

# Wwise 2019.2 update report

Gaspard Morel 2020.04.10

Notes : the “API” in this text will refer to the Spatial Audio API, meaning the set of programming function integrated into Wwise for calculating these advanced position data (see the Wwise Spatial Audio PPT document for more details)

1. [Spatial Audio and Reflect plugin update introduction](#)
2. [Setup and features changes](#)
  - a. [General](#)
  - b. [Spatial Audio setup](#)
  - c. [Geometry](#)
  - d. [Portals](#)
  - e. [Obstruction, Occlusion, Diffraction and Transmission](#)
  - f. [Room tones](#)
3. [Performance analysis](#)
  - a. [Unreal Engine](#)
  - b. [Unity](#)
  - c. [Audio comparison for diffraction](#)
4. [Profiler update](#)
5. [Links](#)

## **1. Spatial Audio and Reflect plugin update:**

Spatial Audio is an API used to simulate position of sound emitters in a 3D environment, taking into account walls, rooms, propagation, occlusion and diffraction, and the paths to all of this information to the player's listener in the 3D world. Coupled to other tools like the Reflect plugin, it is the Wwise in-house solution for accurate sound propagation, reactive soundscapes and immersive refinements for greater fidelity in virtual acoustics in games using its audio engine.

Prior to this update, Spatial Audio was a setup that needed a lot of manipulation of the game engine to get the logic, behavior, sources and geometry to be sent to the API to be processed by Wwise. The authoring was only used for setting up Reflect settings on Aux buses, using Reverb Aux buses for 3D reverbs in rooms and sound coming out of rooms, and global Occlusion/Obstruction filters depending on positions in the game.

Now, the promise is to get more granular and bit easier on the management and creative side of this API's usage. Less setup to do in the game engine is less time lost and less errors made, more control over the sounds routing from the authoring is also more logical for designer to manipulate the desired effects, and having some parts of the API automated (like the Reflect calculation done per active sound instead of per game object, the objects collisions with rooms, the global values for reflections raycast) also comes with interrogations concerning the potential performances issues or improvements made on that front.

Let's detail the changes a bit, then measure up in different scenarios how the performance compares to before.

## **2. Setup and features changes**

-General: Spatial Audio functions are now setup in the Wwise authoring instead of in the game engine. (Functions like giving a sound source to the Spatial Audio API, defining the number of Reflections and the sounds routed to it).

There is no more specific Spatial Emitter to register in the Game Engine, the API knowing automatically now which sounds are needed to be considered for calculations. Additionally, the Spatial Audio data is now unique to each sound, not linked to a specific game object emitter.

In the Wwise authoring application, Properties can be applied to Sound Objects and Bus objects for being sent to the API when being played on a game object. This include enabling Early Reverberation specific Aux send for using reflections, Game-defined Aux send to be included in Rooms, and Wwise Reflect effect, and enabling Diffraction.

Enabling the Early Reverberation Send is sending the sound to an Aux bus using a Reflect plugin in order to render the early reflection (based on the calculation done in the engine from the emitter that the sound is playing on). The level of send to this aux bus can be set there as well.

Enabling the Diffraction is starting the computation of the diffraction for the paths between the emitter and the Spatial Audio Listener (including through Portals and across geometric

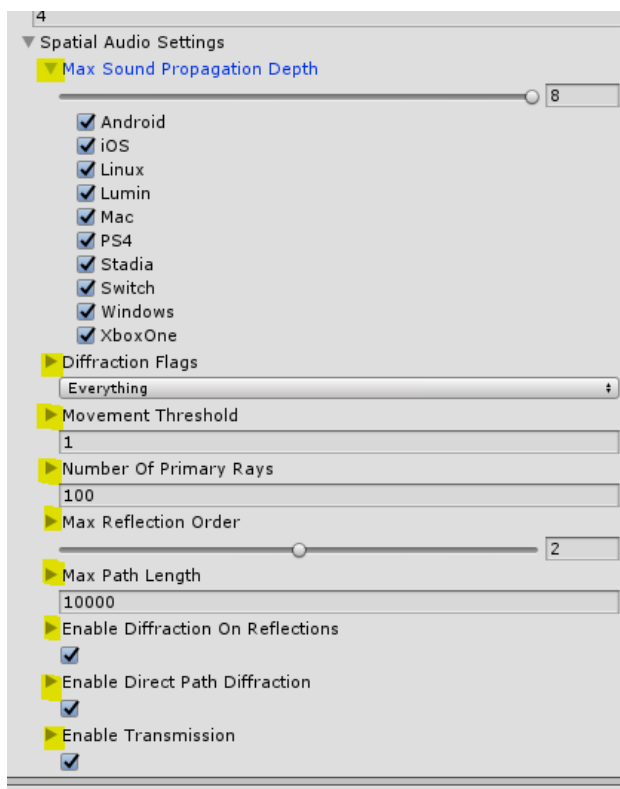
edges). It also enables the Obstruction value of the sound to be based on its diffraction value, sets the diffraction built-in rtpc parameter (for the emitter of this sound) based on its diffraction value, and finally use the virtual positions of a sound event using a multi-position configuration to have the diffraction being calculated for each virtual position.

It also means that the maximum length path of rays used by a sound in this new behavior is now automatically tied to the Attenuation settings of a sound. The API will not go over the length of the maximum distance set in the attenuation for a sound, thus restraining computation automatically when it is not needed.

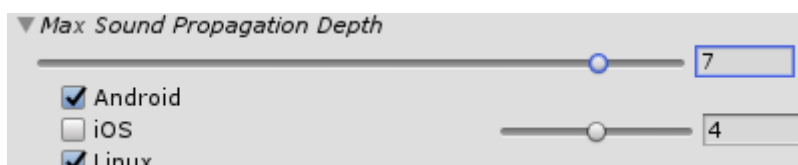
-Spatial Audio setup is more detailed, granular and global to the project

Since there is no more specific Spatial Emitter registration, a lot of parameters are now a configuration shared by all sounds that are going to be processed by the API, located in the Wwise settings of the project.

In Unity (the Wwise Initialization script):

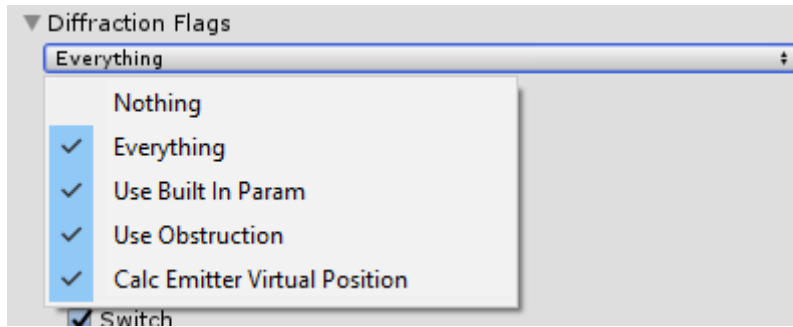


(note that all the settings can be platform-dependent)



Max sound propagation Path: the maximum number of portals a sound path can go through at a time

Diffraction flags: Can be used to override the functions activated by the use of Diffraction on a sound (as described earlier). Options are to use or not each computation and the diffraction value driving the obstruction and built-in diffraction RTPC values.



Movement threshold: the amount an emitter or a listener has to move for the API to recalculate the reflections/diffraction. Higher is less accurate but can reduce the CPU load.

Number of primary Rays: number of rays used in stochastic (randomly determined) shooting for determining an emitter position in the 3D space.

Max reflection order: The number of maximum rebounds a reflection path can have. Higher means more detail but higher CPU usage.

