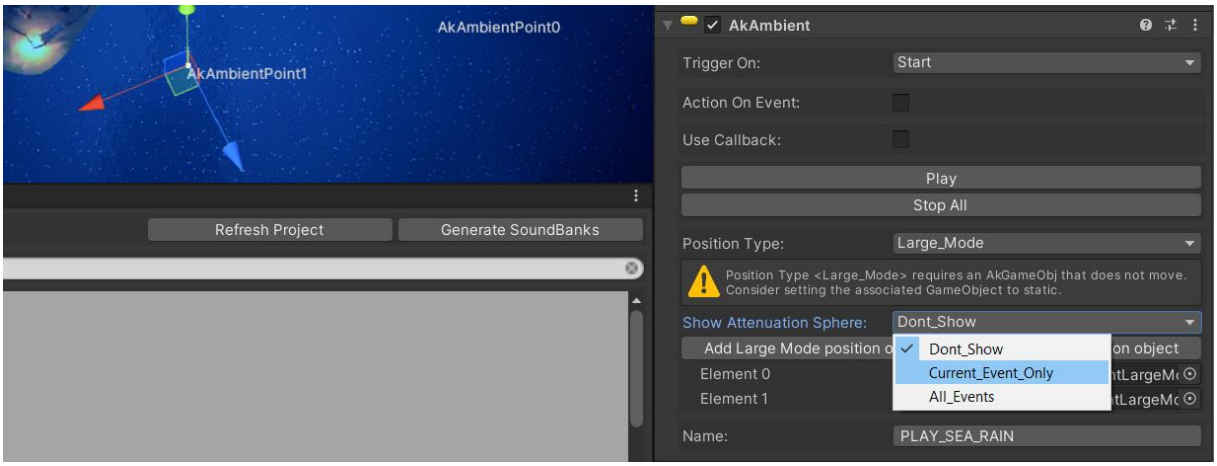
Wwise – Unity tip: HOW TO DISPLAY ATTENUTATION SPHERES

I will show you in this note how to resolve the old attenuation sphere issue in Unity WWise APi.

Issue Description

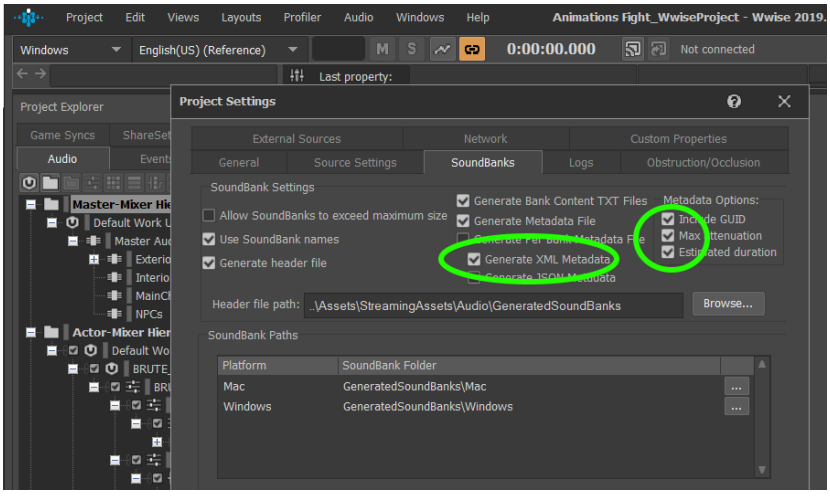
If you load a AkAmbient component in your Unity project, you would normally be able to see attenuation sphere when you choose to "show attenuation sphere" in "current event only" or "all events" modes.



In some cases, the attenuation spheres **are not displayed** (especially if you work on French Work Stations).

My Solution

First, you have to check in your WWise project that you selected "generate XML metadata" and "Max attenuation".



Then the max attenuation distance will be able to be exported in XML data to be read by UNity-Wwise API.

