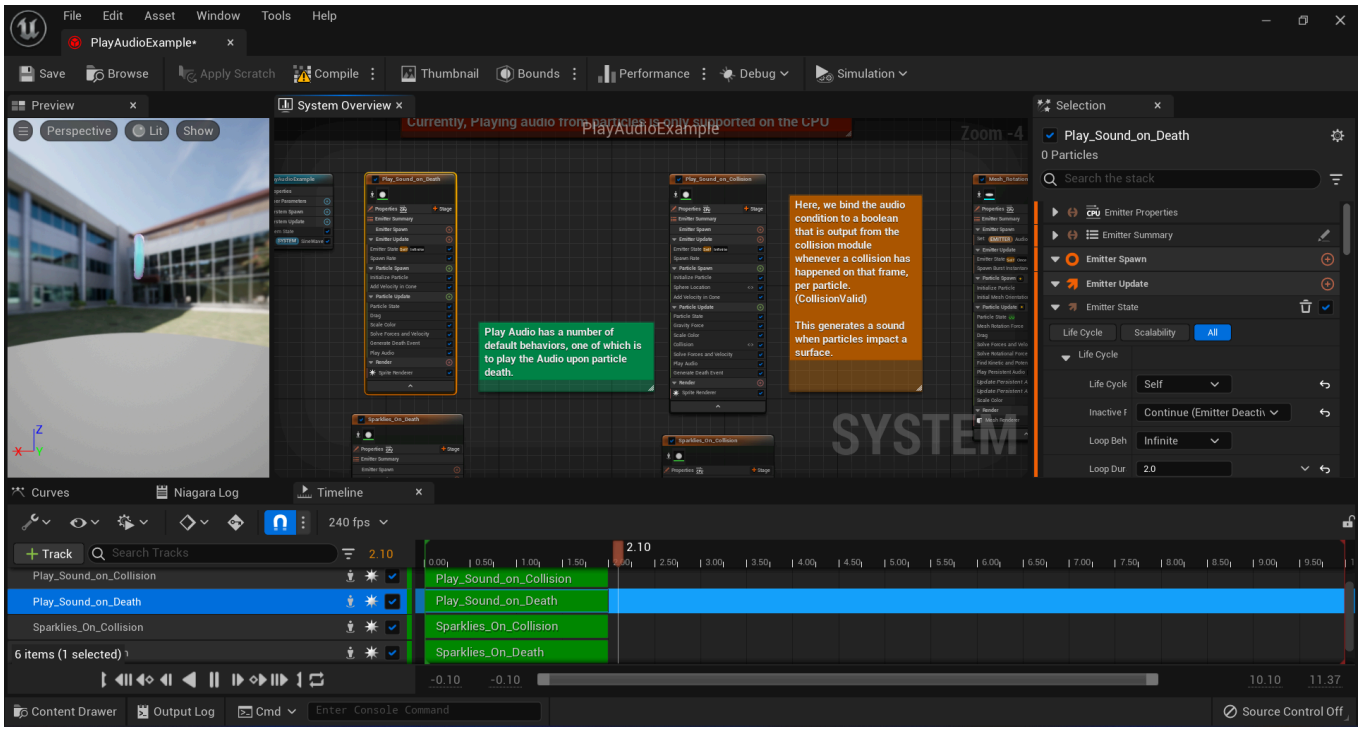
Audio Module Example

1. Download the Content Examples project, then open it.
2. In the Content Drawer, search for **Play Audio**.



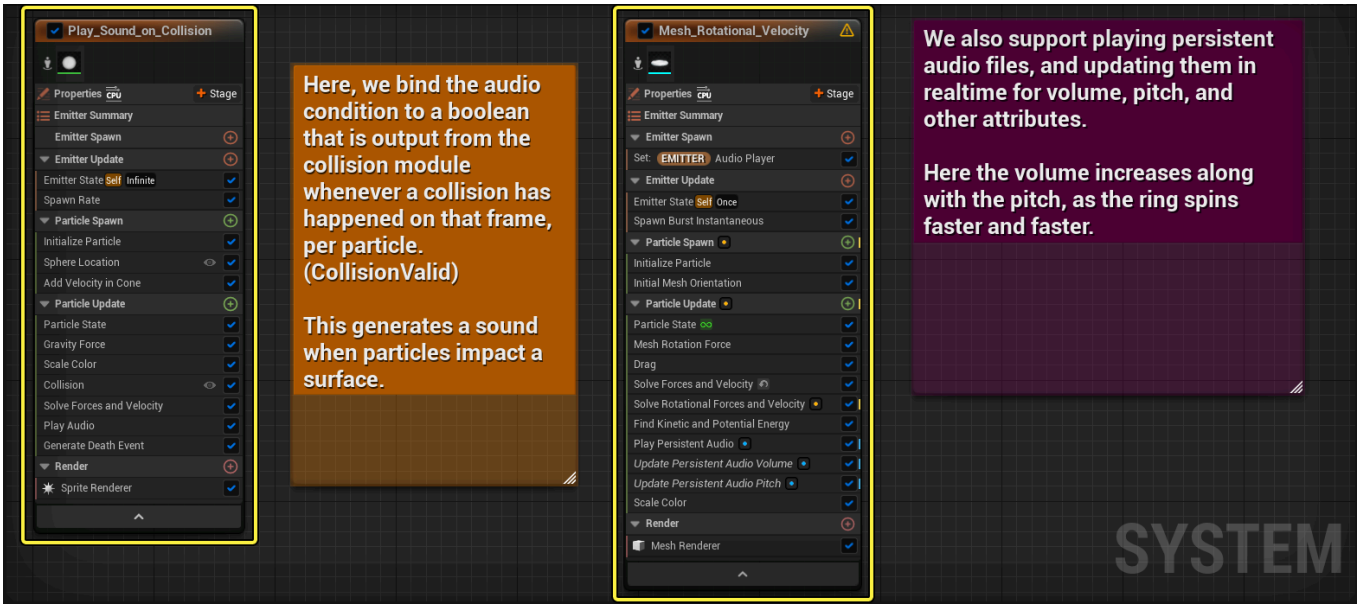
Click image for full size.

3. Double-click **PlayAudio Example** to open the **Niagara System** example.



