Audio Synesthesia

An overview of how the Audio Synesthesia plugin produces audio metadata that can be used in Blueprint scripting.

Learn to use this **Beta** feature, but use caution when shipping with it.

The **Audio Synesthesia** plugin exposes automatically extracted audio metadata for use in gameplay scripting with Blueprint. This feature helps designers drive animations, effects and other elements that are tightly coupled to sounds.

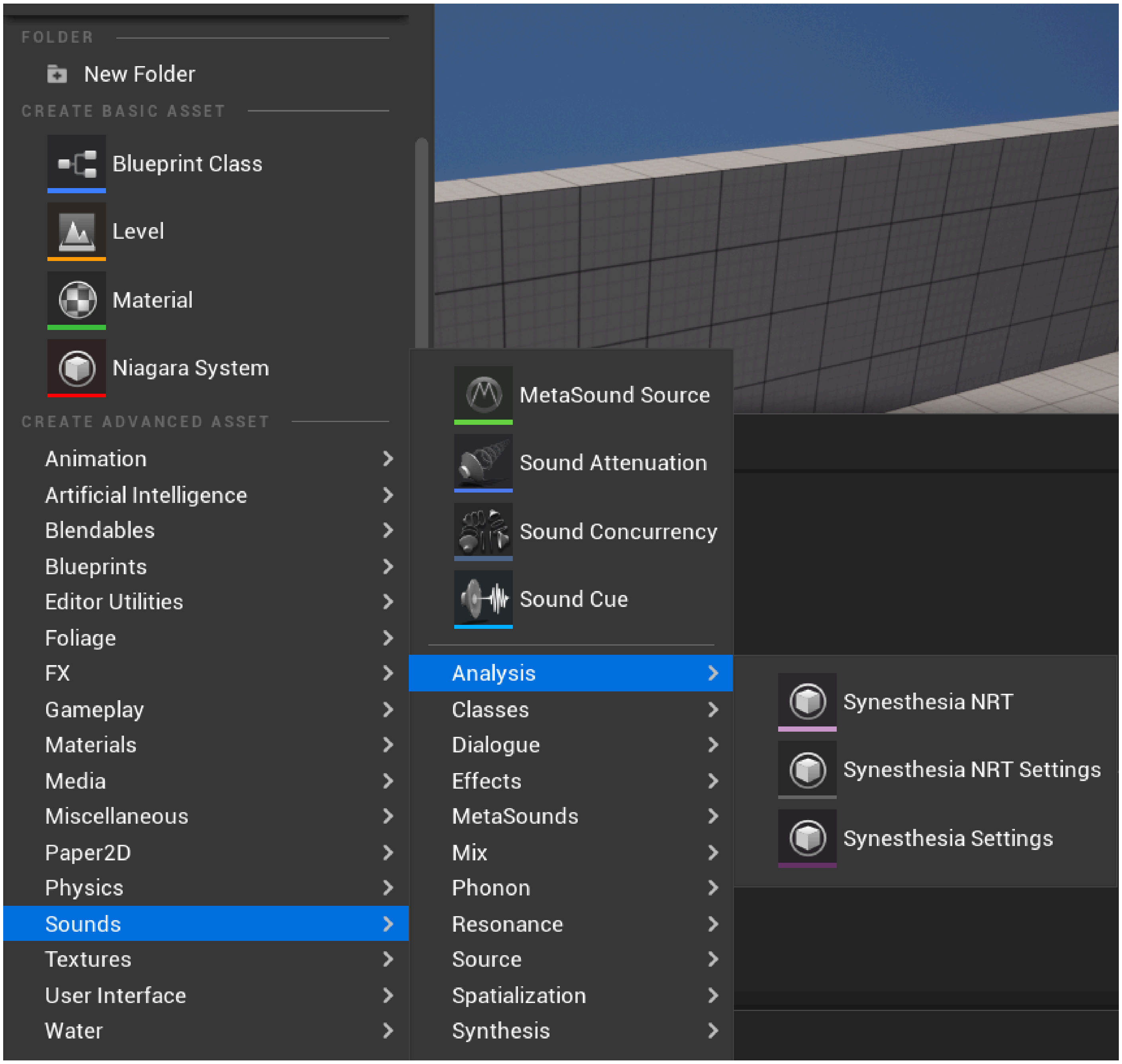
Getting Started

The **Audio Synesthesia** plugin must be activated to use this feature. Go to **Edit > Plugins > Audio**, then check **Audio Synesthesia** to enable. Restart the engine when prompted.