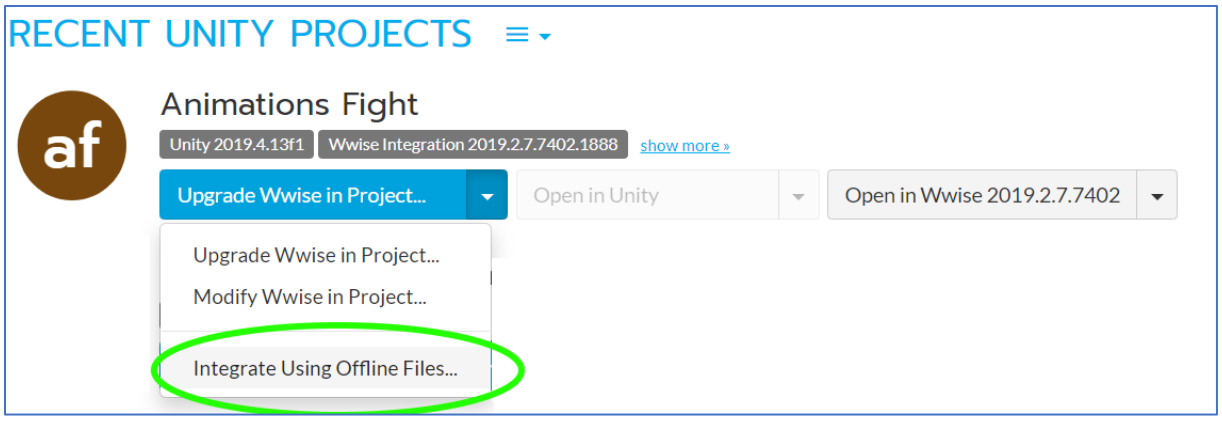
In fact the WWise APi try to read an XML value with a dot to express max distance (ex: "10.50"). In some languages like French, this value would have to be expressed with a comma (like "10,50"), and not with a dot.

So the XML reader script fail to understand “10.50".

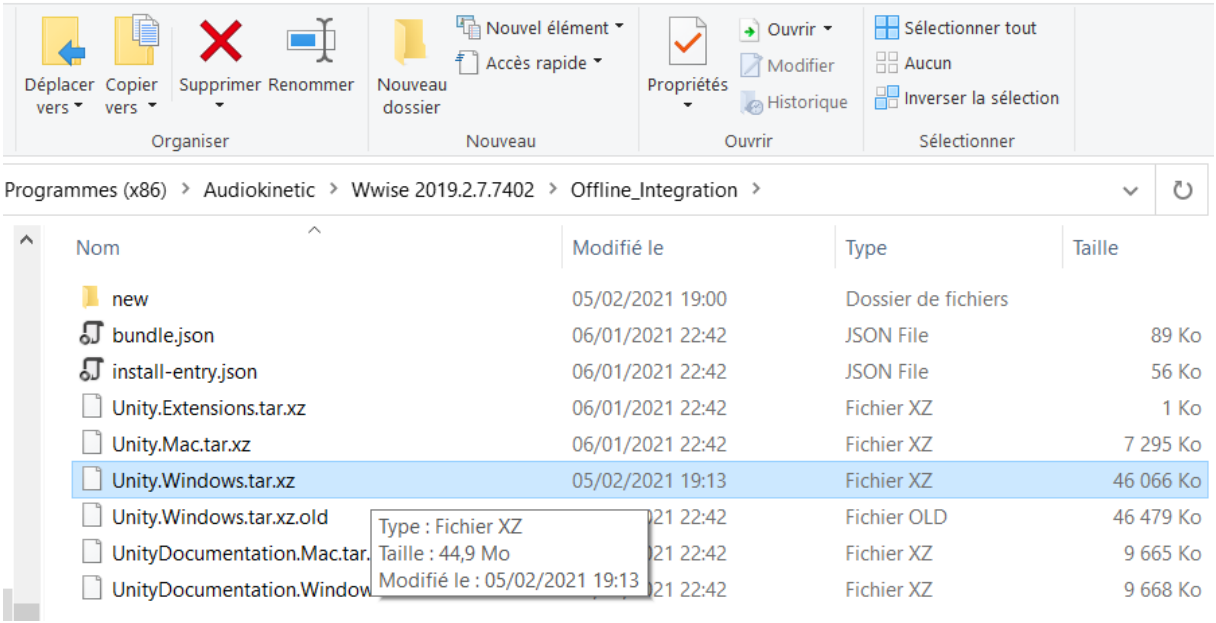
I corrected the script of the XML reader class in WWise Unity API, in a way that the XML parser can understand international naming convention with dots.

You can replace AkWwiseXMLBuilder.cs file in you want to see attenuation spheres in the following folder: “MyUnityProject\Assets\Wwise\Editor\WwiseWindows” (see code in annexe or download).

