Decompression and assets loading

Files reading:

I don't think we have any advantages using unreal engine .uasset to store sound data. We cannot edit it there because the changes need to be synchronized with workstation. But by unify the approach to files streaming over all XJ softwares we wouldn't need to support multiple versions of one stuff and encounter with different problems for different environments. What I propose should be implemented as ideal inside XJ library.

Points:

- We prepare compressed files by ourselves. For example each template can have one batch file compressed files.

- XJ library starts decompression and return to plugin memory pointer to where the data is located.

- Engine plugin storing compressed data inside it's asset type. So engine now need to store only one .xj asset where all the information stored.

- Such library as Oodle can be used in C++ to help with compression and decompression.

Streaming:

Solutions:

1. **All in memory:**

    1.1. All the audios is preloaded into memory. We decide which audios to load by which template we start with.

    1.2. To make loading process faster we can start loading only audios that are needed by the minimum value of intensity.

2. **Do not pre decompress rely on speed:**

    2.1. XJ do not perform decompression on anything that do not need right now to play.

    2.2. When we need audios that still isn't ready we start playing last segment from start and here is crucial role playing is how fast decompression we can make.

2.3. Advantages is that we wouldn't use much memory at all we will able to make some cache memory for recent assets, but in general we will delete from memory already used assets. But for sure if the computer isn't coop with game it's will lead to problems of audio looping too long and it's

3. Compomise

3.1. XJ operates with two options preloading all the audio from template into memory or using decompression right before we need audio. When performance is changed switch is occurring on runtime and can lead to freeing memory because we switching to decompression on run or allocating memory to have all the preloaded data.

3.2. XJ library have a performance manager that measure how current system is coop with current method of streaming assets.

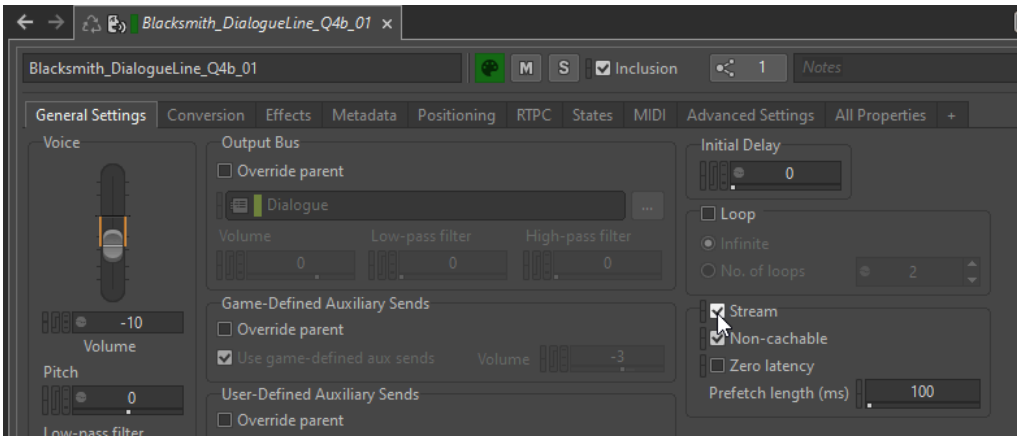
3.3. XJ use intensity as one type to improve performance. Intensity change by performance manager when decreased performance is detected.

4. Other ideas

4.1. Stream trigger point. When in game taxonomy change is depend on distance we can create trigger point at that place and audio only will be preloaded when we will be near it.

4.2. Inside workstation we can set the option on audio to allow **stream from file**(when the track only will be loaded when its need to played), **no latency**(to preload file from the start and avoid any delay in playing it).

This for example what options Wwise have:



- **Non-Cachable:** Leaves no cache when streamed audio plays. Good for long loops or non-recurring sounds that have no reason to stay cached in memory after completion. In WAG, this option is selected for all dialogues because they usually only occur once.

- **Zero Latency:** Eliminates the latency of streamed files by creating a small buffer of the beginning part of the audio file. Useful for the music in WAG because two themes are playing concurrently and, as such, the timing of the triggering is essential.
- **Prefetch length (ms):** Refers to the milliseconds of time you'd like to store in the memory. With everything in WAG this is left to its default of 100 milliseconds.

And FMOD:

Each asset have a loading mode:

- **Compressed** - Assets are loaded into memory in compressed format when loaded, and then decompressed when played. This requires less memory than decompressed loading mode, but requires slightly more CPU when playing assets.
- **Decompressed** - Assets are completely decompressed into memory when loaded. This uses more memory than streaming loading mode and compressed loading mode, but requires very little CPU when playing assets. This loading mode is most commonly used for short assets that are played frequently and for mobile platforms.
- **Streaming** - Assets are continuously loaded piecemeal from disk into a small memory buffer while playing. This substantially reduces the memory required to load and play assets. However, most platforms can only support a single-digit number of simultaneously playing streams, as streaming assets require constant disk access while playing. Streaming loading mode is most commonly used for music and background ambiences.

Also FMOD have an options to automatically stream audios that exceed some time amount:

References:

- FMOD
- Wwise