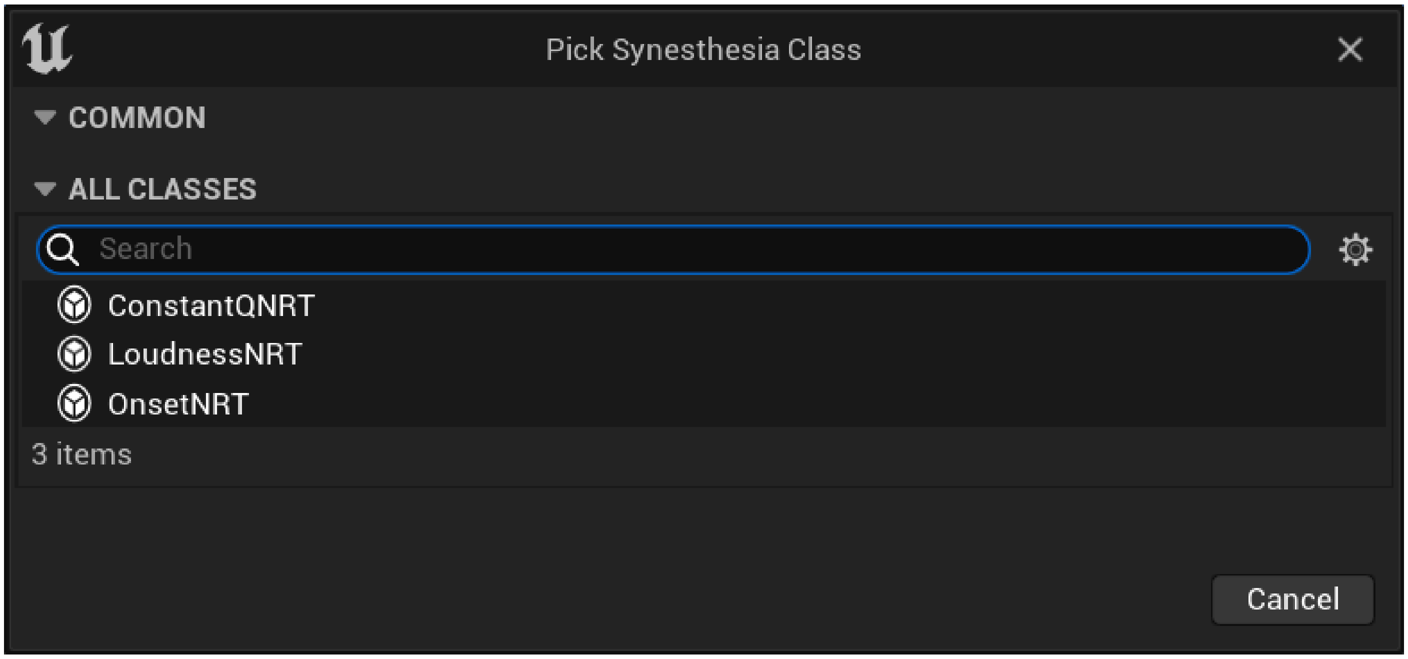
If you add a new Audio Synesthesia asset, Unreal Engine will automatically create an **Audio Analyzer** for it. You can add three types of asset:

- **Audio Synesthesia NRT**: An analyzer that supports non-real-time (**NRT**) analysis of a `USoundWave`.

- **Audio Synesthesia NRT Settings**: The settings for the Audio Synesthesia NRT that you can use to change settings from the default values.
- **Synesthesia Settings**: The settings for the Audio Synesthesia that you can use to change settings from the default values.



