Custom Audio for Life v7

MacGruber's Tutorial Series

This tutorial gives you an introduction on how to add custom breathing audio and matching animation data to the Life#Breathing plugin.

Overview

Introduction

Adding audio to the plugin is a bit complex and most importantly, very time-consuming. This is because we want our animations nicely synchronized to the audio, which requires some data that can't be gathered automatically. The good thing is that after you get the first few breaths audios added, you get into a routine and it's getting faster.

There are some things you need:

- You need audio files, obviously. Some random audio from you favourite porn vid most likely won't do. The section "Where to get good audio?" details on what is needed.
- You need some reasonable audio editing software, I'm using Audacity. Not perfect, but good enough if you don't have access to professional studio software.
- You should be familiar with how to export files from Unity as AssetBundle. I recommend to have a look at my Unity AssetBundle Guide, if you haven't done so already.
- You will need some kind of text editor to edit JSON files, Notepad++ will do just fine.
- Being familiar with the JSON data format certainly helps.
- Obviously, you will also need VaM and the Life plugin, this tutorial uses Life 7.

Where to get good audio?

Getting proper audio for this purpose is not easy. When in VR, audio in VaM is played as spatial audio, meaning breathing actually seams to come from the characters mouth/nose. Even the slightest bit of noise would also appear to come from that spatial location as well, ruining the immersion instantly. Noise in this context means anything from white noise from the recording, but also clicks, slaps, bed squeaks, etc. This kind of thing is especially an issue for regular breathing as it is VERY low volume and common noise filter software thinks the breathing is noise.

Repetition of the same audio sample is noticeable very quickly for the player. This is especially true for moaning sounds, as you easily have a recognizable pattern. I recommend to have at least 5, better 8-10 variations for each intensity level.

These requirements do pretty much rule out just ripping audio from your favourite random porn vid. Not to mention obvious legal issues, if you want to share the audio.

This most likely means you will need someone to record your audio at a proper sound studio. Expect a lot of trial and error before getting the breathing recordings right, so make sure your actor is prepared for that. If you have to pay for studio and actor, this won't come cheap.

Thankfully there are some specialized actors out there you can hire for this kind of thing. Most of the recordings for Life were made by VoiceLikeCandy, who has her own quality sound studio.

Specialized actors for game audio might offer to cut the audio for you. Don't let them do that! Not only will that likely be cheaper, you will need data on how far breaths are apart. This is best gathered from just a recording of continuous breathing.

If you plan to share your custom audio and happen to hire an actor who doesn't do this on a normal basis, maybe just some friend, maybe a stripper girl or something like that, please consider this: Make sure the actor is aware that his/her breathing/moaning will float the internet for all eternity. Moaning and breathing can easily put of out context, far more easily than some voice lines. You have no control over what other people might be doing, and the internet is full of weird people. Make sure your actor understands what that means and is fine with it before you start.

How does Breathing work?

The Breathing sub-plugin of Life is doing more than just playing audio files in random order. In order to play synchronized animation for chest, stomach, mouth, nose and lips the plugin needs a lot of timing and some other data for each individual audio file. This needs to be provided in form a JSON database that you have to include in the AssetBundle together with your audio. Creating this database is a quite time-consuming part of adding custom audio.

The database lets the plugin know what files are actually in the AssetBundle. Inside the database, audio files are organized as "datasets", that's the dropdown you see in the plugin UI. Each dataset then contains "entries" which are the actual audio files and are sorted by "intensity".

The Breathing plugin chooses a random breath from the selected dataset within the limits of the set intensity and variance and which breaths were chosen previously to avoid repetition. Since animations spread across breaths, the plugin already chooses the next future breath the moment the current breath just starts playing. This is why changing settings in the plugin can take a while to affect anything.

Datasets and Entries

This is how a breathing database looks like:

```
{
    "info" : {
        "name"     : "Breathing",
        "author"   : "MacGruber",
        "date"     : "2019-03-14"
    },
    "datasets" : {
        "Candy Nose" : {
            "path" : "Candy-Nose/",
            "format" : "inout",
            "desc" : "Some description, just write anything here.",
            "entries" : [

                ...some audio entries here...

            ]
        },

        ...more datasets here...
    }
}
```

Let's go through the fields:

- The "info" section is just fluff. You can put anything in there, the plugin does not read it. Still, I recommend to put your creator name there and a date.
- "datasets" contains the list of, well, datasets. "Candy Nose" is the set name as it will be displayed in the plugin UI.
- "path" is the folder path inside the AssetBundle. By default audio files and the database itself have to be in a folder named "Breathing", this is hardwired in the plugin. So, in this case the actual path where the plugin looks for audio files would be "Breathing/Candy-Nose/"
- "format" can be either "inout" or "outin". With "inout" you tell the plugin that your audio files are each a breath-in followed by a breath-out. As you probably guess, "outin" means the breath-out comes first. The difference here is essentially where you want the plugin to insert some random wait time, as it can't do that within an audio file, only between them. You format choice should be very natural: For slow / normal breathing you will likely want "inout", for faster/moaning you will want "outin".
- "desc" is again a fluff field, you can put anything here. I usually put a description of the markers (explained further down) here to make the JSON easier to read.

Inside the “entries” list you then have an entry like this for every single audio file:

```
{   "file":"nd11.wav", "tags":"noseIn,noseOut", "depth":1.00,
    "markers":[["bi",0.023],["hi",2.047],["bo",3.304],["ho",6.825],
               ["be",7.322],["nx",8.877]]},
```

As before, let’s go through the fields:

- “file” is obviously the audio filename. This entry is referring to “Breathing/Candy-Nose/nd11.wav”
- “tags” can give the plugin some context to the audio. Currently supported are “noseIn” and “noseOut”. These indicate that breathing in/out happens through the nose rather than through the mouth. This causes the plugin to use different animation morphs and blend them differently. Unknown tags are ignored, so you can add additional tags if it helps you organize things. E.g. there are some entries tagged with “moan”, although it doesn’t change anything.
- Optional: “depth” indicates the depth of the breath. Its essentially a scale applied to chest, stomach, mouth and nose morphs. If depth is not set, it will be calculated based on intensity, see Intensity section. Have a look at the “Breathing Debug Tool” section in this article for how to tweak these values quickly without having to rebuild the AssetBundle all the time.
- Optional: In addition to “depth”, morphs groups can also be scaled individually if needed using “mouth”, “nose” and “lung” parameters. Lung means the combination of chest and stomach morphs plus chest rotation.
- Optional: “vcs” stands for VelocityClampScale, a parameter used by the DriverThrust sub-plugin. You can adjust how quickly the plugin adapts to velocity changes. Most breaths won’t need this, as you would just use the slider in the plugin UI. However, it becomes useful for orgasm sounds where the breathing becomes very irregular, causing wild velocity shifts of the thrust. Values smaller 1.0 dampen the velocity change a bit, making thrust more stable.
- “markers” is a list of events with in the audio file. Each of those are a pair of a short name and a time index in seconds. The order of these markers is essential for the plugin to function correctly. This does not only mean the order in the list, but also the time index! The plugin does some sanity checks to help you against typos and common issues, but it does not catch everything. If weird things happen, check your markers. The next section details the available timing markers.

Timing Markers

These markers tell the plugin where a breath starts and ends. This information is used by the plugin to calculate even more markers which then allow the event system to figure out how to drive the breathing animations. Defining these markers is not an exact science in some cases. Sometimes you have to listen to a breath several times to figure out where you should place the marker.

- “bi” indicates where “breathing in” starts. I usually have about 0.02 seconds of silence at the beginning of the file.
- “hi” indicates where “breathing in” stops and the characters holds breath
- “bo” indicates where “breathing out” starts
- “ho” indicates where “breathing out” stops and the characters holds breath
- “be” is essentially the length of the audio file. I recommend to have 500-1000ms buffer after “hi” or “ho”, it is sometimes hard to guess where a breath actually ends. Although the body movement already ended, you can still hear the breath sometimes.
- “nx” indicates where the next breath should start. Note that this is just a hint for the plugin, there is some randomness and also smoothing happening. It’s best to gather this from a continuous breathing recording, so you can just measure how far breaths are apart.

When using “inout” format, you need a sequence of bi → hi → bo → ho → be → nx.

Similarly, for “outin” you need a bo → ho → bi → hi → be → nx sequence.

Intensity

After having added all the entries, these need to be sorted by “intensity”. The intensity slider in the plugin UI just maps directly to the order of entries in the database. In the cases I had so far it worked best to just sort in descending order by the length or each breath. That means slow breaths come first, faster breaths last. You could for example just use the “hi” or “ho” marker, but it’s not an exact science.