Click image for full size.

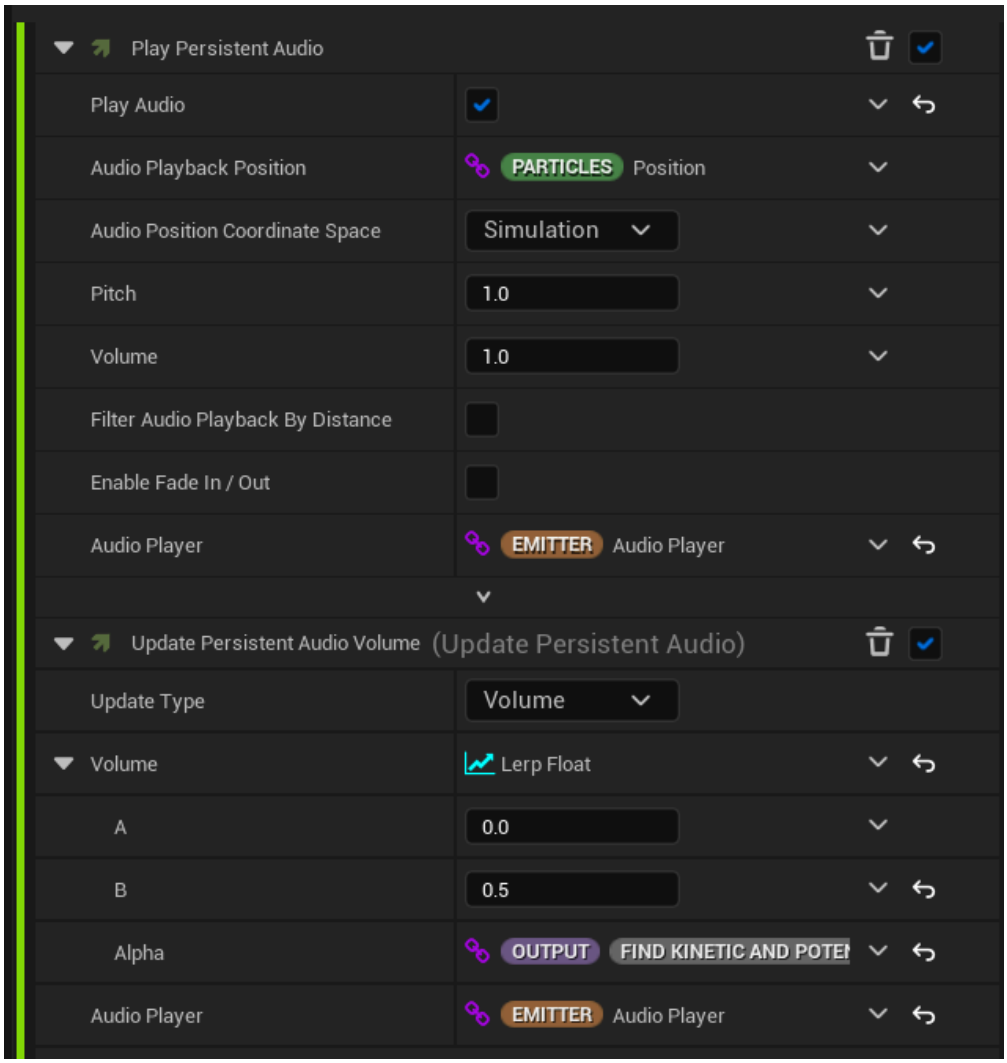
4. There are different types of audio examples in this project. You will find one emitter called **Play_Sound_On_Collision** that uses the **Play Audio module**. The emitter called **Mesh_Rotational_Velocity** uses a different method, the **Play Persistent Audio module**.