When you create a new asset, a dropdown menu will appear. Select the type of analysis you want to perform.



Basic Concepts

What an `AudioSynesthesiaNRT` Object does:

- Links the analysis algorithm settings to the sound wave being analyzed.
- Maintains the results of analysis as a `uasset`.
- Provides Blueprint function access to the results.

All analysis of audio using an `AudioSynesthesiaNRT` Object is performed in the editor. During gameplay, the results of the analysis are read from a file. This limits `AudioSynesthesiaNRT` Objects from analyzing audio that is generated in game, but avoids performing costly audio analysis during runtime.

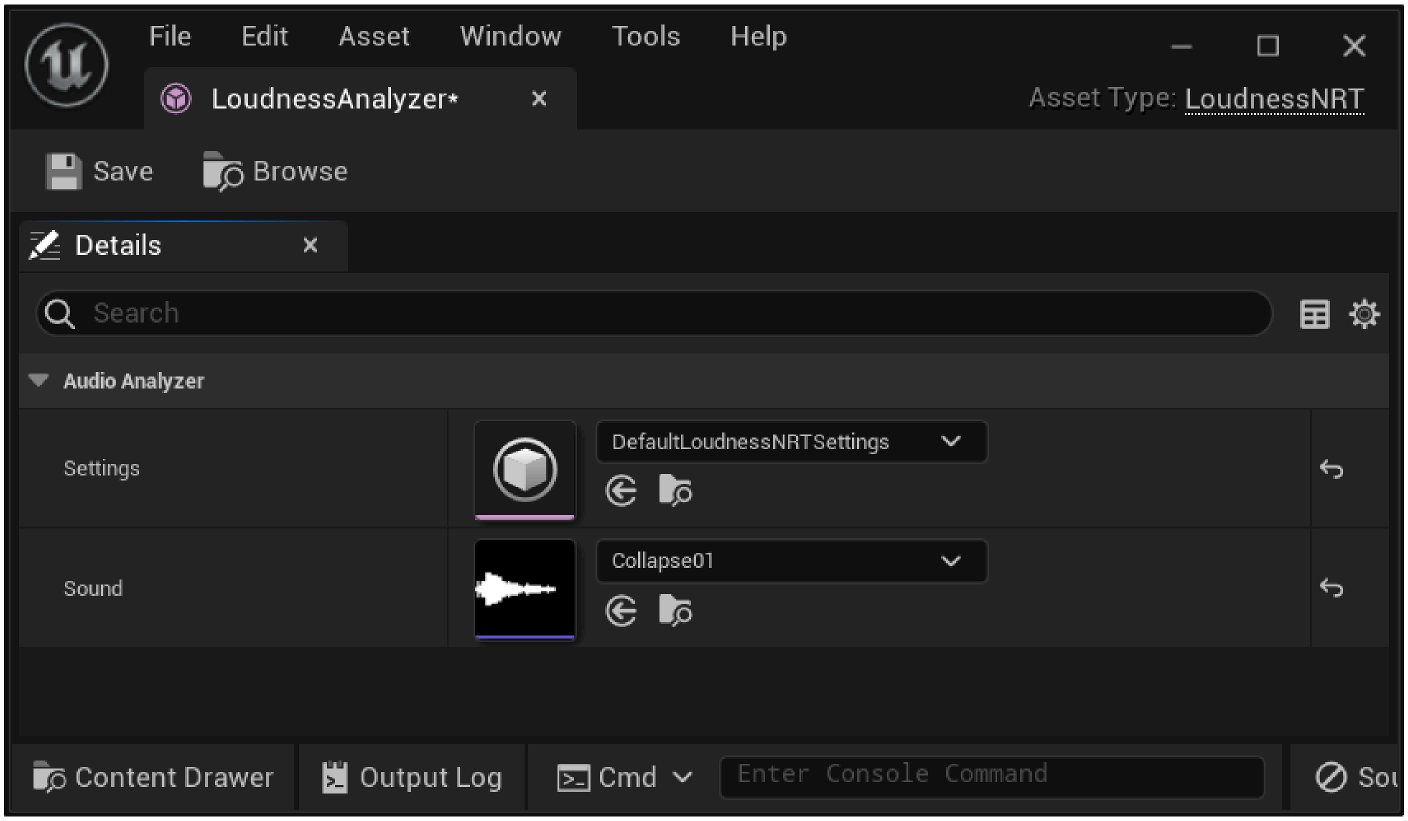
Analysis is performed whenever a setting is updated, or when the sound property is updated in the analyzer.