After sorting you ideally will have a nice intensity distribution, each breath a tiny bit more intense than the last. In reality you might discover that you have gaps in your coverage. I had for example large intensity jumps or about a second length difference. You can try to doctor this a bit by adding/removing some silence to some other nearby breath. If you happen to add or cut exactly 100 or 200ms for example, you can just copy&paste the entry and shift all following markers accordingly.

Keep in mind that repetition is very noticeable. Avoid to use the directly adjacent entries to fill a gap. You can also try to recombine breath-in’s with different breath-out’s to add more variation if needed.

Multi-Breath Entries

Some more complex breaths (like an orgasm) do consist of multiple breaths that need to be played in sequence. For this you can just have that entire sequence in a single audio file, which then can be played as one.

You can just enter more markers and more scale values:

```
{   "file":"os01.wav", "tags":"",
    "lung":[0.5,0.5,0.4,0.6,0.8,0.8,0.6,0.6],
    "mouth":[0.6,0.6,0.6,0.7,0.9,0.9,0.8,0.6],
    "vcs":0.7,
    "markers":[["bo",0.020],["ho",0.285],["bi",0.297],["hi",0.408],
               ["bo",0.413],["ho",0.675],["bi",0.708],["hi",0.880],
               ["bo",1.056],["ho",1.405],["bi",1.425],["hi",1.529],
               ["bo",1.550],["ho",1.962],["bi",1.987],["hi",2.321],
               ["bo",2.352],["ho",2.870],["bi",2.890],["hi",3.296],
               ["bo",3.485],["ho",4.095],["bi",4.115],["hi",4.393],
               ["bo",4.412],["ho",5.041],["bi",5.080],["hi",5.861],
               ["bo",5.923],["ho",7.208],["bi",7.346],["hi",8.571],
               ["be",9.128],["nx",10.128]]},
```

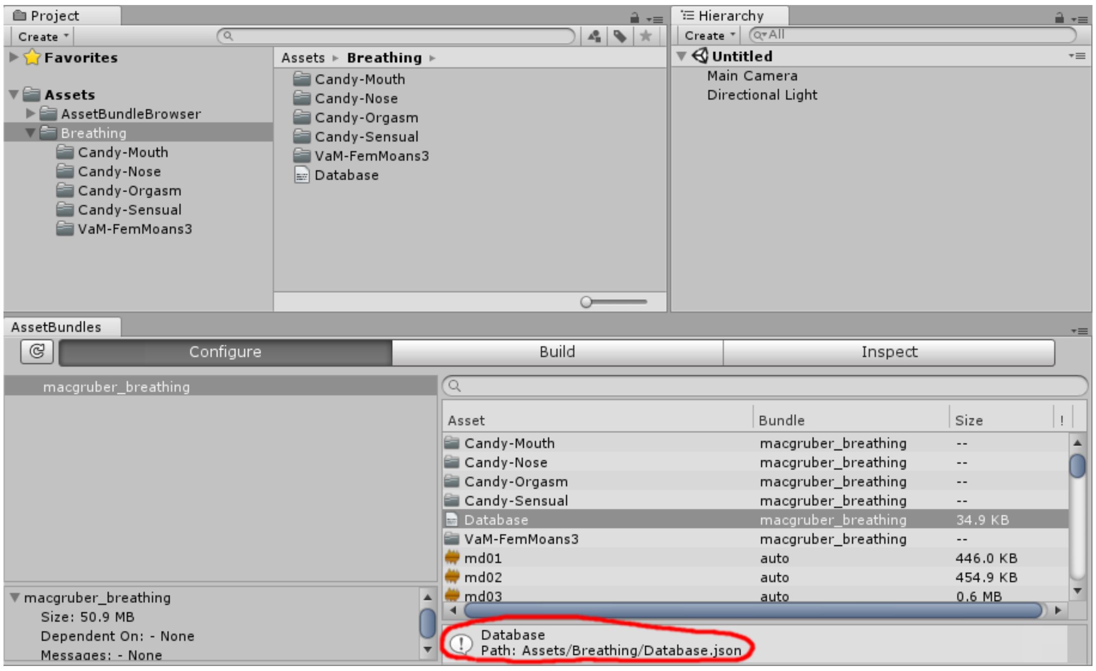
For each breath in the file you have a bo → ho → bi → hi sequence and only a single be → nx at the end of the file. For “inout” format this would be bi → hi → bo → ho, of course. Some additional “be” markers will be inserted between the breaths automatically.

“depth”, “mouth”, “nose”, “lung” and “vcs” parameters all can be either a single value or an array with a value for every breath in the entry.