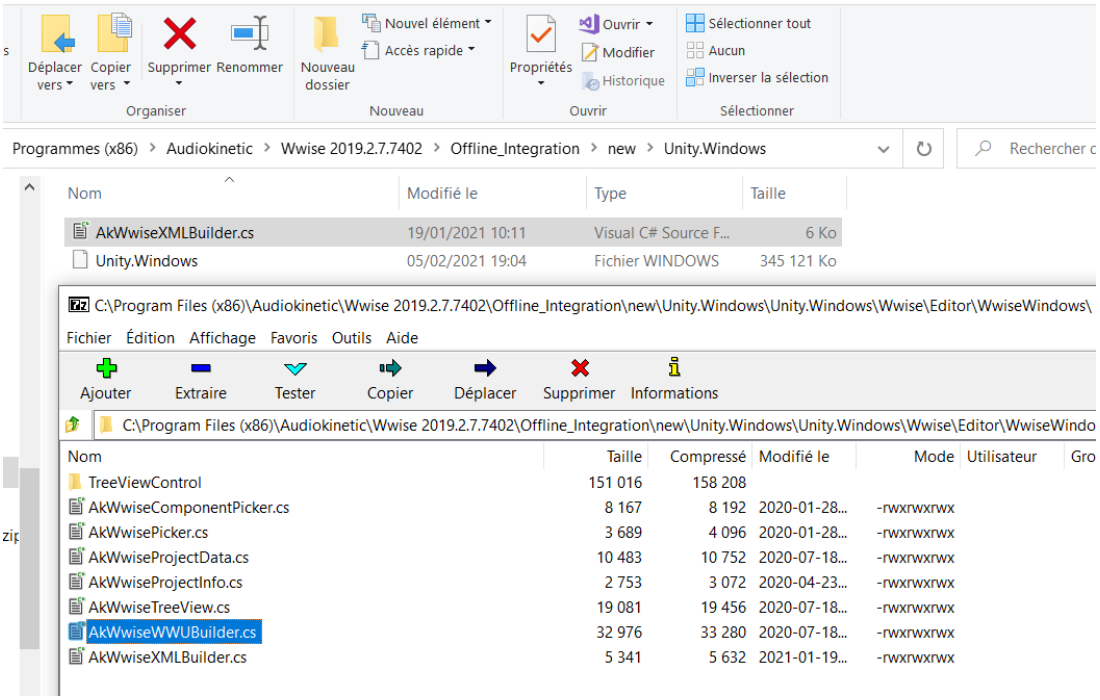
If you do not want to copy this file each time you create a project, you will have to integrate Wwise using Offlines WWise Api:



And locate the downloaded archive in Wwise install folder (named Unity.Windows.tar.xyz):



You will finally have to replace in this archive the AkWwiseXMLBuilder.cs file by the new one:



In that way, each time you integrate Wwise in a new Unity project, the right cs file will be automatically copied.