Example of How the Analyzer Can Be Used

Create and Set Up a LoudnessNRT Analyzer

1. Add an `AudioSynesthesiaNRT` asset. The `AudioSynesthesiaNRT` assets are found in **Audio > Analysis > AudioSynesthesiaNRT**.

2. From the dropdown menu of possible AudioSynesthesiaNRT analyzers, choose **LoudnessNRT**.
3. **Optional:** Add an **AudioSynesthesiaNRTSettings** Object from **Audio > Analysis > AudioSynesthesiaNRTSettings** by selecting **LoudnessNRTSettings** from the dropdown.
4. Update the LoudnessNRT Sound property with an imported sound.



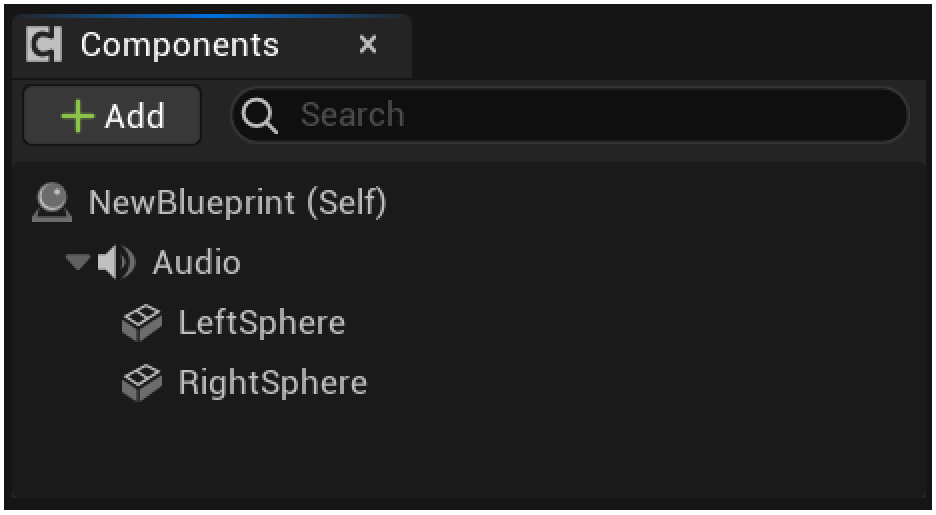
Settings: DefaultLoudnessNRTSettings
Sound: Collapse01

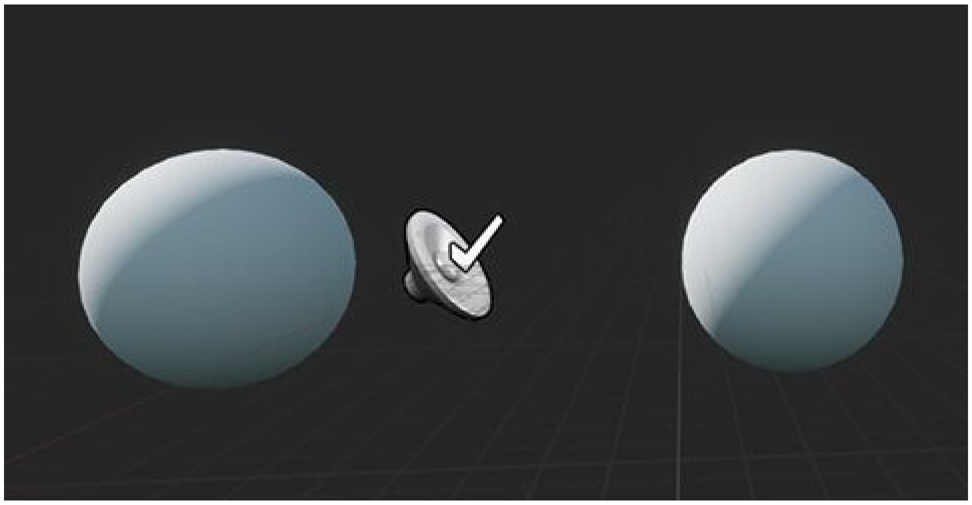
A LoudnessNRT Settings property updated with previously created LoudnessNRTSettings.