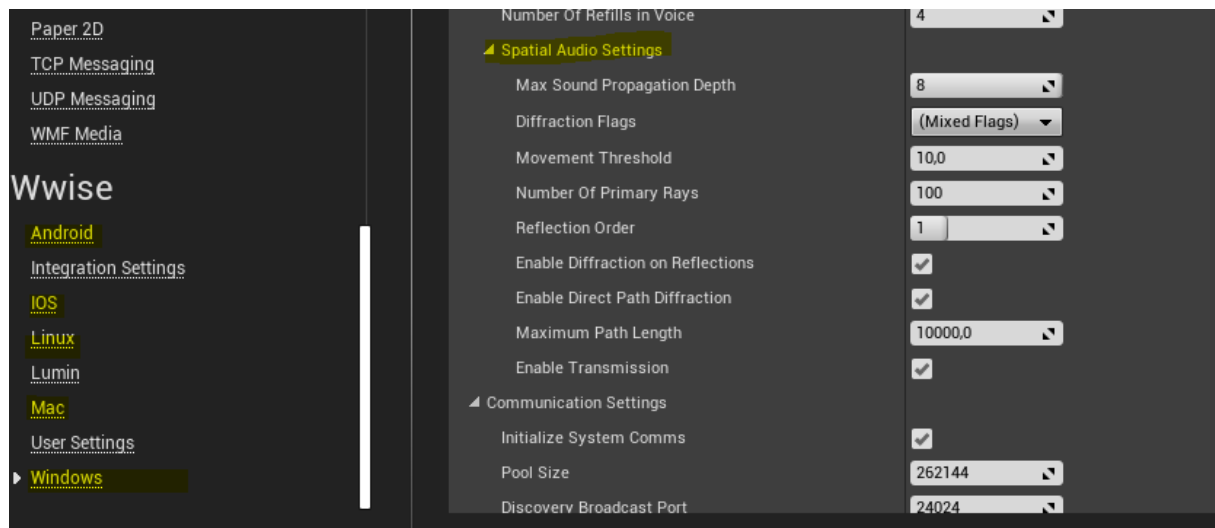
Max path length: This is a global value to restrict maximum length of individual segments of reflections or diffraction paths (rays). Beware, Attenuation is considered infinite when a far-right volume point on the attenuation graph is above the volume threshold set in the project. So, the Attenuation of a sound supposed to be using the API must be considered conscientiously.

Enable diffraction on Reflections: Using diffraction on reflection path. If enabled, it can make the early reflections to fade in and out smoothly when the emitter or listener move around the reflections' shadow/edge area.

Enable direct path Diffraction: Enabling or disabling the diffraction around geometry for sounds are using diffraction. If disabled, it will only use diffraction on paths going through Portals.

Enable transmission: Enabling or disabling the Transmission behavior through walls (see Occlusion and Transmission later in this document).

## In Unreal Engine 4:



The only difference here is the presentation for platform-specific settings, that are different values accessible on the left.

### -Unreal Engine (4.23 and above) Geometry

Unreal can now use meshes actors to get geometry sent to the API. Previously the Reflectors could only be attached to geometry primitives (using the AKSpatialVolume cubic object or using blueprint), but now AKGeometry component can be attached to static or collisions meshes. If you wanted to add reflectors to static meshes, the best practice was to add a size-matching AKSpatialAudiovolume around the mesh.

Acoustic textures can also be mapped to physical materials in the UE project, so that AKGeometry or Surface Reflector component automatically assign Wwise acoustic textures to UE physical materials for Early Reflections.

In Unreal Engine, all sounds will now be automatically considered for positions in rooms to get sound propagation through rooms and portals and reaching the Spatial Audio Listener in a scene.

### -Unity (2018.2 and above) Geometry

Unity benefits from the simplification of the setup and the increase in performances for diffraction, and reflections, since it can now select any mesh for an AkSurfaceReflector (the geometry passed to the API) component on an object. As with UE4, it can also assign different Acoustic textures to the mesh's materials. Performances comparisons are made later in this document.

Compared to UE, Audio game objects in Unity still need to get specific function to be interacting with Spatial Audio Rooms and being seen in a specific position related to Rooms and Portals by the API. You need to add the AK Room Aware script on every audio game object you want to be using Room positioning.

-Portals have automatic attenuation, meaning that attenuation settings on the bus of a reverberation attached to a room is no longer necessary (but can be used for aesthetic purposes). Wwise now applies the attenuation of a sound on its dry/direct path but also on the wet path going through the reverberation.

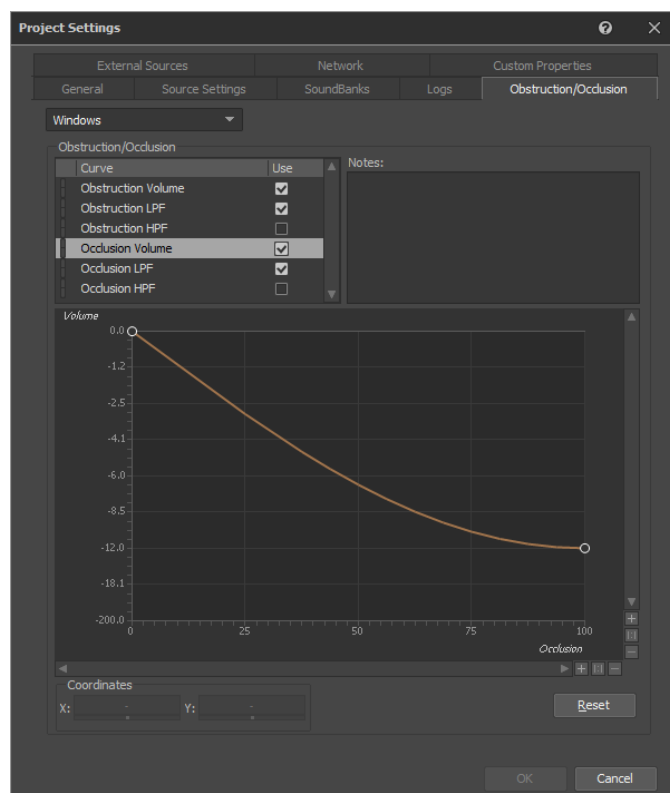
Useful example quoted from the Wwise SDK documentation:

A common issue that users had with portals before Wwise 2019.2, was that the relative mix of two sounds with different attenuations was not consistent in the reverb that was transmitted through portals. For example, suppose both footsteps - a short attenuation, and gunfire - a long attenuation, play through the same portal. The sounds were mixed together at the location of the portal then re-spatialized at that same position. The sounds' attenuations were evaluated using the distance between the emitter and the portal, which was only a portion of the total distance between the emitter and the listener - Spatial Audio relied on the portal's attenuation to further attenuate the reverb. The result was that as the listener got further away from the portal, the relative volume of the two sounds remained the same instead of the footsteps decaying faster than the gunfire as the sound's original attenuation curve dictated. Note that this was only an issue with the wet path - the direct path worked as expected.

In Spatial Audio for Wwise 2019.2, the problem is solved by effectively putting the portal's "pickup" location at the same position as the listener. This way, the entire distance of the path between the sound source and the listener is applied to the attenuation curve of the sounds before mixing the sound into the room's 3D bus. Attenuation applied to the portal (ie room bus) is no longer necessary, but may still be desired. An attenuation post-mix would serve to further reduce the volume of the reverb or apply additional filtering to the reverb that passes through the portal.

## -Obstruction and Occlusion

The foundation of these systems is relying on the Obstruction/Occlusion curves in the Wwise project.