Click image for full size.

Using the Play Persistent Audio Module

This module is similar to Play Audio, but keeps a reference to each spawned sound effect so it can be updated over time.

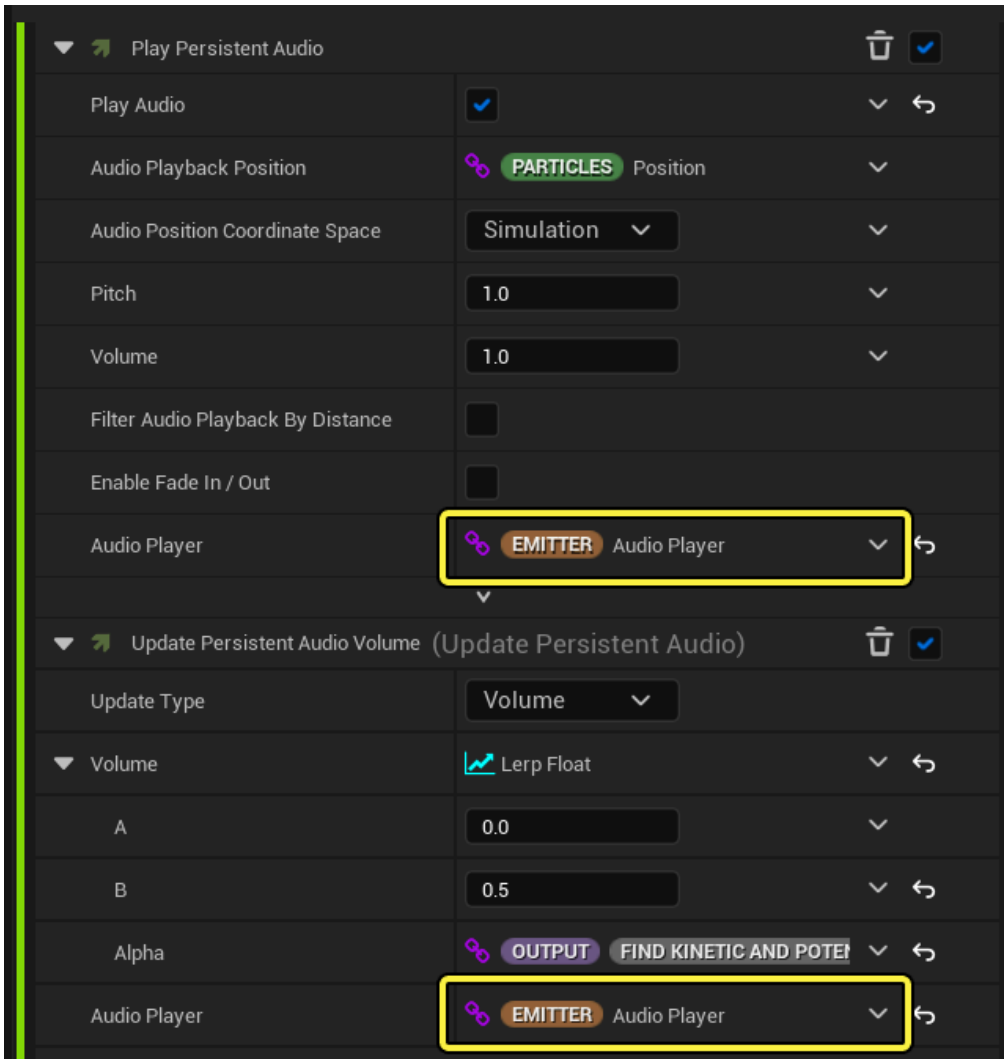


Click image for full size.

This module also allows for some advanced features, like fading or setting sound cue parameters. However, is a bit trickier to set up than the first method because to work, it requires two modules—**Play Persistent Audio** and **Update Persistent Audio**.

