VisSoundSync - Subtitle Audio Sync

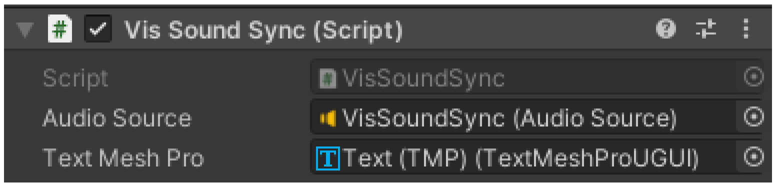
Introduction

Thanks for trying out VisSoundSync! I built this out of frustration because I couldn't find a simple, quick way to sync my subtitles with my game's audio. I tried to keep things as simple as possible, but if you have any questions please reach out at **info@skarkjets.com**, or for a speedier response, join my discord at **discord.sharkjets.com** and talk to me there!

Requirements

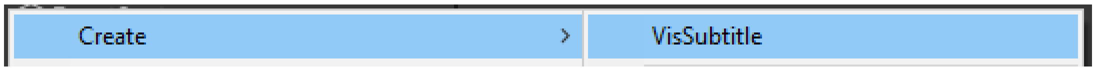
- Unity 2019+
- TextMeshPro
- An audio file and a JSON file with the timings and text. (*more on this below*)

Quick Setup
1. Install the package.
2. Add the VisSoundSync component to an object of your choosing.
3. Assign your AudioSource and TextMeshPro objects in the fields provided.

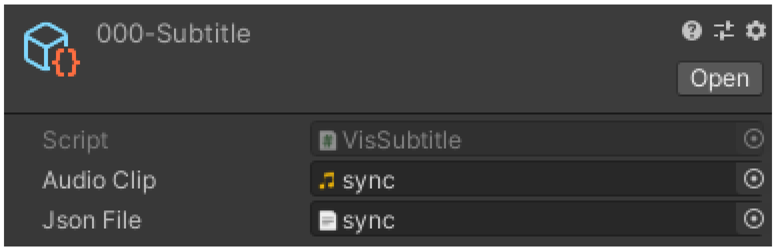

Vis Sound Sync (Script)

Script	VisSoundSync
Audio Source	VisSoundSync (Audio Source)
Text Mesh Pro	Text (TMP) (TextMeshProUGUI)

4. Right-click on your Assets folder and select Create -> VisSubtitle to create a new subtitle entry and name the file to a name of your choosing.

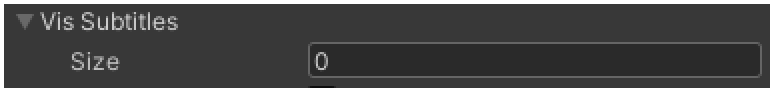


5. Assign the AudioClip (your sound file) and the corresponding subtitle JSON file in the fields provided.

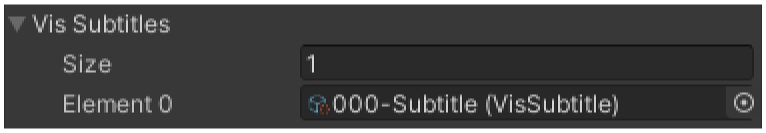

000-Subtitle

Script	VisSubtitle
Audio Clip	sync
Json File	sync

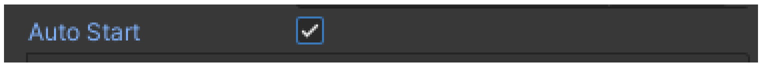
6. Back on the VisSoundSync component, expand the “Vis Subtitles” section, and change the number of entries from 0 to however many subtitles you are going to use.