Utilize LoudnessNRT within a Blueprint

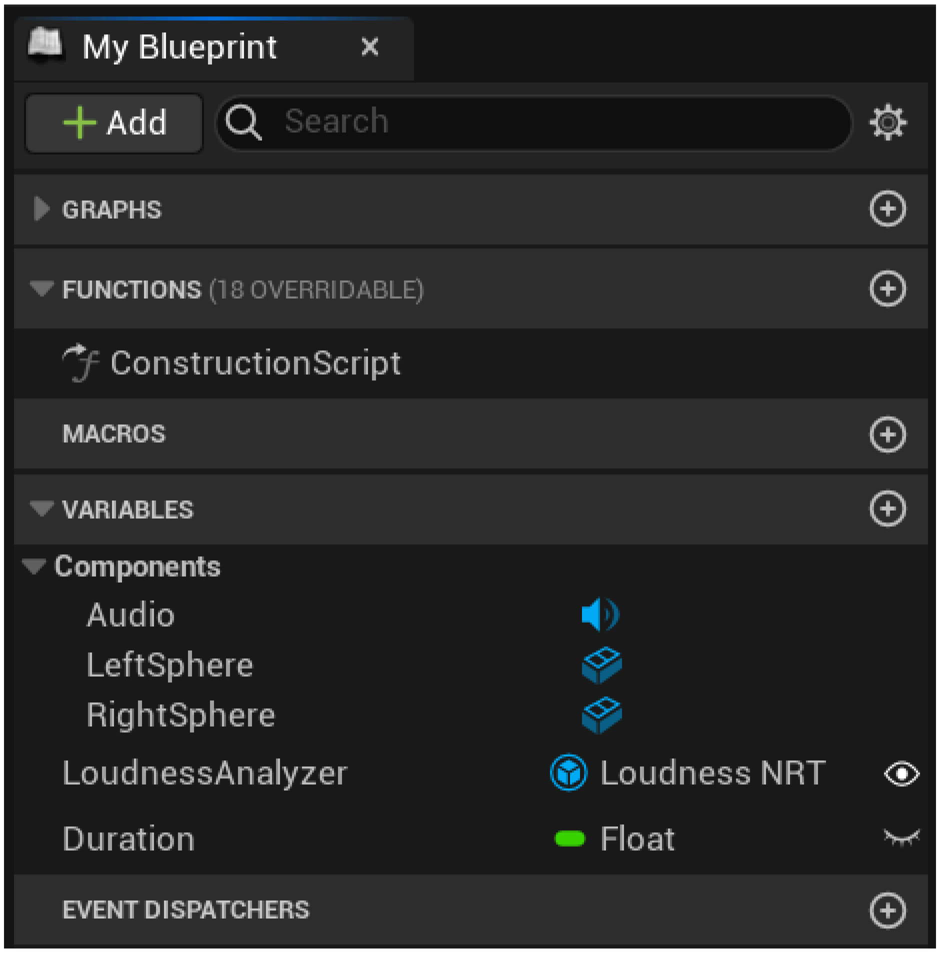
There are many ways to use LoudnessNRT within a Blueprint. Here is an example where two spheres change their sizes in relation to the loudness of the audio that is playing.

1. Create a new Actor Blueprint and add it to your scene.
2. Open the Blueprint, then add an AudioComponent and two spheres. In the example shown, the spheres are named *LeftSphere* and *RightSphere*.