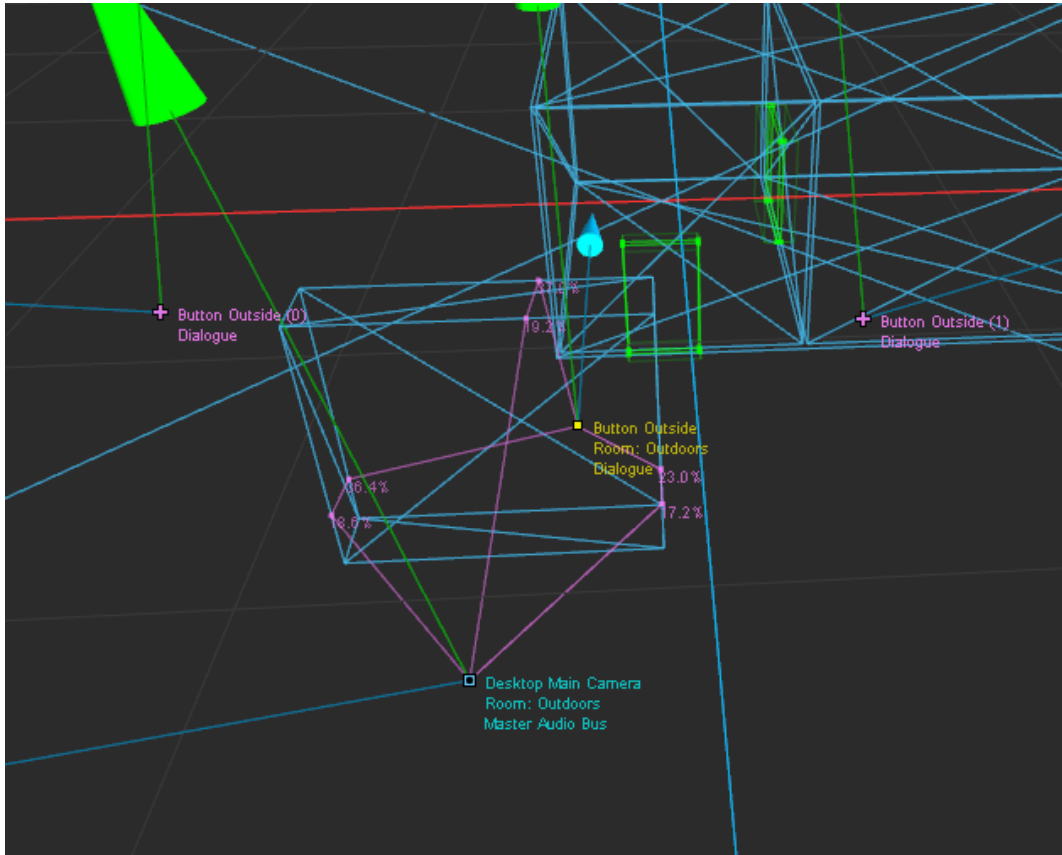
The basic obstruction is a system that checks if there is an obstacle in the path between an emitter and a listener, and the only way to setup this was to get each Sound emitter in the scene that should react to these values to be set up with specific values or components (in Unity using the `AKEmitterObstructionOcclusion` component, or the `Occlusion` value on an `AKComponent` in UE4).

If there is an obstacle, the system will basically transition from 0 to 100 occlusion/obstruction using the specified fade time. This system was efficient but didn't give a lot of control on different sorts of sounds, situations or geometry components (Up to Wwise 2019.1.7, Unity had the room walls and surface reflector using occlusion value when using Spatial Audio, Obstruction being applied over time using the specific component, and UE was using the Visibility check between listener and emitter. Both engines could use only Occlusion on emitters when no Spatial Audio Room were used in the scene, using Obstruction values only when calculating paths in relation to Rooms.).

Spatial Audio in 2019.2 make this system a bit easier and more flexible, and using it on the two engines is also now close to being the same. Up to 2019.1.7, Diffraction calculations were difficult to set up as an additional step, and required specific management for having manageable performances.

#### → Difference between Obstruction and Diffraction in 2019.2

Diffraction angles can drive the Obstruction value. Both are still usable as RTPC built-in parameters (for using even more custom setups in the authoring) but diffraction gives more precise values based on angles and respect attenuation cutoff for sound objects to be processed. The API also calculates virtual positions for the obstructed source instead of using the original 3D position of a source behind an obstacle.



Diffraction is set up in authoring per sound objects, not in the game engine. Using UE4, obstruction/occlusion has to be turned off (0 refresh interval) on AKComponents when using Diffraction.

No need to set up layers in Unity for paths and obstacles collisions, the diffraction is calculated between Wwise objects (emitters, rooms, reflectors and portals) only. In UE4, using Diffraction is not influenced by the Collision Channel used by the global occlusion parameter as well.

Diffraction can be used for Obstruction without Rooms, since Wwise will only use Occlusion when no Room is in the scene. Obstruction is applied to emitters that are in the same room as the Spatial Audio listener, when Diffraction is applied through Portals.

No need to set up Obstruction on Portals in Unity as well, since the sound coming from the Portals will also be Diffracted automatically based on the paths going through (in both engines).

#### -Difference between Occlusion and Transmission in 2019.2

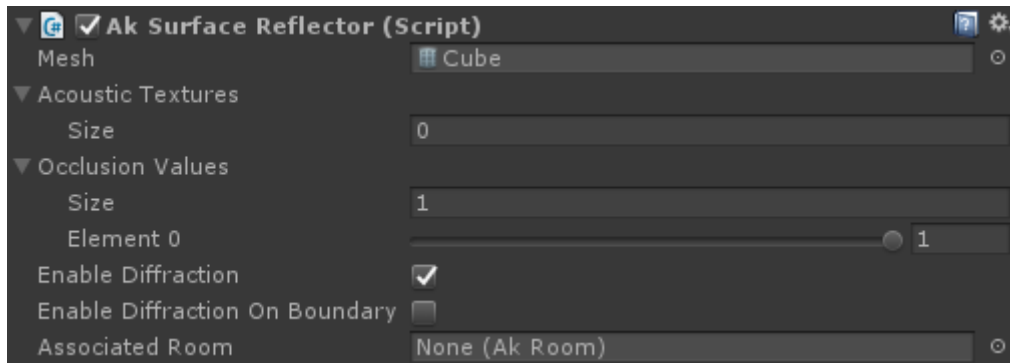
*Same as Diffraction, Transmission is calculated by the API between Wwise object (emitters, geometry, rooms and listener) without using layers in Unity or collision channel setup in UE.*

The amount of Transmission (sound going through an obstacle) is determined by the amount of Occlusion (0 to 1 or 0 to 100) set up on Room walls (Unity and UE), Surface Reflectors and

