Bug Report [SOLVED] – 09/31/2021 – C++ Core Audio Engine and C# integration

*Release version 1.0*

*The At\_AudioEngine has a new file structure. Everything is on a single folder.*

|  |  |
| --- | --- |
|  | *The .unitypackage do not provide the C++ library for the core engine of the spatializer. You must copy the folder “\x86\_64” with the file “AudioPlugin\_AtSpatializer.dll” in \Assets\At\_3DAudioEngine\Other\Plugins\.*  *You also need a folder named “Resources” in the “Assets” folder. It is where json file containing the path for “Audio Extern Asset Folder” et “States Extern Asset Folder” is saved.* |

Bug Solved with this version :

* Loose of audio file path : you can now safely duplicate, copy, paste, drag and drop a prefab
  + /!\ the attribute values of the players are not saved in the prefab. So the usefulness of a prefab is very reduce for the audio engine.
  + /!\ we use a method for cleaning the json file for state. It compares the players in the scene and the players in the json file and delete the reference of the players whose do not exists in the *scene.*

|  |  |
| --- | --- |
|  | You can use the button “Clean States file” in the At\_MasterOutput GUI, but this should not be necessary because the cleaning method is called quite often during editing. |

* + /!\ /!\ Cleaning use the method “FindObjectsOfType<>()“ to list the object in the scene. So if a GameObject with a player is not active, we consider it does not exist. So its reference are deleted in the json file, causing the loose of all attributes including file path.
* Different scenes using the At\_AudioEngine can now be created safely. All the json file are saved in the *States Extern Asset Folder* with the name “{Scene Name}\_States.txt”.
  + /!\ I am waiting for the serialization system from the developers of the team, which should provide paths for States and Audio Extern Asset Folder and an “Event” for loading all the engine states. For now, all the states for each scene are loaded and store in a dictionary in the constructor of the static class “At\_AudioEngineUtils.cs”. The engine try to read all the .txt file in the “*States Extern Asset Folder”*
  + /!\ You must use different names for each scene. If you change the name of a scene, you will loose all the attributes of your players and master output because a new json file with a new name will be created. You can reverse it by manually changing the name of the original json file.
* The public boolean “isPlaying” of the At\_Player class is now true ONLY if the player is really playing (i.e. actually reading the sample of audio file and feeding the output buffer)

------------------------------------------------ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ------------------------------------------------

* You must call a new method to trigger a player with the At\_DynamicRandomPlayer component.

public void AddOneShotInstanceAndRandomPlay(bool isRandomPosition, Vector3 position)

* + if isRandomPosition is true, it use spawnMinAngle, spawnMaxAngle and spawnDistance to randomly generate a position.
  + if isRandomPosition is false, it use the 3D position given as second argument
* Because the spatializer uses the distance of a player from each virtual microphone to apply a delay to the audio source, we define a “maximum distance” which give the size of the buffer used for this delay (i.e. a maximum delay). GameObjects with a 3D At\_Player component should not be beyond this distance for each virtual microphone.

|  |  |
| --- | --- |
|  | We use Gizmos to draw a red circle for each virtual mics.  In the case illustrated here, the player should then always be inside the white circle.  You can change the “Max distance for Spatializer” to fulfill your needs.    Be careful, this value directly impacts memory usage.  Example : for a value of 37.8m and a sample rate of 48kHz, the engine create a float array of 6144 samples for the delay. A sample is 4 bytes, so this represent approx. 24Kbytes of memory |