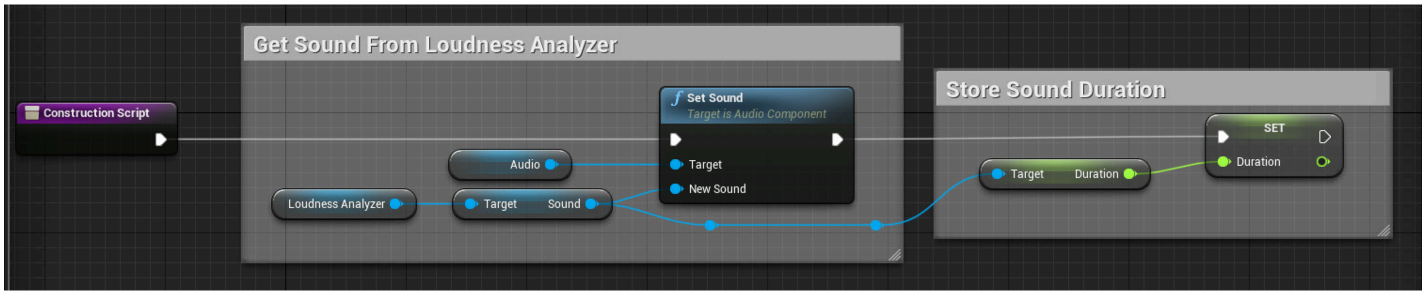




3. Add two variables: **LoudnessAnalyzer** and **Duration**. Duration should be a **float**, and LoudnessAnalyzer should be a **LoudnessNRT**.



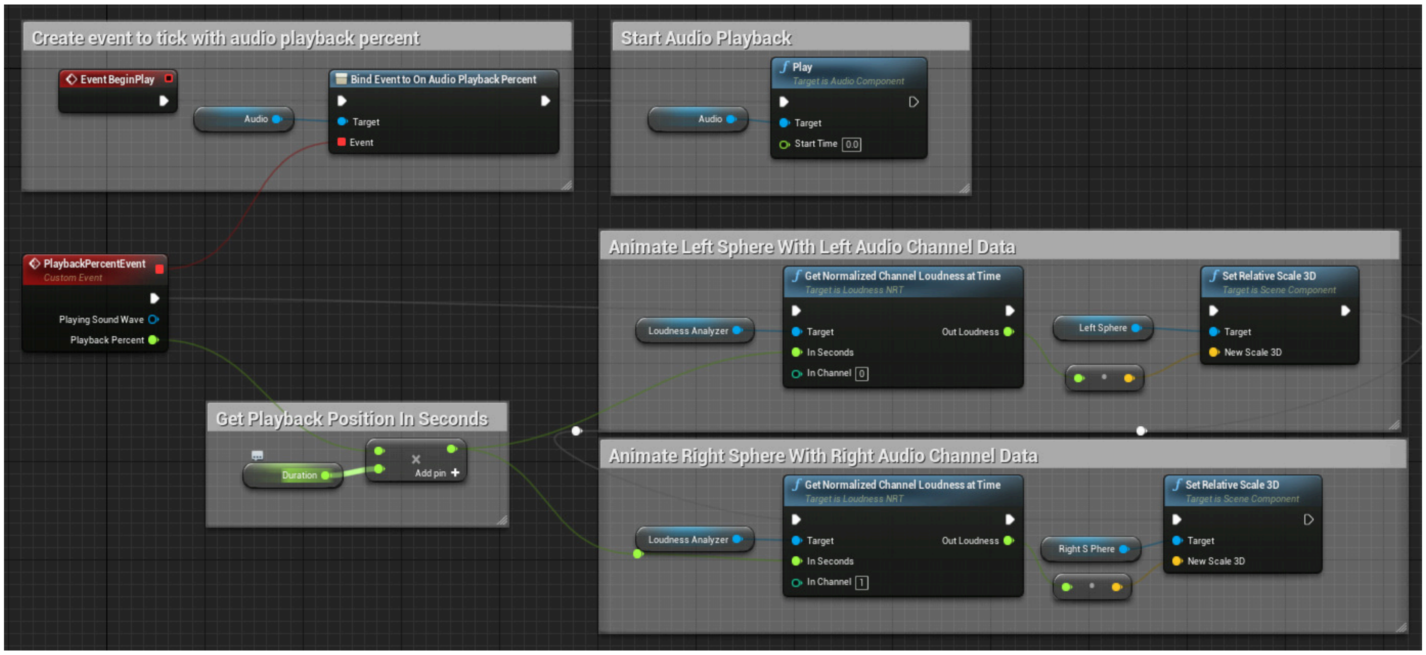
4. In the construction script:
    - Set the sound of the audio component to be the sound used in the Loudness Analyzer.
    - Store the overall duration of the sound.