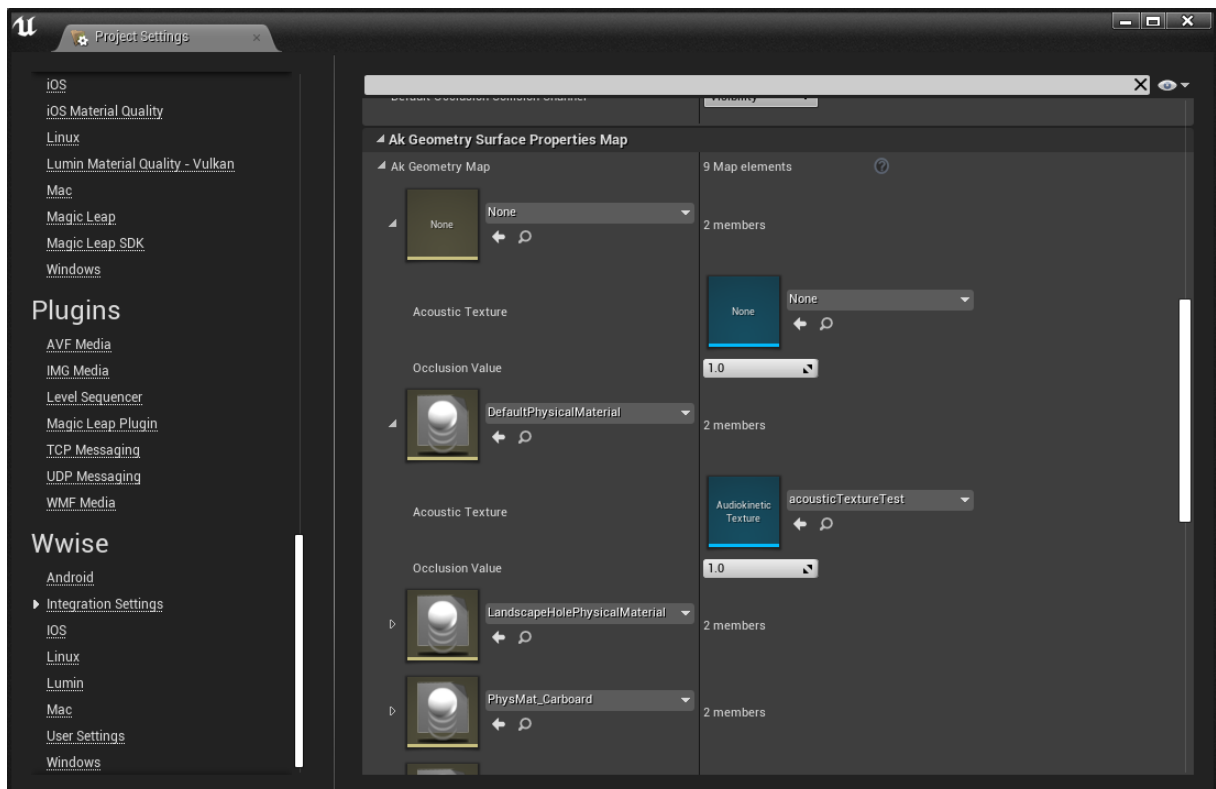


their collision mesh (global collision box of an object) or specific Acoustic textures on the geometry (AKSurfaceReflector in Unity, AKGeometry in UE), or Physical Materials (in UE).

Finally, same as Diffraction, the use of Transmission instead of Occlusion makes the setup easier and more precise in Unity (no layers, no fade time, different values on submeshes) and easier and more flexible in Unreal (independent from diffraction, no fade time, different values on geometry or materials).



*In Unity, Use the size value to get different occlusion values on each submesh of an object*



*Here in UE4, use the global geometry integration to associate Physical Materials, Acoustic textures and Occlusion values automatically in the project.*

Transmission is only used between room when there is no active portals (see below)

## Summary of Sound Propagation Features

The table below summarizes the features of Spatial Audio Rooms and Portals by grouping them in terms of acoustic phenomena, describing what Spatial Audio does for each, and how sound designers can incorporate them in their project.

Acoustic Phenomenon	Spatial Audio	Sound Design in Wwise
Diffraction of direct path	<ul style="list-style-type: none"> <li>• Obstruction or Diffraction built-in parameter</li> <li>• Modified game object (virtual) position</li> </ul>	<ul style="list-style-type: none"> <li>• Volume, filtering, or any property on Actor-Mixer</li> <li>• 3D panning, Distance attenuation</li> </ul>
Diffuse field (reverb)	<ul style="list-style-type: none"> <li>• Send to room's auxiliary bus</li> <li>• Constant power transitions</li> </ul>	Reverb, bus volume and game-defined send offset on Actor-Mixer
Room coupling: reverb spatialization and diffraction of adjacent room's diffuse field	<ul style="list-style-type: none"> <li>• Obstruction or Diffraction built-in parameter</li> <li>• Room object positioning and spread</li> <li>• Send to listener's room's auxiliary bus</li> </ul>	<ul style="list-style-type: none"> <li>• Volume, filtering, or any property on Bus</li> <li>• 3D panning of busses, reverb, bus volume and game-defined send offset of Auxiliary Bus to other busses</li> </ul>
Transmission (only when no active portal)	Occlusion	Volume or filtering on Actor-Mixer

-Rooms can now use room tones events that are spatialized on 360 degrees around the listener when entering a room. Like in 2019.1.7 these sounds will transition to point source emitters at the portals, when the listener is outside a room, in Unity. UE could not use integrated roomtone events with a Room until 2019.2.

### 3. Performance analysis (Unreal Engine and Unity tests)

All of this is great on paper, but what about performances? Is the audio result worth the setup and CPU/memory use, instead of using simple reverberation and global obstruction/occlusion? Is it usable in a project context with multiplying audio sources, even with only Spatial Audio usage for diffraction and transmission?

Map setup and audio content is archived in the Notes documents. Broadly, the scenes used are consisting of two interconnected Wwise rooms “inside” with a priority at 0, the biggest room being connected to the outside room, and occasionally an outside room with priority at -1 for exterior reverb and reflections. There is two portals in the basic configuration.

One Audio emitter is placed in each room. The sources are one mono audio file used in one play event in the Wwise project, to measure only the impact of spatial audio calculations evolution.

-UE 4.22 and Wwise 2019.1.4 compared to UE 4.23 and Wwise 2019.2

Key points:

-Performances comparison (CPU plugin and plug memory) with only rooms and portals, 3 or 12 sources for the Spatial Audio API with automatic management

/	Wwise 2019.1	Wwise 2019.2
<b>Rooms&amp;Portals 3 sources CPU usage Avg/Peak</b>	1.67%/2.27%	1.60%/6.89%
<b>Rooms&amp;Portals 12 sources CPU usage Avg/Peak</b>	2.50%/3.81%	2.30%/4.36%
<b>Rooms&amp;Portals 3 sources Total used Memory Avg/Peak</b>	6.9Mb/6.9Mb	6.2Mb/6.9Mb
<b>Rooms&amp;Portals 12 sources Total used Memory Avg/Peak</b>	7Mb/7Mb	6.4Mb/7.1Mb
<b>Rooms&amp;Portals 3 sources Spatial Audio CPU usage Avg/Peak</b>	0.25%/0.91%	0.25%/0.79%
<b>Rooms&amp;Portals 12 sources Spatial Audio CPU usage Avg/Peak</b>	0.32%/0.72%	0.6%/0.85%
<b>Rooms&amp;Portals 3 sources Spatial Audio Memory usage Avg/Peak</b>	132.9Kb/132.9Kb	145Kb/146.1Kb (geometry use is 141Kb)
<b>Rooms&amp;Portals 12 sources Spatial Audio Memory usage Avg/Peak</b>	156Kb/156Kb	149Kb/151.2Kb (geometry use is 141Kb)

- Performances comparison (CPU plugin and plug memory) for the Reflect plugin with second order reflections on 3 or 12 sources

