Assessment Task 1 – Write a Modular Complex System Brief

Audio Manager – Julian Pahor

Purpose:

The purpose of this system is to greatly streamline the implementation of audio within Unity. Making it easy to essentially plug and play audio from multiple sources and easily change how they interact with one another.

Functionality:

The system relies on a chain of scriptable objects for the singleton AudioManager to call from.

The first chain refers to AudioFiles, AudioFileBundles + AFEvent scripts:

These are for any audio clips that serve the purpose to be fired off once or in a repetitive manner. The AudioFile object allows the user to select a specific imported audio clip and change its local volume, and pitch, as well as some basic sound editing in the form of fade-ins or outs. These all apply to this individual audio clip. The user can then group these individual audio files into a bundle where they can change settings that refer to, slight volume/pitch changes (for the clips not to sound repetitive), and whether or not they want the clips to randomize their order on repeat.

The user can then use the component AFEvent to attach to whatever game object they want to be emitting the sound, this gives the user options on when they want to fire audio events. Whether that be on default unity values such as OnStart, OnEnable, OnDisable, etc... Or to specific function calls on their own scripts, using UnitysEvent system to do so.

The second chain refers to AudioLoops, AudioLoopBundles + ALEvents scripts:

These provide almost identical functionality but are more focused on audio clips that will be looping themselves in the background. Such as music or ambiance.

I want to also open up the unity audio bus routing through this system but have less experience working in script with Unitys audio bus fx. I would love to implement this as it opens up a ton of creativity for the users to play with but envision this more as a stretch goal at the current time.

Some of the references I've been using to think up a system that works well are the two below.

References:

https://github.com/jackyyang09/Simple-Unity-Audio-Manager

https://assetstore.unity.com/packages/tools/audio/master-audio-2022-aaa-sound-212962

Reliant Libraries:

This system will primarily use Unity's default libraries.

I will also be looking into using Unity's burst compiler to try and optimize the code to its fullest.

Mathematical Operations:

Some of the mathematical operations I will be using are for the following specific cases:

Fade In + Out applications:

Fade In + Out will be input by the user by a simple float time for however long they want the fade to be. (This will be restricted by half the total time of the clip they are trying to implement. So technically they could have the entire first half of the clip fade in and then the second half fade out.) This will calculate the corresponding sample at the given time the user wants the fade to end/start, and then apply a logarithmic fade from the beginning/end of the clip to that sample.

Sample Accurate Looping:

This involves calculating how long the clip is based on its sample count and frequency and scheduling it to play immediately after it is finished, using the Unity AudioSource.PlayScheduled function. I've read articles that Unitys audio source looping is not the best so I wanted to implement this sample accurate looping, but will do some stress testing to see if it's worth using overall.

<u>Advanced Algorithms:</u>

Wave Pixel Drawing (For Inspector):

This algorithm creates a new texture, to be used for the inspector preview of the given audio file, and reads the pulse code modulation data out of the audio clip to draw a picture of the wave onto that texture.

Non-Repititive Audio Clip Looping:

The AudioFileBundle scriptable object is a group of audio clips that are expected to be fired repetitively. A group of footstep sounds is the best example of this. This algorithm will make sure that the same audio clip will never play twice in a row, as this can be heard quite easily by the player and could potentially ruin the immersion of a realistic audio environment. It creates a randomized order for the clips to play in and when it reaches the end it creates another randomized list checking that the last clip played doesn't match the first.

AudioManager Asset Searching:

This is an algorithm that exists on the AudioManager instance that will automatically search through the user's asset folders for any related scriptable objects they have created to work with this audio system. It will fill its inspector with them so the user has easier access to find them and see if they are missing any.

<u>How it will be made modular + Integration:</u>

It will be implemented as a package that can be pulled into any unity project with a thorough read.me guide to introduce all the functionality provided. I am also considering adding some automated audio bundle creation using Resources.Load and detectecting correctly named audio files to automatically group them together in scriptable objects.