Unity Export

While VaM does support loading individual audio files at runtime, loading them from an AssetBundle is quite a lot faster. The Life plugin loads ".audiobundle" files, which are just renamed AssetBundles. Export from Unity is fairly straight forward once you have your Unity project setup and understood how exporting works in general. I recommend to have a look at my Unity AssetBundle Guide if you are unfamiliar with this.

All files need to be within a folder named "Breathing". The path to the database needs to be "Assets/Breathing/Database.json", otherwise the plugin won't find it. I recommend to have a folder per dataset for the audio files, although this is not required. You can simply drag&drop the database and the folders into the AssetBundleBrowser. Individual files within the folders will be added automatically.

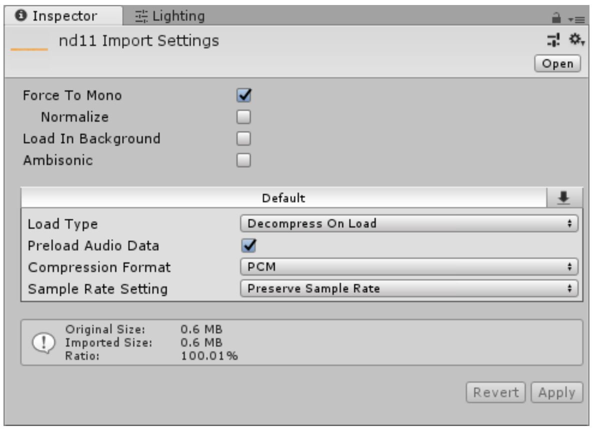


Asset	Bundle	Size
Candy-Mouth	macgruber_breathing	--
Candy-Nose	macgruber_breathing	--
Candy-Orgasm	macgruber_breathing	--
Candy-Sensual	macgruber_breathing	--
Database	macgruber_breathing	34.9 KB
VaM-FemMoans3	macgruber_breathing	--
md01	auto	446.0 KB
md02	auto	454.9 KB
md03	auto	0.6 MB

macgruber_breathing
Size: 50.9 MB
Dependent On: - None
Messages: - None

Database
Path: Assets/Breathing/Database.json

Import settings for your audio files should look like the following. Note that you can select multiple files to change the import settings all at once. PCM compression format is best for very short sound effects like breathing, for longer sounds, longer voice lines or music you might want to choose Vorbis/MP3 instead to reduce bundle size.



