Creative Coding for Sound Assignment 2
Author: B200247

Sleep Catcher

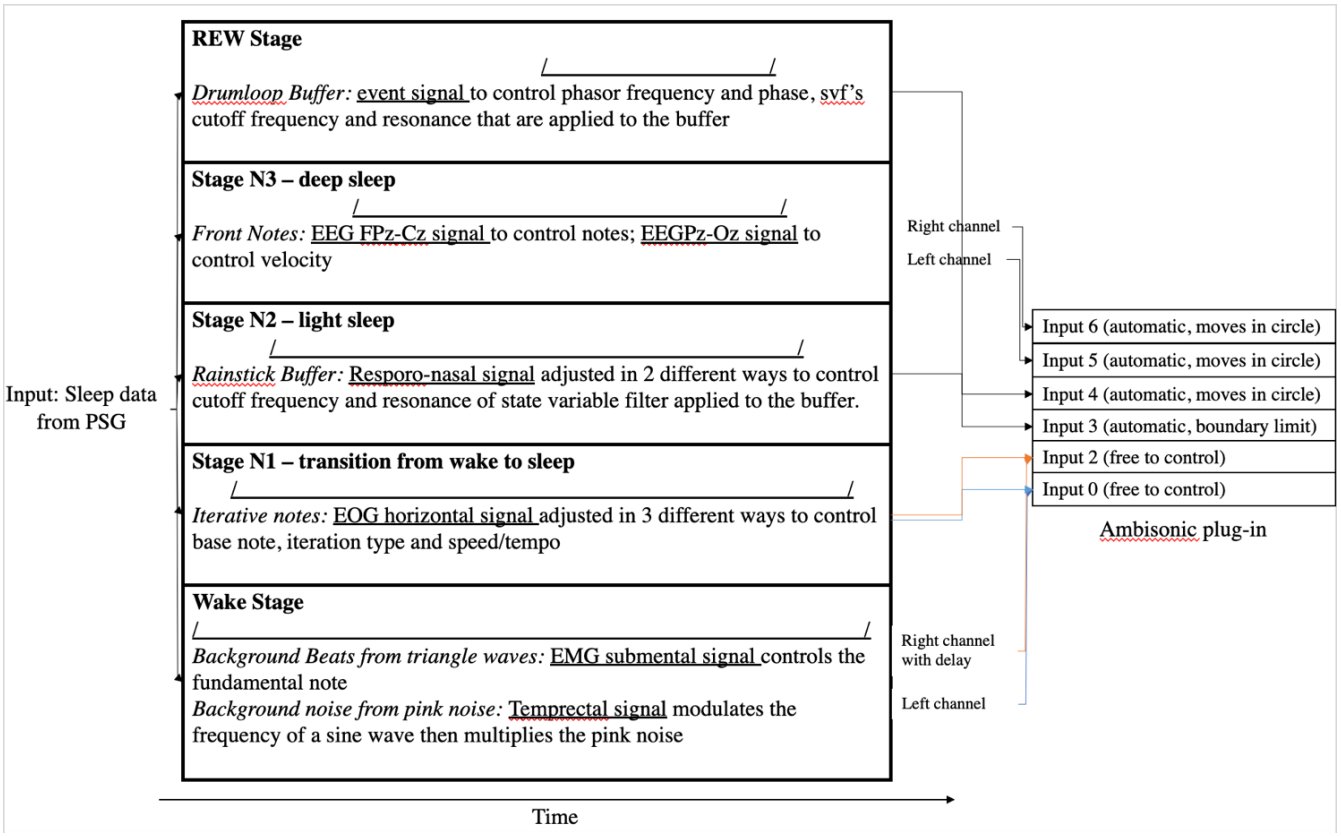
This work makes sonification from polysmnographic sleep recording. It explores:

1) Reconstruction of sleep: usually a sleeping circle contains several sleep stages in time order, e.g. N1 sleep (transition from wake to sleep), N2 sleep (light sleep), N3 sleep (deep sleep), (here may have N2 stage again or even wake up), REW (rapid eye movement sleep), etc.. Different stages have different features, for example in stage N2 sleep spindles and K-complexes can be observed. However, in real-world these stages hardly overlap, and it takes a really long time to go through a full sleep recording. So I wonder if I can 'accelerate' the experience of sleeping in some ways, and reconstruct sleep stages vertically (by layers of sound) instead of horizontally (by time). In this way, people are able to experience sleep in both a new angle and a shorter time, while different sleep stages are still distinctive.

2) 3D experience: this could be easier to do by sound than by visual, since multichannel speakers or even just stereo headphones could be more common in daily life than 3D glasses. I'd like to allow users to adjust the source position by themselves, in other words, adjust listener's position by themselves, so as to have fresh experiences. By adding some subtle differences such as delay in advance, it originally makes a feeling of space – just as if you are standing somewhere in someone's dream, and some features are controllable while some are not. This also adds chaos – like what we experience.