> Use the same audio player reference in both modules.

You can create an audio player data interface as an emitter attribute and bind it to the modules. In the screenshot above, the **Emitter.AudioPlayer** reference is in both modules.

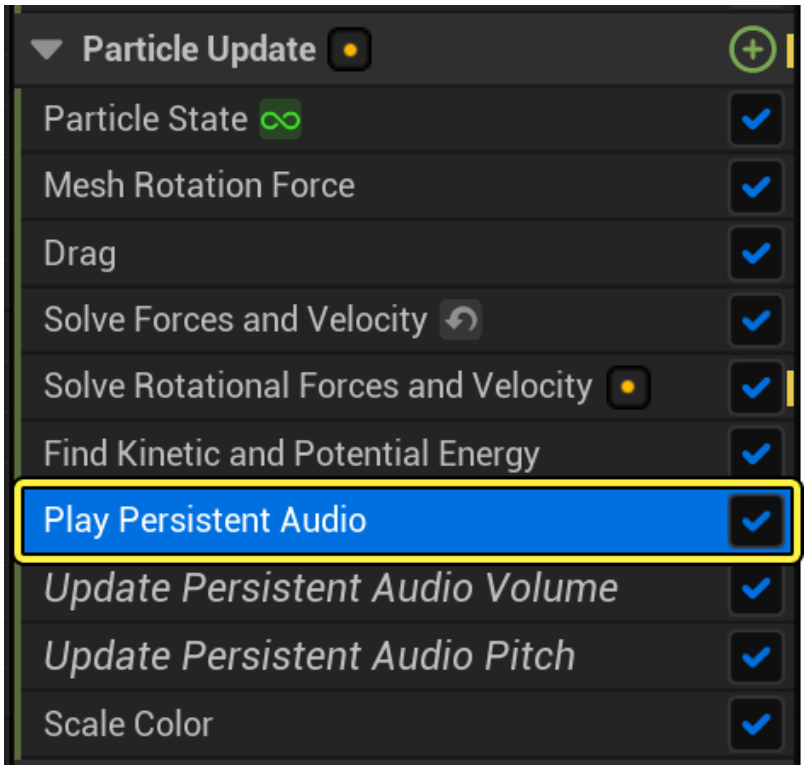


Click image for full size.

The chain symbol to the left of **EMITTER** indicates that they are referencing an existing attribute.