Correction made by Thibault BEGIN

Thibault.begin@gmail.com

Annexe: new AkWwiseXMLBuilder.cs script

```
#if UNITY_EDITOR
//////////////////////////////////////////////////////////////////////
//
// Copyright (c) 2014 Audiokinetic Inc. / All Rights Reserved
// 19/01/2021 Thibault BEGIN (FRANCE): Corrected exception when parsing max attenuation for french Work
Stations
//////////////////////////////////////////////////////////////////////

using System;
using System.Globalization;
using System.Xml;
using UnityEngine;

[UnityEditor.InitializeOnLoad]
public class AkWwiseXMLBuilder
{
	private static readonly System.DateTime s_LastParsed = System.DateTime.MinValue;

	static AkWwiseXMLBuilder()
	{
		AkWwiseXMLWatcher.Instance.PopulateXML = Populate;
		UnityEditor.EditorApplication.playModeStateChanged += PlayModeChanged;
	}

	private static void PlayModeChanged(UnityEditor.PlayModeStateChange mode)
	{
		if (mode == UnityEditor.PlayModeStateChange.EnteredEditMode)
		{
			AkWwiseProjectInfo.Populate();
			AkWwiseXMLWatcher.Instance.StartWatcher();
		}
	}

	public static bool Populate()
	{
		if (UnityEditor.EditorApplication.isPlayingOrWillChangePlaymode ||
UnityEditor.EditorApplication.isCompiling)
		{
			return false;
		}

		try
		{
			// Try getting the SoundbanksInfo.xml file for Windows or Mac first, then try
to find any other available platform.
			var logWarnings = AkBasePathGetter.LogWarnings;
			AkBasePathGetter.LogWarnings = false;
			var FullSoundbankPath = AkBasePathGetter.GetPlatformBasePath();
			AkBasePathGetter.LogWarnings = logWarnings;

			var filename = System.IO.Path.Combine(FullSoundbankPath,
"SoundbanksInfo.xml");
			if (!System.IO.File.Exists(filename))
			{
				FullSoundbankPath =
System.IO.Path.Combine(UnityEngine.Application.streamingAssetsPath,
AkWwiseEditorSettings.Instance.SoundbankPath);

				if (!System.IO.Directory.Exists(FullSoundbankPath))
					return false;

				var foundFiles = System.IO.Directory.GetFiles(FullSoundbankPath,
"SoundbanksInfo.xml", System.IO.SearchOption.AllDirectories);
				if (foundFiles.Length == 0)
					return false;

				filename = foundFiles[0];
			}

			var time = System.IO.File.GetLastWriteTime(filename);
			if (time <= s_LastParsed)
```

```
                        {
                                return false;
                        }

                        var doc = new System.Xml.XmlDocument();
                        doc.Load(filename);

                        var bChanged = false;
                        var soundBanks = doc.GetElementsByTagName("SoundBanks");
                        for (var i = 0; i < soundBanks.Count; i++)
                        {
                                var soundBank = soundBanks[i].SelectNodes("SoundBank");
                                for (var j = 0; j < soundBank.Count; j++)
                                {
                                        bChanged = SerialiseSoundBank(soundBank[j]) || bChanged;
                                }
                        }

                        return bChanged;
                }
                catch
                {
                        return false;
                }
        }

        private static bool SerialiseSoundBank(System.Xml.XmlNode node)
        {
                var bChanged = false;
                var includedEvents = node.SelectNodes("IncludedEvents");
                for (var i = 0; i < includedEvents.Count; i++)
                {
                        var events = includedEvents[i].SelectNodes("Event");
                        for (var j = 0; j < events.Count; j++)
                        {
                                bChanged = SerialiseEventData(events[j]) || bChanged;
                        }
                }

                return bChanged;
        }

        private static float GetFloatFromString(string s)
        {
//here is my solution to resolve the issue, the float parsing must be done in
CultureInfo.InvariantCulture.NumberFormat
                return string.Compare(s, "Infinite") == 0 ? UnityEngine.Mathf.Infinity : float.Parse(s,
CultureInfo.InvariantCulture.NumberFormat);
        }

        private static bool SerialiseEventData(System.Xml.XmlNode node)
        {
        XmlAttribute maxAttenuationAttribute = node.Attributes["MaxAttenuation"];
                var durationMinAttribute = node.Attributes["DurationMin"];
                var durationMaxAttribute = node.Attributes["DurationMax"];
                if (maxAttenuationAttribute == null && durationMinAttribute == null &&
durationMaxAttribute == null)
                        return false;


        Debug.LogWarning("Sphere Attenuation Issue Correction now applied");

        var bChanged = false;
                var name = node.Attributes["Name"].InnerText;
                foreach (var wwu in AkWwiseProjectInfo.GetData().EventWwu)
                {
                        var eventData = wwu.Find(name);
                        if (eventData == null)
                                continue;
            try
            {
                if (maxAttenuationAttribute != null)
                                {
                    string value = maxAttenuationAttribute.Value;
                    string inner = maxAttenuationAttribute.InnerText;
```

```
                    float maxAttenuation  = GetFloatFromString(maxAttenuationAttribute.Value);

                                if (eventData.maxAttenuation != maxAttenuation)
                                {
                                        eventData.maxAttenuation = maxAttenuation;
                                        bChanged = true;
                                }
                        }

                        if (durationMinAttribute != null)
                        {
                                var minDuration =
GetFloatFromString(durationMinAttribute.InnerText);
                                if (eventData.minDuration != minDuration)
                                {
                                        eventData.minDuration = minDuration;
                                        bChanged = true;
                                }
                        }

                        if (durationMaxAttribute != null)
                        {
                                var maxDuration =
GetFloatFromString(durationMaxAttribute.InnerText);
                                if (eventData.maxDuration != maxDuration)
                                {
                                        eventData.maxDuration = maxDuration;
                                        bChanged = true;
                                }
                        }
                }
                catch (Exception ex)
                {
                    Debug.LogError("AkWwiseXMLBuilder->SerialiseEventData:" + ex.Message);
                }
        }

                return bChanged;
        }
}
#endif
```