/	Wwise 2019.1	Wwise 2019.2
Reflections 3 sources CPU usage Avg/Peak	5%/7.99%	2.6%/4.66%
Reflections 12 sources CPU usage Avg/Peak	16%/23.14%	3.5%/8.16%
Reflections 3 sources Total used Memory Avg/Peak	7Mb/7.2Mb	5.6Mb/7.2Mb
Reflections 12 sources Total used Memory Avg/Peak	7.9Mb/8Mb	6.9Mb/7.7Mb
Reflections 3 sources Spatial Audio CPU usage Avg/Peak	1.10%/2.45%	1.10%/2.7%
Reflections 12 sources Spatial Audio CPU usage Avg/Peak	5%/8.33%	2.35%/5.38%
Reflections 3 sources Spatial Audio Memory usage Avg/Peak	220Kb/228Kb	340Kb/1.5Mb (geometry use is 160Kb)
Reflections 12 sources Spatial Audio Memory usage Avg/Peak	280Kb/285.5Kb	280Kb/1.5Mb (geometry use is 160Kb)
Reflections 3 sources Reflect plugin CPU usage Avg/Peak	2.65%/3.5%	0.045%/0.10%
Reflections 12 sources Reflect plugin CPU usage Avg/Peak	8.5%/11.03%	0.10%/0.148%
Reflections 3 sources Reflect plugin memory usage Avg/Peak	1Mb/1.4Mb (Lower Engine Memory?)	85Kb/105Kb (Reflection/Diffraction memory)
Reflections 12 sources Reflect plugin memory usage Avg/Peak	2Mb/2.1Mb (Lower Engine Memory?)	110Kb/126.8Kb (Reflection/Diffraction memory)

- Performances comparison of obstruction/occlusion/diffraction/transmission with the outside portal closed and in-between room portal opened (and diffraction activated in both version of the audio engine)

/	Wwise 2019.1	Wwise 2019.2
Obstruction/Diffraction 3 sources CPU usage Avg/Peak	4.3%/6.44%	/
Obstruction/Diffraction 12 sources CPU usage Avg/Peak	10%/19.34%	4.7%/7.69%

<b>Obstruction/Diffraction 3 sources Total used Memory Avg/Peak</b>	6.5Mb/7.2Mb	/
<b>Obstruction/Diffraction 12 sources Total used Memory Avg/Peak</b>	6.9Mb/8Mb	13.4Mb/13.7Mb
<b>Obstruction/Diffraction 3 sources Spatial Audio CPU usage Avg/Peak</b>	0.75%/1.84%	/
<b>Obstruction/Diffraction 12 sources Spatial Audio CPU usage Avg/Peak</b>	1.4%/3.7%	2.2%/5.05%
<b>Obstruction/Diffraction 3 sources Spatial Audio Memory usage Avg/Peak</b>	200Kb/215.17Kb	/
<b>Obstruction/Diffraction 12 sources Spatial Audio Memory usage Avg/Peak</b>	233Kb/271.3Kb	275Kb/517.8Kb (Reflection/Diffraction paths memory is 81Kb/137.4Kb)

-Impact of the optimization of the diffraction between the old and new version (CPU average 12 sources)

We can see the CPU usage being more balanced in the 2019.2 version, with a Spatial Audio usage being a little bit more with these 12 emitters than 2019.1 (0.8% more on average) but the global CPU usage being reduced by a half. This is most likely due to the new logic of activating Diffraction by audio objects in Wwise and their status (active or virtual voices), as well as optimization on what to calculate when the player's position is updated, even with diffraction activated on reflection paths with the Reflect plugin (Reflect CPU usage still being around 0.12% in this scenario). (Considering this difference in performance is also to weight against the aesthetic result showed and discussed in the next part "Subjective result analysis".)

-Extreme situation performance optimization assessment (12 sources, and up to 13 portals)

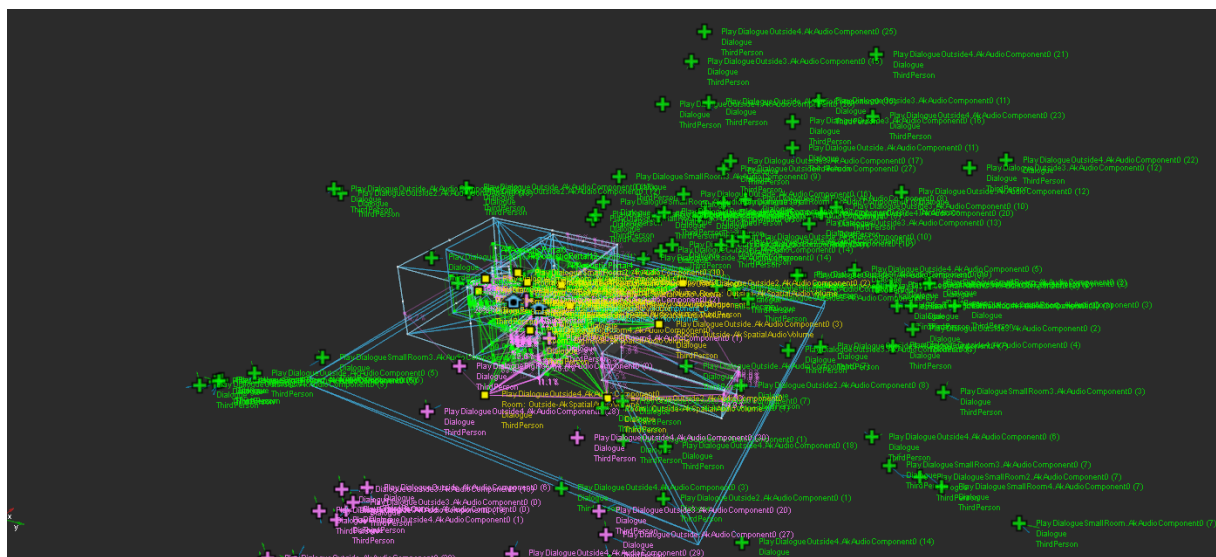
In extreme scenarios, the diffraction in 2019.1 and Unreal was making the game execution either slow down really badly or freeze after 4 active sources and going through a portal (with no room coupling for reflectors, and 13 opened portals even with the paths limited to 7). With room coupling for reflection/diffraction objects, and 13 portals, the data indicates a CPU usage by Spatial Audio at average 70-75% (but going as low as under 1%... so it is very unstable), and some peaks at 120% to 160%, and the memory used looks alright, peaking at 1.3Mb. After some voice starvation errors (my theory is too many virtual images for diffraction), Spatial Audio seems to stop working when entering the big room (the Listener is not even indicated as being in a room).

While this happens, the game is running at a very low framerate (around 10 with 8 portals, even lower but doesn't freeze with 13 portals), so in conclusion, it was unplayable with diffraction with that combination of too many portals and emitters.

In addition to the notes about Diffraction in 2019.1, the Reflect usage with the maximum number of portal and sources is around 3.5% (with peaks at 8%) outside, and 11% (with 13% at peak) within the rooms. It is substantial but not problematic in this scenario.

As for 2019.2, adding the full 13 portals and diffraction around obstacles within the rooms, the Spatial Audio performance never chugs the game's framerate performances, but goes to 8.5% on average and 22% at some peaks. (Global CPU usage is at 14% average with peaks above 25%, Reflect stays at its marginal sub 0.2% which is around 3% less than in 2019.1). Limiting reflectors to the rooms they are in reduces these numbers by 2 to 4%.

This is obviously far from the incredible range of CPU usage by the API in the 2019.1 tests, and the stability is also one of the big things. Considering this is not a likely scenario to have (12 sources being diffracted at the same time) in a project for mixing and clarity reasons, this indicate a pretty solid step in the optimization of these functions and their usability, even on a mobile platform.