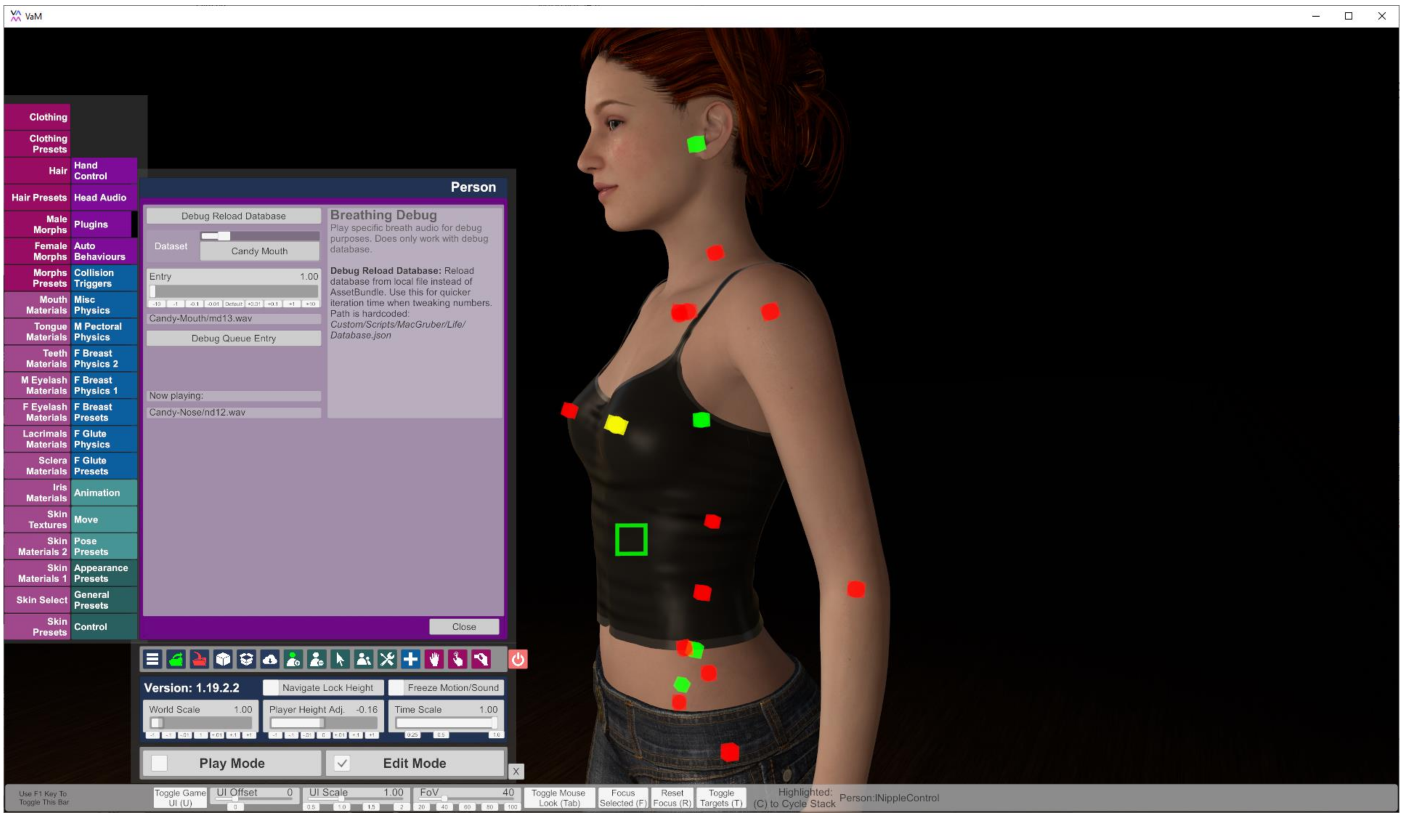
nd11 Import Settings

Force To Mono	✓
Normalize	☐
Load In Background	☐
Ambisonic	☐
Load Type	Decompress On Load
Preload Audio Data	✓
Compression Format	PCM
Sample Rate Setting	Preserve Sample Rate

Original Size: 0.6 MB
Imported Size: 0.6 MB
Ratio: 100.01%

Once you build your AssetBundle, copy it somewhere into your VaM folder and make sure the file extension is ".audiobundle".

Breathing Debug Tool



Debug Reload Database

Dataset: Candy Mouth

Entry 1.00

Candy-Mouth/md13.wav

Debug Queue Entry

Now playing:

Candy-Nose/nd12.wav

Breathing Debug

Play specific breath audio for debug purposes. Does only work with debug database.

Debug Reload Database: Reload database from local file instead of AssetBundle. Use this for quicker iteration time when tweaking numbers. Path is hardcoded: *Custom/Scripts/MacGruber/Life/Database.json*