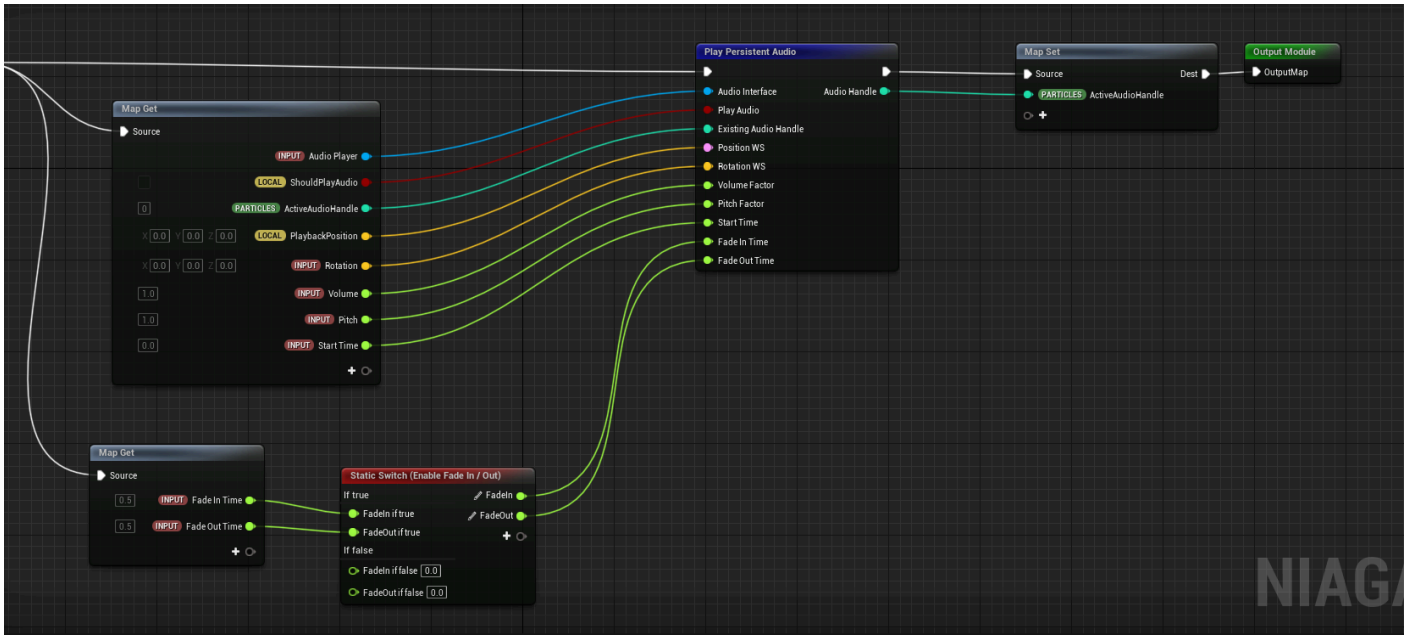
By default, each module creates its own object that cannot be shared with other modules. If you want the modules to use the same object, you must create it in the emitter or system spawn script, then reference it in the modules.

To see how the Play Persistent Audio module works, in the Content Samples file, from the **Content Drawer** search for **Play Audio**, then double-click **PlayAudio Example** to open. Locate the emitter called **Mesh_Rotational_Velocity**. You will find on this emitter a module called **Play Persistent Audio**.



Click image for full size.

Double-click **Play Persistent Audio** to open the **Node Graph**.



Click image for full size.

This method is useful when sound properties need to change during the simulation, such as when a sound needs to travel with the particle position.