*Example of an extreme scenario and its creation of diffraction paths and virtual images*

## -Unity 2018.2 and Wwise 2019.1.4 compared to Unity 2019.2 and Wwise 2019.2

Key points:

-Performances comparison (CPU plugin and plug memory) with only rooms and portals, 3 or 12 sources for the Spatial Audio API with automatic management

/	Wwise 2019.1	Wwise 2019.2
Rooms&Portals 3 sources CPU usage Avg/Peak	2.10%/5.55%	1.35%/3.48%
Rooms&Portals 12 sources CPU usage Avg/Peak	2.30%/5.18%	2.10%/3.91%
Rooms&Portals 3 sources Total used Memory Avg/Peak	6.5Mb/7.1Mb	7Mb/7Mb
Rooms&Portals 12 sources Total used Memory Avg/Peak	6.5Mb/7.1Mb	7.1Mb/7.2Mb
Rooms&Portals 3 sources Spatial Audio CPU usage Avg/Peak	0.08%/0.34%	0.09%/1.69%
Rooms&Portals 12 sources Spatial Audio CPU usage Avg/Peak	0.10%/0.18%	0.17%/2.11%
Rooms&Portals 3 sources Spatial Audio Memory usage Avg/Peak	142Kb/145.9Kb	275Kb/316.7Kb (geometry use is 149.5Kb)
Rooms&Portals 12 sources Spatial Audio Memory usage Avg/Peak	133Kb/135.6Kb	285Kb/325.7Kb (geometry use is 149.7Kb)

- Performances comparison (CPU plugin and plug memory) for the Reflect plugin with second order reflections on 3 or 12 sources

/	Wwise 2019.1	Wwise 2019.2
Reflections 3 sources CPU usage Avg/Peak	3.5%/15.38%	2.6%/4.66%
Reflections 12 sources CPU usage Avg/Peak	12.3%/23.30%	10%/25.54%
Reflections 3 sources Total used Memory Avg/Peak	6.6Mb/7.2Mb	5.6Mb/7.2Mb
Reflections 12 sources Total used Memory Avg/Peak	7.8Mb/8Mb	8Mb/9.2Mb
Reflections 3 sources Spatial Audio CPU usage Avg/Peak	0.25%/0.47%	0.13%/22%
Reflections 12 sources Spatial Audio CPU usage Avg/Peak	0.45%/2.19%	0.25%/3.26%
Reflections 3 sources Spatial Audio Memory usage Avg/Peak	175Kb/184.3Kb	325Kb/378.4Kb (geometry use is 162.4Kb)
Reflections 12 sources Spatial Audio Memory usage Avg/Peak	240Kb/262.8Kb	460Kb/594.9Mb (geometry use is 160Kb)

<b>Reflections 3 sources Reflect plugin CPU usage Avg/Peak</b>	1.6%/6.3%	2.7%/11%
<b>Reflections 12 sources Reflect plugin CPU usage Avg/Peak</b>	5.5%/17.44%	9%/22%
<b>Reflections 3 sources Reflect plugin memory usage Avg/Peak</b>	900Kb/1.4Mb (Lower Engine Memory?)	106Kb/151Kb (Reflection/Diffraction memory)
<b>Reflections 12 sources Reflect plugin memory usage Avg/Peak</b>	1.7Mb/2.1Mb (Lower Engine Memory?)	110Kb/126.8Kb (Reflection/Diffraction memory)

- Performances comparison of obstruction/occlusion/diffraction/transmission two portal opened (and diffraction activated in both version of the audio engine)

/	<b>Wwise 2019.1</b>	<b>Wwise 2019.2 4 and 4.3</b>
<b>Obstruction/Diffraction 3 sources CPU usage Avg/Peak</b>	5%/32.24%	4.10%/6.49%
<b>Obstruction/Diffraction 12 sources CPU usage Avg/Peak</b>	21%/140.33%	11.8%/25.35%
<b>Obstruction/Diffraction 3 sources Total used Memory Avg/Peak</b>	6.9Mb/7.3Mb	7.3Mb/7.5Mb
<b>Obstruction/Diffraction 12 sources Total used Memory Avg/Peak</b>	7.8Mb/8.1Mb	8.4Mb/9.2Mb
<b>Obstruction/Diffraction 3 sources Spatial Audio CPU usage Avg/Peak</b>	2.3%/29%	0.20%/1.7%
<b>Obstruction/Diffraction 12 sources Spatial Audio CPU usage Avg/Peak</b>	6.5%/129.3%	0.45%/2.05%
<b>Obstruction/Diffraction 3 sources Spatial Audio Memory usage Avg/Peak</b>	202Kb/219.3Kb	320Kb / 594.9Kb (Reflection/Diffraction paths memory is 100Kb/170.5Kb)
<b>Obstruction/Diffraction 12 sources Spatial Audio Memory usage Avg/Peak</b>	270Kb/302Kb	530Kb/597.9Kb (Reflection/Diffraction paths memory is 120Kb/170.5Kb)



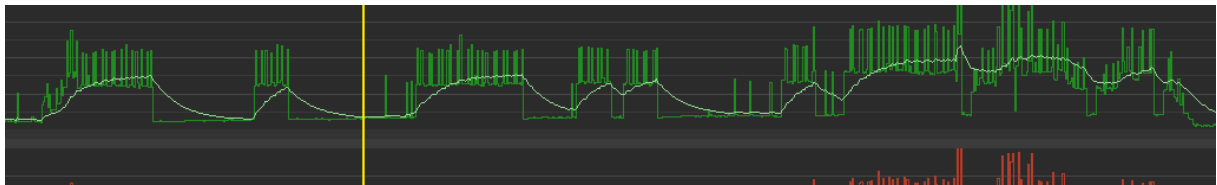
-Impact of the optimization of the diffraction between the old and new version (CPU average 12 sources)

Again, the CPU usage with diffraction activated seems more stable than in the 2019.1 version. The optimization data shows Reflect use being roughly the same if you don't take into account the extreme values that happen once (in 2019.1 the plugin ranges from 0.2% of CPU to 12, and 05 to 22% in 2019.2), but the Spatial Audio CPU values are extremely lower in the new version, especially on average. The other data stays pretty similar in this Unity test between the two versions, only memory augmenting significantly but as noted in the Reflect plugin measures part, the documentation was not clear were the memory for these calculations was counted before.

(Considering this difference in performance is also to weight against the aesthetic result showed and discussed in the next part "Subjective result analysis".)

-Extreme situation performance optimization assessment (12 sources and up to 15 portals)

Using 2019.1, while the scenario in which all 15 portals and 12 emitters were playing sound and calculating diffraction at the same time was not freezing Unity's execution, the limit of an already high enough voice maximum of 256 in Wwise was not enough. The Voice starvation errors were piling up. When trying to limit the reflections on the geometry to the room the objects are situated, the data can show results, but very unstable. The Spatial Audio usage was going between 10 and 311% of the CPU, with frequent peaks at 60/75% even if the average value on the graph don't go this high.



Reflect plugin CPU usage varied between 3 and 15% in this scenario, somewhat equivalent to the results shown in the Unreal tests and the test focusing on the Reflect plugin.

Using 2019.2, the results were not as drastic as using Unreal: the Unity execution was not feeling slowed down, but the Wwise engine shows a high CPU usage of around 23% average and peak at 60%, especially when outside with all 12 emitters activated and the 15 portals opened. The biggest culprit is the Reflect plugin using 8 to 50% CPU, and makes the values change rapidly when in the 12 emitters activated situation, a bit like with 2019.1 but with less range.

When optimizing a bit and limiting reflectors to the rooms they are in, the values of the averages get down a bit (around a 3% difference), but the peaks stay the same as before, especially with the Reflect plugin going around 25%-30% up to 50 again, when the listener is getting in the outside room.