Vis Subtitles
Size 0

7. Drag and Drop the new VisSubtitle files you created into the proper entry fields.

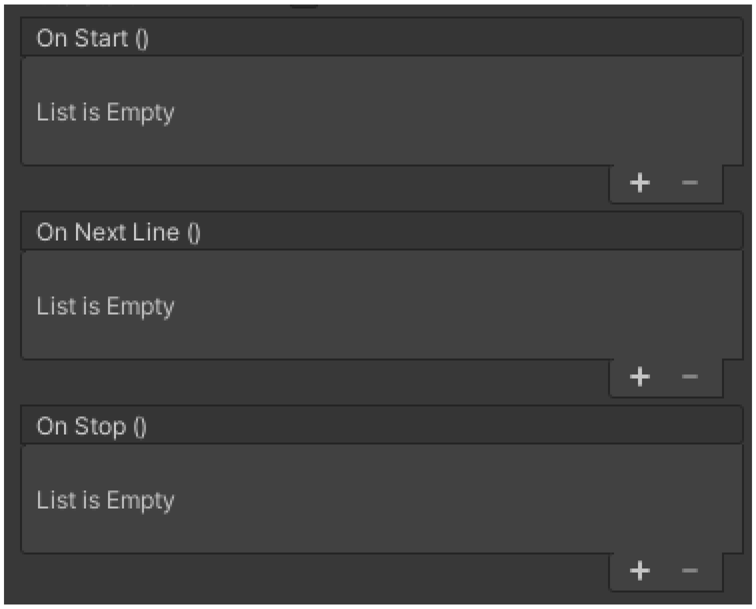

Vis Subtitles
Size 1
Element 0 000-Subtitle (VisSubtitle)

8. If you want it to start immediately when the app starts, check the Auto Start button, otherwise you’ll have to play the clip via code. (*more on this in the API section below*)


Auto Start ✓

That’s it! You’re ready to roll!
But that’s not all..

- There are 3 Unity events you can assign actions to..


On Start ()
List is Empty
On Next Line ()
List is Empty
On Stop ()
List is Empty

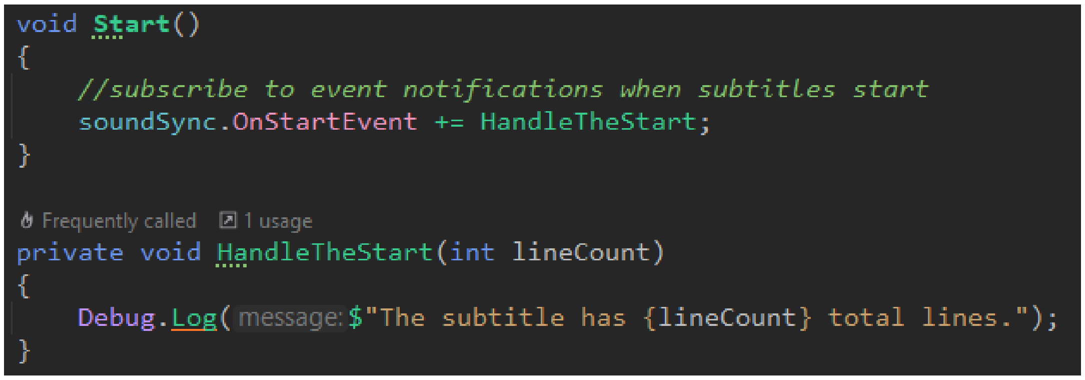
  - OnStart fires when the sound is Played.
  - OnNextLine fires when a new line is displayed.
  - OnStop fires when the last subtitle is displayed.

API - Code-based interactions

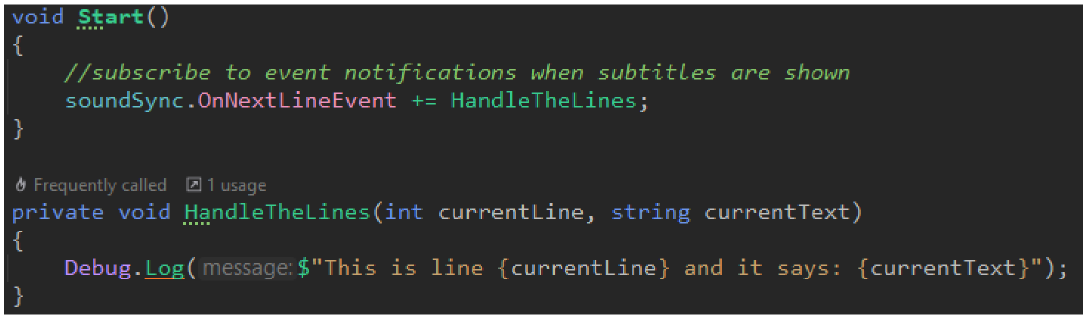
The component exposes the following interfaces for using it via code once you have a reference to it.

```
//get a reference to the VisSoundSync component
[SerializeField] private VisSoundSync soundSync;
```

