Ignition / AudioChannels

Namespace **Playniax.Ignition**

Inherits from **MonoBehaviour**

Script can be found in **Assets/Playniax/Framework/Ignition/Scripts (MonoBehaviour)/AudioChannels.cs**

Public fields	Description
`static bool mute = false`	Global variable to enable or disable sound.
`static AudioSource[] channels`	Allocated channels.

Public Methods	Description
`static AudioSource GetAvailableChannel()`	Returns the first available channel.
`static AudioSource Play(AudioClip audioClip, float volumeScale = 1f, float panStereo = 0, float pitch = 1)`	Play AudioClip using the first available channel.

The AudioChannels component is an audio manager to manage the sounds that are played using the Igntion AudioProperties component.

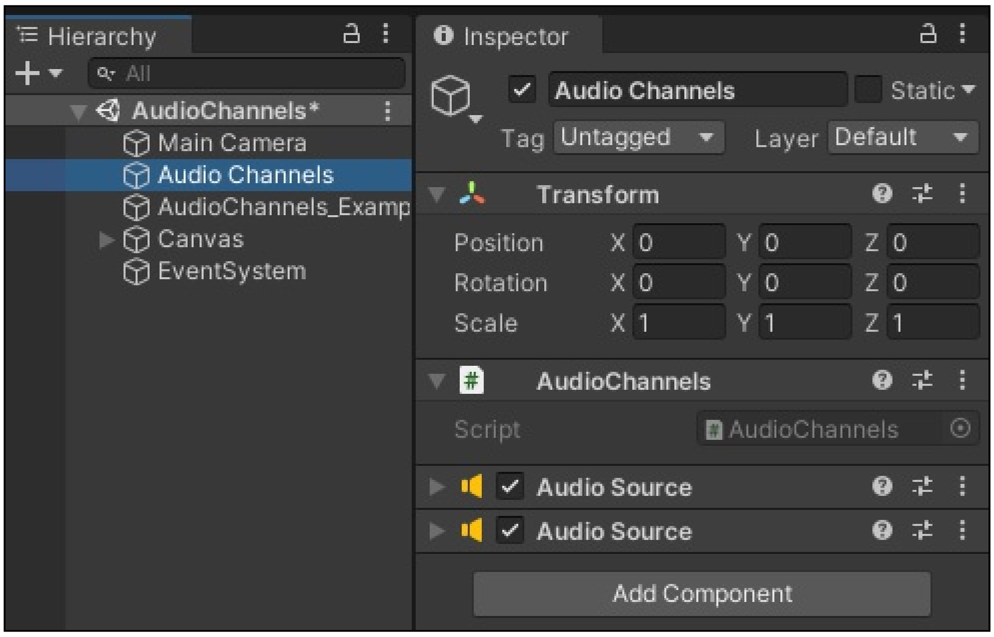
AudioChannels require Unity AudioSource components.

The number of AudioSource components present in the scene determine how many sounds can be played at the same time.

Basic setup requires a GameObject with the AudioChannels component and the number of AudioSource components for each channel you need.

You can then use the AudioChannels Play command to play a sound or use the AudioProperties Play command to play a sound.

Example of the GameObject:



Audio Channels — Static
Tag: Untagged — Layer: Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

AudioChannels
Script: AudioChannels

Audio Source
Audio Source

Example can be found in **Assets/Playniax/Framework/Ignition/Examples/01 - Framework/AudioChannels.unity**

Example script can be found in **Assets/Playniax/Framework/Ignition/Examples/01 - Framework/Scripts (MonoBehaviour)/AudioChannels_Example.cs**

```
using UnityEngine;
using Playniax.Ignition;

public class AudioChannels_Example : MonoBehaviour
{
    // Ignition sound object.
    public AudioProperties audioProperties;

    void Start()
    {
        // State to console.
        Debug.Log(AudioChannels.mute ? "off" : "on");
    }
    public void Play()
    {
        // Play sound
        audioProperties.Play();
    }

    public void Mute()
    {
        // Toggle sound
        AudioChannels.mute = !AudioChannels.mute;

        // State to console.
        Debug.Log(AudioChannels.mute ? "off" : "on");
    }
}
```

This way sound effects in your game can be turned on or off with just a single command by changing the mute variable to a true or false state.

The SimpleGameUI uses this system too when you toggle sound.

By using this system the SimpleGameUI can control the sound effects and allow them to be played or not.