Play Persistent Audio Module

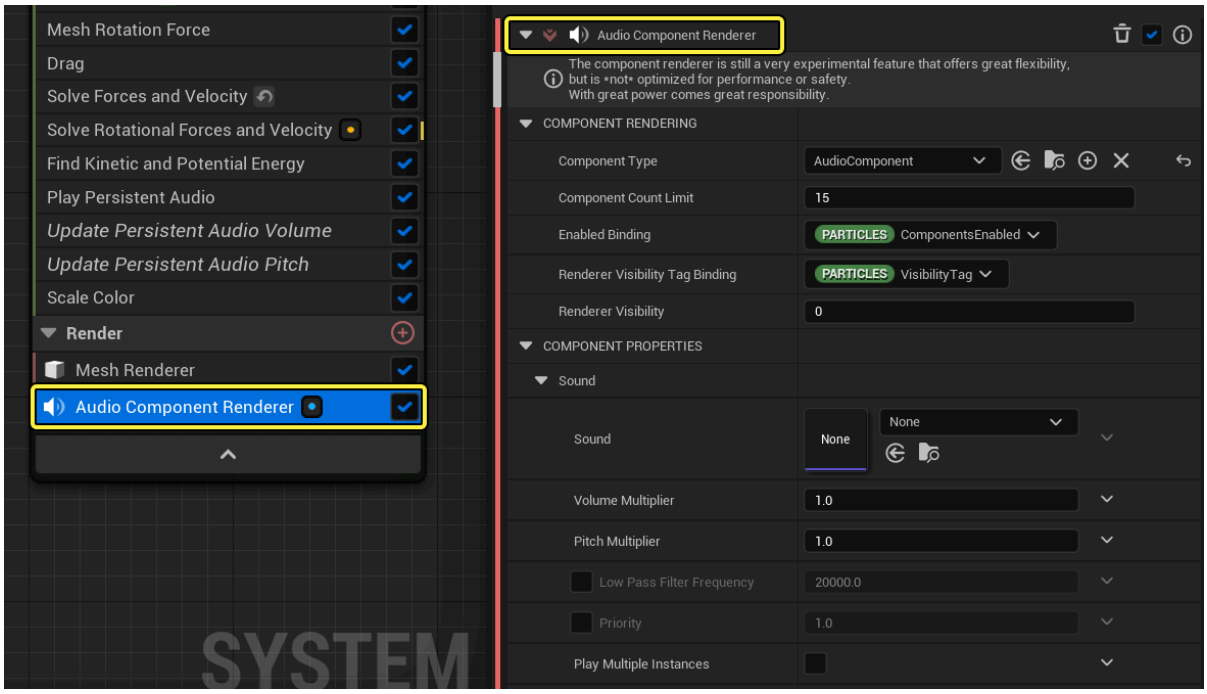
Pros	Cons
* Change volume, pitch, location and rotation at runtime	* More complex to set up
* Set sound cue parameters and end sound when simulation stops	* Less performant than Play Audio
* Filter sounds based on camera distance	

Using the Audio Component Renderer

> The Audio Component Renderer is experimental. Use it as you would any experimental feature in Unreal Engine.

The **Audio Component Renderer** can be used to spawn audio components. To add the Audio Component Renderer, click on the **Plus Sign (+)** in the **Render** group. Select **Component Renderer**.

Once the Component Renderer has been added, from the **Selection** panel, click the **Component Type** dropdown menu, then select **AudioComponent**.



Click image for full size.

This is a very flexible approach, since it gives full control over the audio component, but you would need to reimplement any features from the Play Audio modules in the particle simulation (such as fading or distance-based filtering). It also does not support setting sound cues without first creating a custom audio component subclass.

Aside from special use cases where the modules do not provide enough control, the component renderer is not recommended.

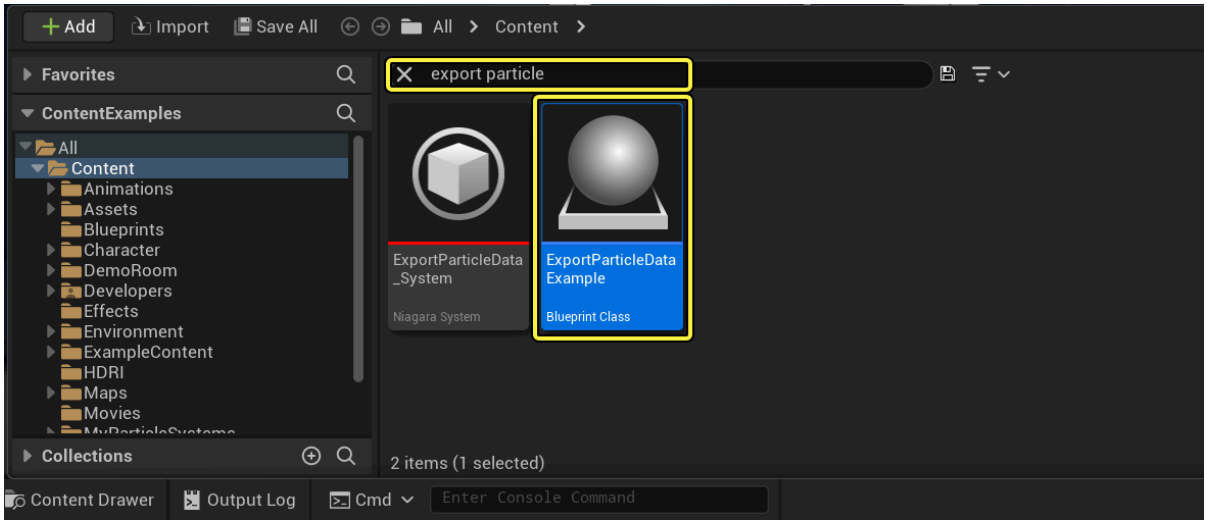
Audio Component Renderer

Pros	Cons
* Very flexible	* Missing many features
* Provides more control over audio	* Less performant
	* Cannot set sound cue parameters