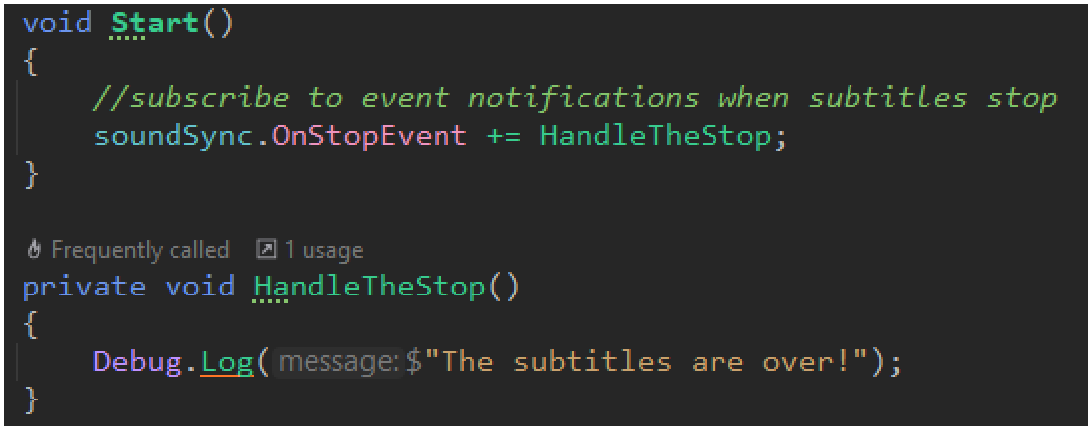
- **Methods**
    - **LoadClip(int)**
      You can assign multiple Vis Subtitles to the component, by passing its index number, you can determine which sound file and subtitles get played next.
      ```
      //plays the 2nd VisSubtitle file loaded into the component
      soundSync.LoadClip(1);
      ```
    - **LoadClip(string)**
      Don't remember the index number, but know the name of the file?
      Pass it as a string to load it by name.
      ```
      //loads the assigned VisSubtitle file by name
      soundSync.LoadClip("000-Subtitle");
      ```
    - **Play()**
      Starts playing the audio/subtitle if Auto Start is false.
      Note: You ***have*** to load a clip with one of the two previous methods before you can Play it.
      ```
      //loads the VisSubtitle then starts it
      soundSync.LoadClip("000-Subtitle");
      soundSync.Play();
      ```
    - **Pause()**
      Pauses or resumes the audio/subtitles.
    - **Stop()**
      Stops and clears the audio/subtitles.
- **Events**
    - **OnStartEvent(int lineCount)**
      An event you can subscribe to which will fire when Play is started.
      It passes the total number of lines in the subtitle file.


```
void Start()
{
    //subscribe to event notifications when subtitles start
    soundSync.OnStartEvent += HandleTheStart;
}

Frequently called   1 usage
private void HandleTheStart(int lineCount)
{
    Debug.Log( message: $"The subtitle has {lineCount} total lines.");
}
```

- **OnNextLineEvent(int currentLine, string currentText)**
  An event you can subscribe to which will fire each time a line is shown.
  It passed the current line number and text.


```
void Start()
{
    //subscribe to event notifications when subtitles are shown
    soundSync.OnNextLineEvent += HandleTheLines;
}

Frequently called   1 usage
private void HandleTheLines(int currentLine, string currentText)
{
    Debug.Log( message: $"This is line {currentLine} and it says: {currentText}");
}
```

- **OnStopEvent()**
  An event you can subscribe to which fires when the subtitles stop.


```
void Start()
{
    //subscribe to event notifications when subtitles stop
    soundSync.OnStopEvent += HandleTheStop;
}

Frequently called   1 usage
private void HandleTheStop()
{
    Debug.Log( message: $"The subtitles are over!");
}
```