Click for full image.

1. In the event script:
    - Set up the **AudioPlaybackPercent** event to trigger an event every time the audio playback percentage is updated.
    - Start audio playback.
    - Get the playback in seconds by multiplying the sound duration by the playback percent.

- Retrieve the normalized loudness values from the analyzer at the current playback time in seconds.
- Set the scale of the sphere based upon the loudness value.



Click for full image.

Non-Real-Time (NRT) Analyzers

LoudnessNRT

The LoudnessNRT analyzer measures the perceptual loudness of audio. It is similar to an envelope follower or even a decibel meter in that it is related to the amplitude of the audio and provides a number that changes over time. It differs from a straight amplitude measurement by taking into consideration how humans perceive sound and are more or less sensitive to specific frequencies. For example, a whistle can have a fairly low amplitude but still be perceived as relatively loud. Compare that to the lowest of the lowest bass notes, which can have very high amplitude but be deemed only moderately loud by many listeners. The LoudnessNRT takes all this into account, representing the whistle as loud and the bass note as moderate.

LoudnessNRTSettings is used to configure the LoudnessNRT analyzer. The most commonly tuned parameters in the LoudnessNRTSettings are:

- **AnalysisPeriod** controls how often the measurement is taken.

- **MinimumFrequency** and **MaximumFrequency** control the frequencies of interest. This is useful if you want to completely ignore some content in the upper or lower registers, or if you want to contrast the loudness activity of multiple frequency ranges.
- **CurveType** controls specifically which perceptual curve should be utilized. This is generally an advanced setting.
- **NoiseFloorDb** denotes what amplitude should be considered silence.

ConstantQNRT

The **ConstantQNRT analyzer** measures the strength of individual bands in audio. It is similar to a spectrogram in that it represents the loudness of a frequency band at a given time, but differs by arranging the bands in a perceptually meaningful way. When using the ConstantQNRT analyzer, the bands are arranged the way notes on a piano or bands in a multi-band equalizer are spaced.

ConstantQNRTSettings is used to configure the ConstantQNRT analyzer. The most commonly tuned parameters in the ConstantQNRTSettings are:

- **AnalysisPeriod** controls how often the measurement is taken.
- **StartingFrequency**, **NumBands**, and **NumBandsPerOctave** control the frequency band spacing, width, location and count.
- **DownMixToMono** signifies whether audio channels should be processed separately or mixed down to a single audio channel before processing.

OnsetNRT

The **OnsetNRT analyzer** detects audio onset events originating from a multitude of sources, including note onsets, percussive hits, speaker plosives, and explosions. The OnsetNRT analyzer provides onsets with both their timestamps and strength.

OnsetNRTSettings is used to configure the OnsetNRT analyzer. The most commonly tuned parameters in the OnsetNRTSettings are:

- **GranulartiyInSeconds** determines the minimum spacing between onsets.
- **Sensitivity** sets the threshold for how loud an onset must be before it is detected.
- **MinimumFrequency** and **MaximumFrequency** control the frequency range to look for onsets. This is useful if you want to completely ignore some content in the upper or lower

registers, or if you want to contrast the onset activity of multiple frequency ranges. * **DownMixToMono** signifies whether audio channels should be processed separately, or mixed down to a single audio channel before processing.