Version: 1.19.2.2

World Scale 1.00

Player Height Adj. -0.16

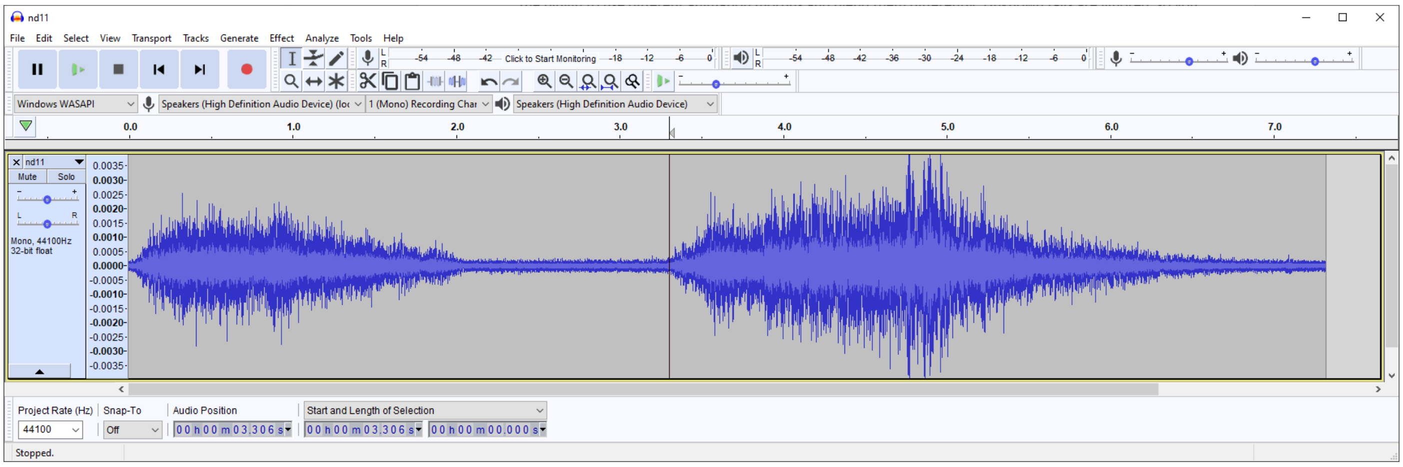
Time Scale 1.00

Highlighted: Person:lNippleControl

The Breathing contains a debug tool which allows you to queue a specific audio entry. It can also hot-reload the database. This is especially helpful when tweaking animation scale or timing values, making sure the animation matches audio perfectly.

1. Add *MacGruber_BreathingDebug.cslist* (or *MacGruber_BreathingThrustDebug.cslist*) onto your character. If your character happens to already have a breathing plugin, please remove that first.
2. Go to the Breathing plugin UI. Load your AudioBundle and make other settings as needed.
3. Go to the BreathingDebug plugin UI.
4. Copy your database JSON file to “Custom/Scripts/MacGruber/Life/Database.json”
5. Press “Debug Reload Database”. This will load the JSON database from above path instead of using the database from the AudioBundle. Whenever you made a change to the JSON file, just hit the button again to reload at runtime. However, note that the actual audio files are still from the AudioBundle.
6. Select a Dataset and an Entry.
7. Press “Debug Queue Entry” to queue the selected breath to be played after the current one. Afterwards the plugin will just go back to normal operation, unless you queue another breath again.
8. You can see the currently playing file at the bottom as “Now playing”.

Tips for handling Audio


nd11
File Edit Select View Transport Tracks Generate Effect Analyze Tools Help
Click to Start Monitoring
Windows WASAPI
Speakers (High Definition Audio Device) (loc
1 (Mono) Recording Char
Speakers (High Definition Audio Device)
0.0 1.0 2.0 3.0 4.0 5.0 6.0 7.0
Mute Solo
Mono, 44100Hz
32-bit float
Stopped.

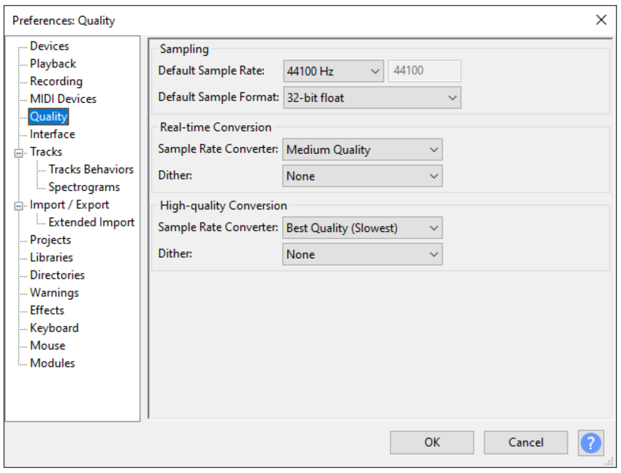
Project Rate (Hz): 44100
Snap-To: Off
Audio Position: 00 h 00 m 03.306 s
Start and Length of Selection: 00 h 00 m 03.306 s, 00 h 00 m 00.000 s

General notes:

- VaM is doing spatial audio for you, this works on mono data. If you happen to have stereo recordings, make sure to downmix them to mono to remove all spatial information.
- As you likely have some base noise from your microphone, I recommend to do a bit of fade in at the beginning of the breath and a fade out at the end to make it less noticeable.

Audacity notes:

- Audacity can zoom on the vertical axis if you hold CTRL and use the mouse wheel while hovering over the vertical axis labels. With SHIFT and mouse wheel you can move up and down to center it.
- 1/100 second precision is enough for timing values. However, I just use the 1/1000 precision from Audacity as it is faster to just enter the value rather than thinking about rounding numbers.
- You might want to disable Dither in Audacity. This is apparently helpful when converting between sample rates, but for some reason it introduces noise when just cutting WAV files at the same sample rate. That "noise" is not really hearable, but it can make it hard to see audio data with volume this low. (Note: Most of the Life audio was already cut before I discovered this setting, though.)



Sampling
Default Sample Rate: 44100 Hz 44100
Default Sample Format: 32-bit float

Real-time Conversion
Sample Rate Converter: Medium Quality
Dither: None

High-quality Conversion
Sample Rate Converter: Best Quality (Slowest)
Dither: None