Exporting Particle Data to Blueprint

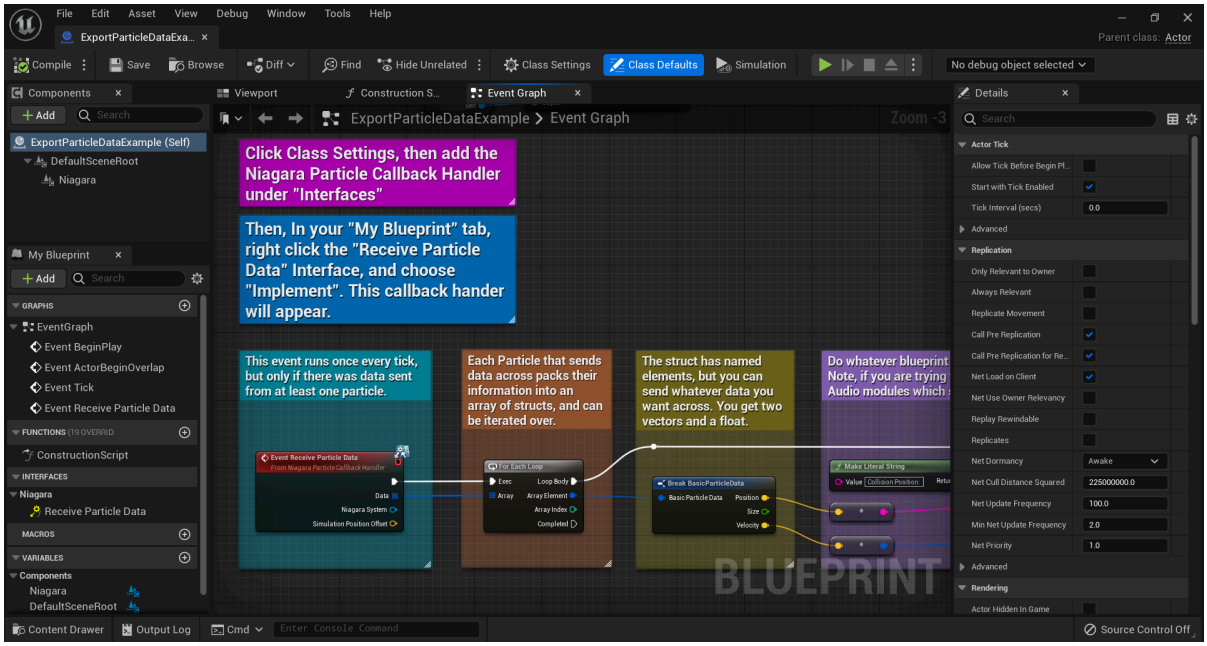
If none of the other approaches are flexible enough for your needs, you can export particle data to Blueprint or C++ and implement your audio logic there.

To do this, use the **Export Particle Data** interface. To see an example in the Content Examples project, search for **export particle** in the Content Drawer.



Click image for full size.

Double-click **ExportParticleDataExample** blueprint for a detailed explanation on how to export data from your particle simulation.



Click image for full size.

One advantage of the export data interface is that it also works on GPU emitters (with some added latency). And once you have the data you need in Blueprint, you can use it not only to

play audio, but also to drive any number of components, such as post processing or user interface (UI) widgets.

Although this is the most flexible way to play audio, it is also the slowest and most complex to set up. You are also somewhat limited by the amount of data you can export, and need to do any mapping between audio component and particle data yourself.

Export Particle Data to BP

Pros	Cons
* Most flexible approach	* Slowest performance
* Works on GPU	* Any module features need to be reimplemented in BP logic
	* Setup is complex