GRUMBLE LABS
ADAPTIVE MUSIC PLAYER
V1.0

December 8, 2015

The Grumble Labs Adaptive Music Player is a free Unity asset written in C# that allows for a simple implementation of audio with multiple synchronized layers. It also includes a collection of songs (with the license to use these songs in your game) which have multiple layers and are ready to be inserted into your project.

The Adaptive Music Player is built off of the concept of synchronizing complete and mastered audio files and fading between them to simulate the song changing character. This approach allows a more simple implementation, fewer audio assets, and allows you to always hear mastered audio without the need for a complex set of DSP in a mixing bus. It does not support an approach of loading in multiple tracks of individuals instruments to create a mix. It also fully supports the ability to play simple non-adaptive music.

It also includes a simple example scene that shows how a basic implementation would work.

FILES

grumbleAMP/	- Adaptive Music Player Dir
Docs/	- Documentation Files
grumbleAMP Example Scene/	- Example Implementation
grumbleAMPExample.cs	- Example Script
Editor/	- Custom Inspector Dir
grumbleAMPEditor.cs	- Custom Inspector Script
Music Tracks/	- Contains Music Tracks
Scripts/	- Contains Main Scripts
grumbleAMP.cs	- Main Script
grumbleSong.cs	- Song Container Script

WORKFLOW

The workflow is purposely kept as simple as possible, and is as follows:

1. Load the Grumble Labs Adaptive Music Player asset into your project.
2. Add the grumbleAMP script as a component to a persistent GameObject.
3. Configure your music in the inspector window for the grumbleAMP script.
4. Create a public grumbleAMP property in one of your own classes.
5. Directly reference this property to the grumbleAMP script's GameObject.
6. Use the grumbleAMP's public functions to control your music.

PUBLIC FUNCTIONS

PLAY A SPECIFIC LAYER OF A SONG (Bool returns true if something goes wrong, int returns the song number if it finds the song, or -1 if something goes wrong.)
public bool PlaySong (int songNumber, int layerNumber, float introFadeLength = 0f)
public int PlaySong (string songNumber, int layerNumber, float introFadeLength = 0f)
public int PlaySong (string songNumber, string layerNumber, float introFadeLength = 0f)
public int PlaySong (int songNumber, string layerNumber, float introFadeLength = 0f)

PAUSE WHATEVER IS PLAYING (Returns true if nothing was playing)
public bool Pause ()
public bool Pause (float time)
public void UnPause ()

STOP ALL PLAYERS
public void StopAll (float fadeOutTime = 0f)

CROSSFADES TO A NEW SONG, WITH SPECIFIED LAYER (Bools return true if something goes wrong. Int returns song number or -1 if something goes wrong.)
public bool CrossFadeToNewSong (int songNumber, int layerNumber = 0, float crossfadeTimeUser = Mathf.Infinity)
public bool CrossFadeToNewSong (string songName, int layerNumber, float crossfadeTimeUser = Mathf.Infinity)
public int CrossFadeToNewSong (string songName, string layerNumber, float crossfadeTimeUser = Mathf.Infinity)

CROSSFADES TO A NEW LAYER OF THE CURRENT PLAYING SONG
(Returns true if something goes wrong.)
public bool CrossFadeToNewLayer (string layerName, float crossfadeTimeUser = Mathf.Infinity)
public bool CrossFadeToNewLayer (int layerNumber, float crossfadeTimeUser = Mathf.Infinity)
public bool CrossFadeToNewLayer (int layerNumber)

RETURNS THE CURRENT SONG NUMBER
public int getCurrentSongNumber ()

RETURNS THE CURRENT LAYER NUMBER
public int getCurrentLayerNumber ()

RETURNS TRUE IF THE SPECIFIED SONG NUMBER IS SET TO LOOP
public bool getLoop (int songNumber)

SETS THE SPECIFIED SONG NUMBER TO LOOP OR NOT LOOP
public bool setLoop (int songNumber, bool loopOn)

RETURNS TRUE IF SOMETHING IS PLAYING
public bool isPlaying ()

RETURNS TRUE IF SPECIFIED SONG NUMBER IS PLAYING
public bool isPlaying (int songNumber)

RETURNS TRUE IF SPECIFIED SONG NUMBER AND LAYER NUMBER IS PLAYING
public bool isPlaying (int songNumber, int layerNumber)

RETURNS A FLOAT BETWEEN 0F AND 1F OF THE RESOURCE LOADING PROGRESS
public float getResourceProgress ()

RETURNS A FLOAT BETWEEN 0F AND 1F OF THE CURRENT GLOBAL VOLUME
public float getGlobalVolume ()

SETS THE GLOBAL VOLUME BETWEEN 0F AND 1F
public bool setGlobalVolume (float newVolume)

SETS THE GLOBAL CROSSFADE TIME TO THE SPECIFIED VALUE
public void setGlobalCrossFadeTime (float newTime)

RETURNS THE GLOBAL CROSSFADE TIME
public float getGlobalCrossFadeTime ()

ADVANCED FEATURES

The Grumble Labs Adaptive Music player works in two basic modes. One is the "normal" unnamed mode which requires all audio files to be drag-and-dropped into the fields in the Adaptive Music Player inspector window. The other is the Resource Request Mode, which can be enabled by ticking the box near the top of the inspector window which says "Resource Request Mode". In Resource Request Mode, each song is defined by a string of the file name and must be located in a Resources folder. The files will load asynchronously after the scene starts, and the progress of which will be a float value between 0f and 1f which can be accessed with the .getResourceProgress() function.

If you try to play a song before all the layers are loaded, it will get queued to play and start as soon as the layers have loaded. Alternately, you can flag each specific music file to "Load In Background" in the normal mode, but you aren't provided with a way to measure how many of your resources are loaded, and trying to play a song without every layer loaded may result in disasterous behavior.

EXAMPLES

If you were to assign song layers in the grumbleAMP inspector and then create this property in your script and reference it to a single instance of the grumbleAMP script (it is not static):

public grumbleAMP gA;

Then you could call:

- Play Layer 2 of Song 3 with a fade in of 500 ms.
gA.PlaySong(3,2,0.5f);

- Play Layer 2 of Song 3 without a fade in.
gA.PlaySong(3,2);

- Crossfade to a layer named "evil" over the course of 2 seconds
gA.CrossFadeToNewLayer("evil",2f);

- If anything is playing, perform a block of code

```
if ( gA.isPlaying() ) {
        doSomething();
}
```