Quite strangely, the optimization using Unity seems to be affecting mainly the API and not the Reflect plugin even if there is less big values of CPU usage in these extreme tests, and it was the opposite in Unreal with number higher for the API than the Reflect plugin. This is not the subject of this document, but it would be interesting to know why such differences exist, if it is limited by the way the two engines operate (and operate differently).

## Subjective result analysis

In this part, the goal is to do an aesthetic comparison between using the “simple” obstruction/occlusion system based on direct paths raytracing, and the diffraction/transmission system based on virtual emitters with paths around geometry.

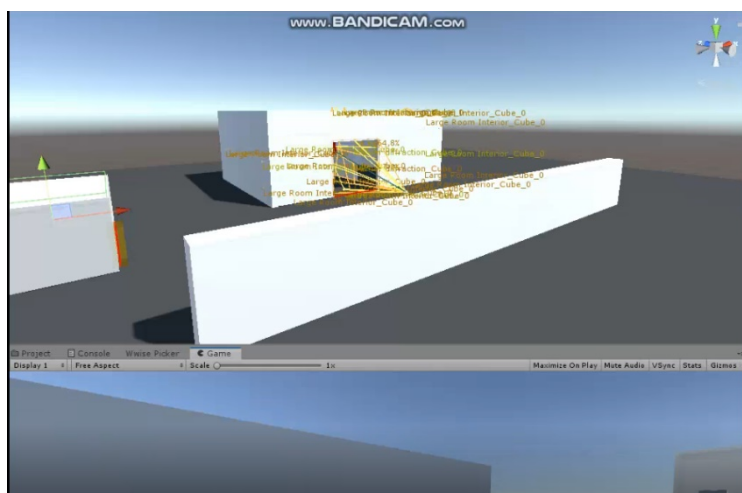
The setup for this comparison is using the default obstruction/occlusion curves, and the default threshold for updates (0.5s for obstruction and 1 engine unit of movement for diffraction). The scene is consisting in 3 rooms (an exterior, a hangar and a smaller room), the same Reverb sends in each room between Wwise versions, and a main bus using Auro 11.1 channel configuration with the Auro3D plugin to be able to hear elevation changes using headphones.

Additionally, the sound emitter (a helicopter sound for its frequency range usage to be heard easier in this comparison) is sent through some amount of Reflect plugin to get a bit more density in the reverbs on both sides of the test. Nonetheless, it is worth underlying again that with these effects, only the Reflect plugin for early acoustic reflections is a plugin you must pay a license for. The Diffraction system is a part of the Spatial Audio API available to everyone using Wwise, like Rooms and Portals.

The test will make the player move around the same spots in the scene to hear the results:

- Around an object, thin (a wall) then bigger (around a building)
- In the same room as the emitter, then in different rooms to hear the sound travel through portals
- Finally moving the emitter in a different room

The video files are “Unity Wwise 2019.1 obstruction test” and “Unity Wwise 2019.2 obstruction test” in \Wwise 2019.2 report screenshots and profiling files\Helicopter obstruction test.

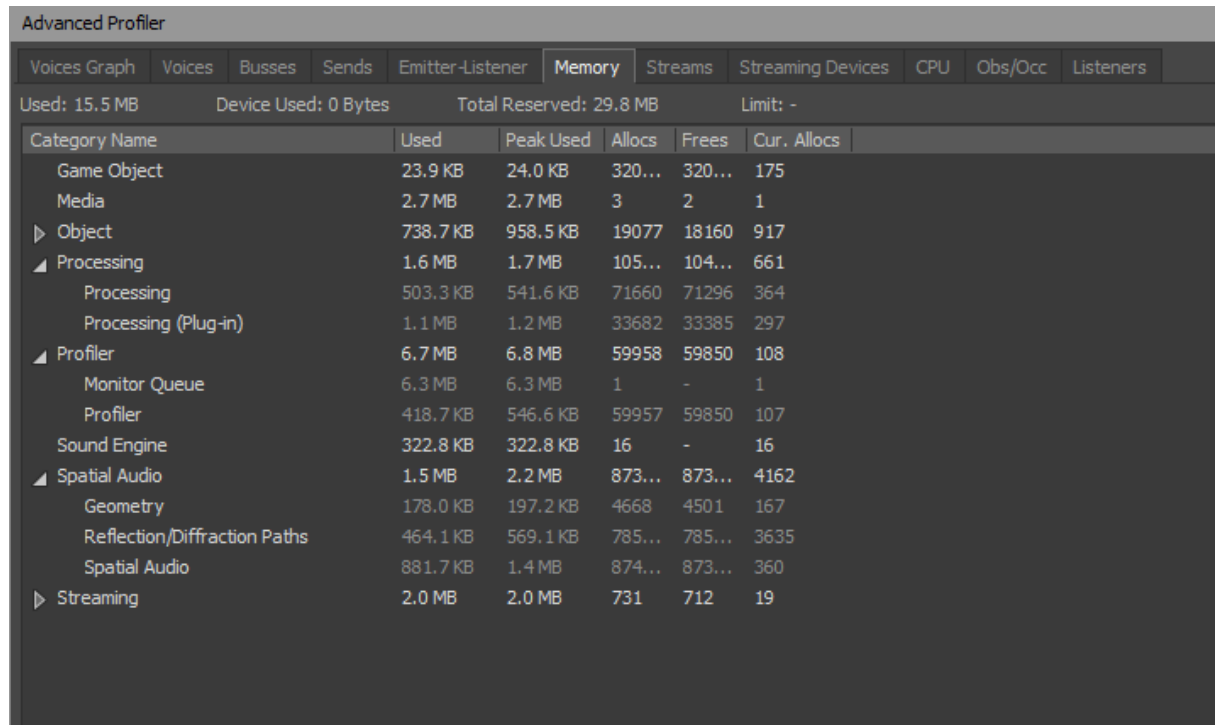


You can hear a more natural sounding result with diffraction in Wwise 2019.2, even in the default setup. The time gap between the direct path and the virtual emitter being created can still be heard at 0:42, further optimization of the obstruction curves and movement threshold would be interesting to follow in order to evaluate these types of scenarios.

When hearing diffractions around the wall outside, you can see and hear some leaking underneath the obstacle, showing how you would have to carefully setup your audio geometry to use this kind of solutions. You can also notice that the 2019.2 version is indeed respecting the attenuation distance from the source through portals, unlike 2019.1 that adds the portals individual attenuations.

#### 4. Profiler(s) Update

Linked to the optimization in CPU and memory usage for several Wwise systems, the profiler data is now better organized and detailed about these processes. Especially the Memory tab, which is clearer now about what is used:



The screenshot shows the 'Advanced Profiler' interface with the 'Memory' tab selected. At the top, it displays summary statistics: 'Used: 15.5 MB', 'Device Used: 0 Bytes', 'Total Reserved: 29.8 MB', and 'Limit: -'. Below this is a table with the following columns: 'Category Name', 'Used', 'Peak Used', 'Allocs', 'Frees', and 'Cur. Allocs'. The table lists various categories and their corresponding memory usage and allocation statistics.