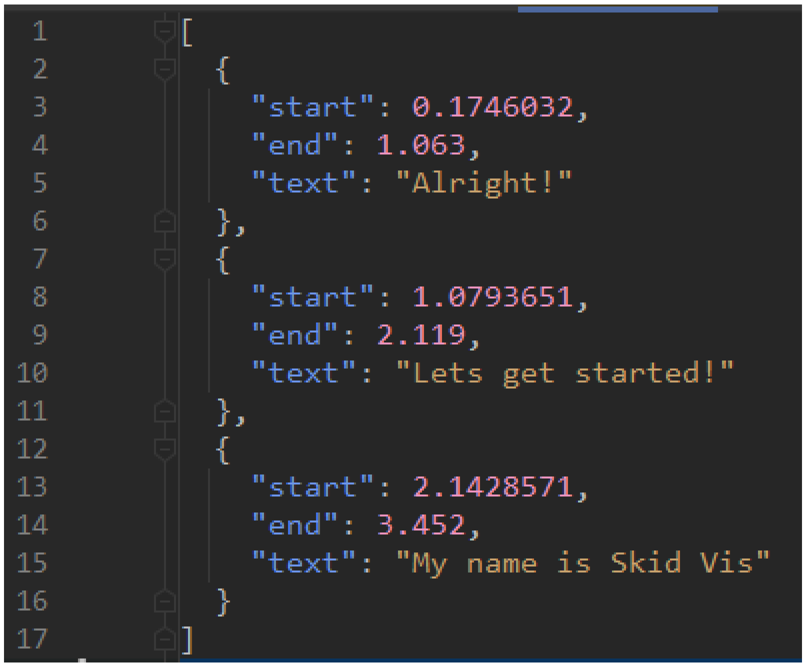
The JSON file structure

The JSON file must be in a specific format in order to be parsed. It is a very basic file structure which consists of a JSON array filled with objects which contain 3 properties:

- "**start**" is a decimal representation of when the subtitle should appear.

- “**end**” is a decimal representation of when the subtitle should disappear.
- “**text**” is a string which contains the text to show as subtitles.

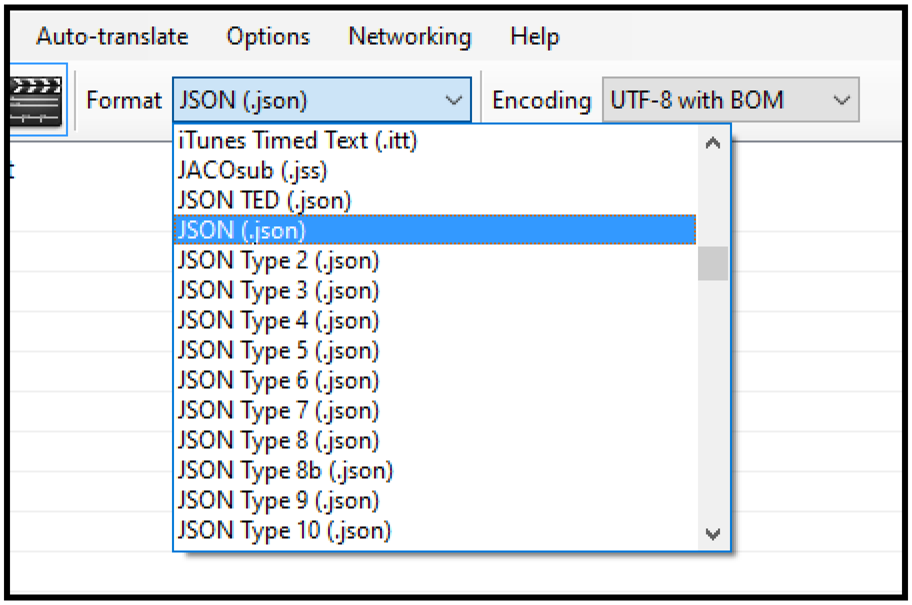
A minimal example looks like this:



```
[
  {
    "start": 0.1746032,
    "end": 1.063,
    "text": "Alright!"
  },
  {
    "start": 1.0793651,
    "end": 2.119,
    "text": "Lets get started!"
  },
  {
    "start": 2.1428571,
    "end": 3.452,
    "text": "My name is Skid Vis"
  }
]
```

This file can be created by hand or with a free program called “Subtitle Edit” available at https://nikse.dk/SubtitleEdit/

When using **Subtitle Edit**, be sure to save the file as JSON:



Format: JSON (.json)
Encoding: UTF-8 with BOM

Youtube channel: youtube.com/sharkjets