Installation in Unity

Step 1. Registering the game

Add the Unity package to the game -

Create an empty SAWDAudioManager game object and add the SAWDAudioManager script to it.

For now, Unity can be minimized.

Create a game in Games by entering a title.

Enter game data - name (if you want to change :)), description, logo.

Remember your game ID.

Step 2. Create base packages

Create your base audio package.

Each audio pack can contain any number of sounds, voiceovers, and anything that has sound.

However, we assume that in basic audio packages you will separate sets of sounds according to some logic.

For example, there are three packages in Example:

1. Base package “package name for people” – created by developer
2. Sub package “Someone pack 1” – created by translator
3. Sub package “Someonce pack 2 on different language!” – created by another translator

Add tags:

For sounds that will be used by arrays, simply create a tag (check example on site).

For sounds that need to be filled in individually, create a subtag based on the main tag.

To do this, select the main tag by mouse click on site and the field for creating subtags will open. (check example on site)

A single click on a tag selects the tag to create subtags.

Double clicking on a tag allows you to rename the tag.

Fill in the list of characters: nothing is being done at the moment, but it is expected that it will be possible to change one voice of a character to another voice.

Step 3: Create Basic Sounds

There are two options here: add a separate sound or fill the sounds with a group.

If you select tags and subtags, then when pouring sounds, they will immediately be created with these tags and subtags (neither tag nor subtag must be selected in this case).

Fill in all the sound options for which you want to create the ability to load other options.

You don't have to upload the sounds themselves here, just the generated string will suffice, but when other voiceovers do the translation, they won't be able to hear the original unless you upload the sound.

Fill in a tag for each sound and, if necessary, a subtag, and optionally enter the voiceover text.

Step 4. Integrate into Unity

We return to Unity.

Enter your Game ID (can be found in your game description on site) in the SAWDAudioManager gameObject you created in step 1.

In order for the sounds to change at runtime, we have two ways:

1. Where there is one AudioSource with one sound - add SAWDAudioSourceHelper, enter Sound_tag - base tag and Sound_sub Tag - base subtag.

Specify the audio source (although it will be pulled up automatically if it is on the same game object)

2. Where you have a lot of sounds that are filled with an array, dictionary or something else like this:

Open the Example file and see the example.

Are you only interested in these two functions

SAWDAudioManager.SubscribeOnAudioLoad("Puck hit", ApplyNewAudios); // subscribe to change the sound (Puck hit is a global tag)

SAWDAudioManager.UnsubscribeAudioLoad("Puck hit", ApplyNewAudios); // unsubscribe from changing the sound (Puck hit is a global tag)

and an example of how to get a list of sounds.

In order to populate your array/dictionary define any callback function. This function will be called every time new sounds are loaded and every time a subscription is made.

```
private async void ApplyNewAudios(List<Audio> audios)
{
    queue.Clear();
    // subtags not provided or ignored!
    for (int i = 0; i < audios.Count; i++ )
    {
        queue.Enqueue(await audios[i].GetClip());
    }

    // or in case you have subtag
    // If there are no subtags, it is considered that files are uploaded by this tag in any amount, otherwise one file is one subtag
    // Only the developer determines how the files will be uploaded, so if you need a lot of sounds like "sword strike" - do not put subtags on them.
    // If you need a lot of sounds like "training" - put some tag "training" and inside each phrase - your own subtag like "welcome", "jump" and so on and so forth

    // todo: uncomment to check how it's work
    // for (int i = 0; i<audios.Count;i++)
    // {
        // soundsDictionary[audios[i].sub_tag.title] = audios[i].clip;
    // }

}
```

Next, use your sounds the same way you used them before. They are simply AudioClip arrays now.

If you have already implemented sounds in the game - everything is the same, do not change your code in any way, but only add a sound subscription so that your sounds are replaced by new loaded sounds.

Step 5. Integration into the unit, part 2

To download a new audio package, the user needs an interface with a choice of audio packages :)

Here the block from the example will help us - AudioManagerUI (you will have to design it yourself or take it as it is).

Just look at the example scene - Example.

The initial packages are filled in as a string from the id separated by commas in the variable SAWDAudioManager.audio_package_ids = "4,14,19"; // this is the same as downloading packages 4, 14 and 19

When the user selects checkboxes, he simply fills in this line with IDs.

When you click on the Apply button, the download of sounds from the server starts.

As soon as the sounds from the packages are loaded - they are saved to the device and wherever there is an active subscription to the tag - they are forwarded to the callback.