Category Name	Used	Peak Used	Allocs	Frees	Cur. Allocs
Game Object	23.9 KB	24.0 KB	320...	320...	175
Media	2.7 MB	2.7 MB	3	2	1
▶ Object	738.7 KB	958.5 KB	19077	18160	917
▲ Processing	1.6 MB	1.7 MB	105...	104...	661
Processing	503.3 KB	541.6 KB	71660	71296	364
Processing (Plug-in)	1.1 MB	1.2 MB	33682	33385	297
▲ Profiler	6.7 MB	6.8 MB	59958	59850	108
Monitor Queue	6.3 MB	6.3 MB	1	-	1
Profiler	418.7 KB	546.6 KB	59957	59850	107
Sound Engine	322.8 KB	322.8 KB	16	-	16
▲ Spatial Audio	1.5 MB	2.2 MB	873...	873...	4162
Geometry	178.0 KB	197.2 KB	4668	4501	167
Reflection/Diffraction Paths	464.1 KB	569.1 KB	785...	785...	3635
Spatial Audio	881.7 KB	1.4 MB	874...	873...	360
▶ Streaming	2.0 MB	2.0 MB	731	712	19

Further possibilities are available to track and filter objects, audio playback and routes, with the new Filter bar at the center of it.

All objects are now automatically added to the Game object profiler tool, as well as the Game Sync monitor in this same view. It was before a case of finding what you wanted to track, set it up, and look at data. Now all the data is there, filtering is the only thing to do to focus on a specific part of the audio that is playing. The values of pinned and filtered game object in the Game Object profiler are also now retained between session, so you can repeat your profiling without losing the focus of your search.

The new filter bar can have its settings shared between all profiling views (Voice Monitor, Voice Explorer, Voices Graph, Voices, Game Sync Monitor, Game Object 3D Viewer), or being applied locally, at will. The main filtering functions are:

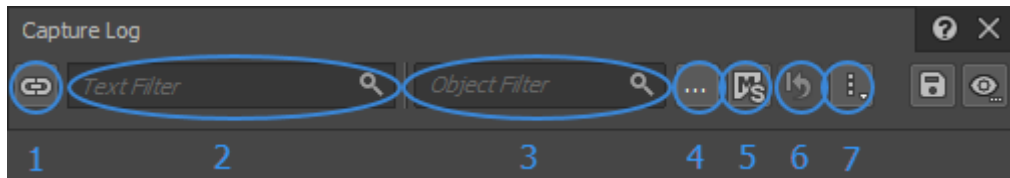
-Unlink filters: the function used to enable or disable the filtering options to apply only to this profiling view or all profiler views (1)

-Text filter: Enter text to filter content. Words typed will be compared to the beginning of names found in the Wwise content, and more words refine the results (this is not case sensitive). You can enter expressions depending on what you type and Wwise can search for, or enter advanced expression if needed (rules are detailed here [https://www.audiokinetic.com/fr/library/edge/?source=Help&id=using\\_profiler\\_filter\\_expressions](https://www.audiokinetic.com/fr/library/edge/?source=Help&id=using_profiler_filter_expressions) ) (2)

-Object Filter and the button for browsing objects: Browse for objects in the Wwise project and display the names there. The filter will then use the Wwise content and their parent-children relation to filter the data displayed in the profiler (3 and 4)

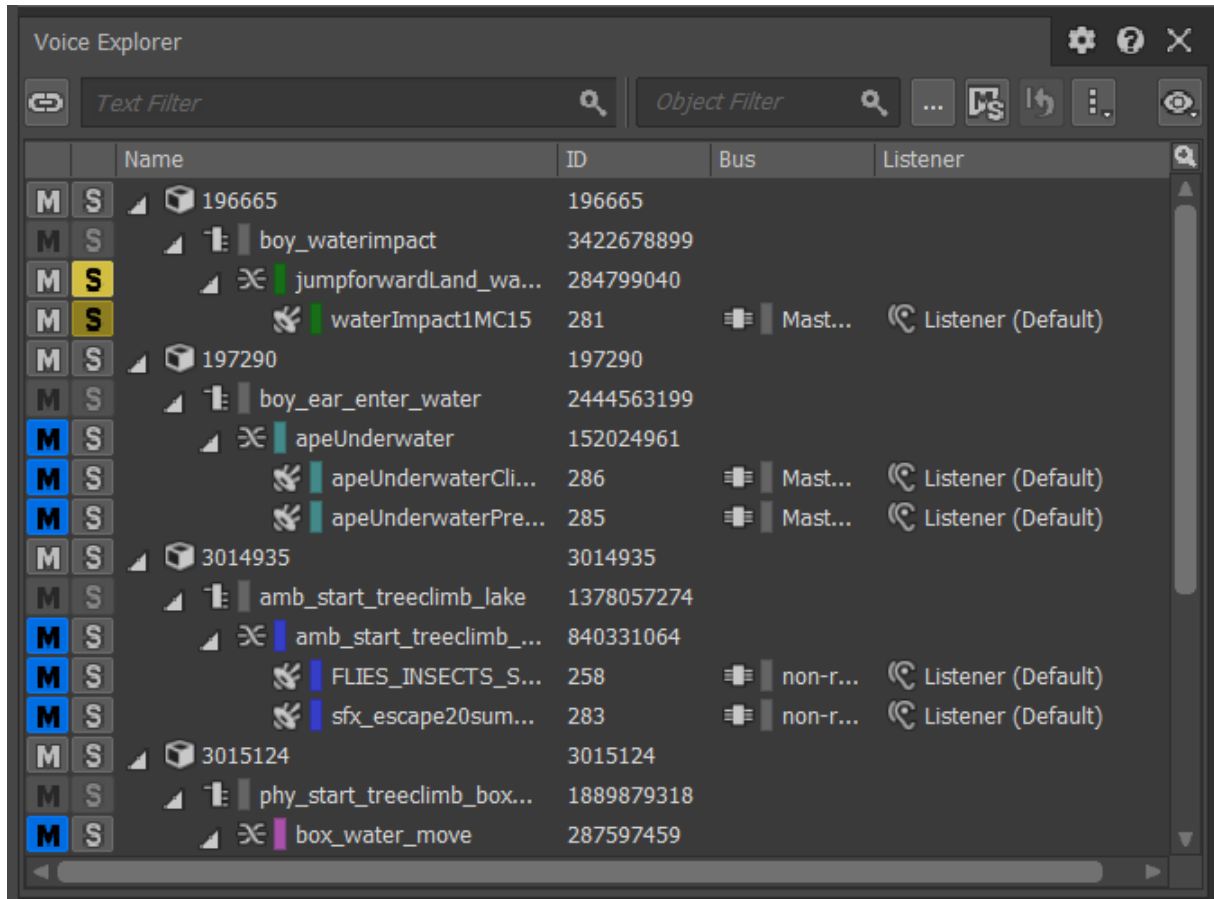
-Mute/Solo: When clicked, the objects with Mute activated are removed from the results, and the objects with Solo activated are displayed exclusively. This is useful to only see the data about what you selected to hear when profiling a game (5)

-Reset filter: Clear all filters (6)



Additionally, a new profiler view has been added, named “Voice Explorer” (Views>Profilers>Voice Explorer).

It allows for a display of the current playing voices at a specific time in the capture session, and is organized by playing instances from the project and an event being posted. Two PostEvents of the same event will display two instances in this Voice Explorer. You can organize the rows to see data about the lines, that are displaying with drop-down lists the instances, then the path to the audio object through the Wwise project.



/////

Links:

The updated Spatial Audio detailed functions

[https://www.audiokinetic.com/fr/library/edge/?source=SDK&id=spatial\\_audio.html](https://www.audiokinetic.com/fr/library/edge/?source=SDK&id=spatial_audio.html)

Migrating a project from 2019.1 to 2019.2

[https://www.audiokinetic.com/fr/library/edge/?source=SDK&id=whatsnew\\_2019\\_2\\_migration.html#migration\\_19\\_2\\_spatial\\_audio](https://www.audiokinetic.com/fr/library/edge/?source=SDK&id=whatsnew_2019_2_migration.html#migration_19_2_spatial_audio)