For full experience, please use headphone
Also please examine the audio recording instead of video sound – screengrab was not able to preserve the 3D effect

The structure is shown below:



Data:

Data is downloaded from https://physionet.org/content/sleep-edfx/1.0.0/, this project uses the 1st recording of subject 2, named after SC4021E0-PSG.edf and SC4021EH-Hypnogram.edf as an example.

Preprocess:

Coding with help from ChatGPT was involved to 1) combine 2 files together to map the PSG data to each stage; 2) transfer edf file to csv file; 3) remove too many wake stage data in the beginning then downsample; 4) normalize data. This is included in the appendix.
Before loading into max, data are further separated into different txt files according to different stages.

	A	B	C	D	E	F	G	H	I
1	EEG Fpz-C	EEG Pz-O:	EOG horiz	Resp oro-r	EMG subn	Temp rect	Event marl	annotation	
2	0.382666	0.516854	0.545926	0.497043	0.941981	0.878474	0.502762	Sleep stage W	
3	0.383768	0.446507	0.560057	0.491692	0.938697	0.878474	0.552486	Sleep stage W	
4	0.39809	0.487543	0.578898	0.488032	0.925561	0.879718	0.524862	Sleep stage W	
5	0.443996	0.530532	0.491286	0.488876	0.936508	0.87557	0.519337	Sleep stage W	
6	0.385971	0.461163	0.583608	0.489158	0.920635	0.879303	0.41989	Sleep stage W	
7	0.362468	0.44553	0.550636	0.490848	0.935413	0.882621	0.61326	Sleep stage W	
8	0.356225	0.479726	0.541215	0.497606	0.920088	0.8764	0.447514	Sleep stage W	
9	0.323173	0.476795	0.734338	0.481836	0.91133	0.878059	0.480663	Sleep stage W	

Read Data:

There is a particular patch to read data, which involves 5 Colls:
Wake stage Coll sends EMG submental signal and Temprectal signal, read every 2000 ms – so that it serves well as background sound with slow evolution and without catching too much attention in the main part of the sound.
Stage 1 Coll sends EOG horizontal signal, I also wanted it to be something in the background, changes but does not change too much. From previous observation, EMG submental signal suits this best. It also reads every 2000 ms.
Stage 2 Coll sends Resporo-nasal signal, read every 1000 ms.
Stage 3 as the most talked about part, deserves to have the most changeable EEG FPz-Cz signal and EEGPz-Oz signal, read every 125 ms.
Stage REW uses the left event signal – as it is where dreams happen, 'events' makes sense too! It is read every 500 ms.

Sonification Design:

Wake stage (which is separated into 2 parts:)

Background beat: beats are used to represent some gradual drowsy feeling, and it is made to be polyphony. The input data, after some scaling and selection, controls the fundamental note;

Background noise: pink noise was initially used as it is relaxing and friendly to sleep. Since I'd also like to add weirdness and uncertainty – sometimes you just can't control what you are thinking of when about to sleep, similar to when distracted. In this sense, it's reasonable to modulate it, and what I did is multiply a sine wave whose frequency is controlled by data, resulting in it sometimes sounds like sand, sometimes tide, sometimes electric current…

Stage 1:
Expected to be more noticeable than the wake stage (the focus is sleep anyway!) but not too noticeable compared with later stages. It uses iteration – with the signal (input data) adjusted in 3 different ways to control base note, iteration type and speed/tempo. In this way the sound loops, like hypnosis technique, but also changes, as anything changes in the real world.

Stage 2:
It's true sleep but not yet deep, I expect some obscure feeling instead of clearer notes, hence rainstick buffer is used – natural white noise works well here. Data is adjusted in 2 different ways to control cutoff frequency and resonance of state variable filter applied to the buffer, and more thoughts goes to the ambisonic part.

Stage 3:
It's deep sleep and I'd like to put the same importance on sound as in this stage is in actual sleep. We are not yet able to know what is going on in our minds now, so the sound is purely a representation of physical data – EEG signal. EEG FPz-Cz signal is used to control notes, EEGPz-Oz signal to control velocity. And as the signal is, changes rapidly, the notes change rapidly too. It uses SurgeXT vst and its Good Childhood preset.

REW stage:
Just intuitively feel drumloop would work here! Isn't exciting that people dream, and it corresponds to the physical rapid eye movement too. Data to control phasor frequency and phase, svf's cutoff frequency and resonance that are applied to the buffer.

Ambisonic design:

The ambisonic effect and presets take reference from
https://www.youtube.com/watch?v=57FIZ9zr_18
And used vst and decoder preset from
https://www.matthiaskronlachner.com/?p=2015
Max's ICST Ambisonics package is also needed

Input 0&2:

It has the background beat and noise from wake stage and iterative notes from stage 1. Both beat and noise have the right channel delayed in advance, so as to create a better feeling of space, and this allows noticeable but not too abrupt changes when users move input 0 and 2 in the monitor. Iterative notes are linked directly to the monitor, indicating more straightforward about the source position or listener's relative position.

Input 3:

It's rainstick from stage 2, I'd like it to move and change uncontrollably like how nature sounds are, hence there is preset movement.

Input 4:

There is also preset movement, just circling around the border, but users are allowed to stop it's movement and adjust by themselves too – as if you wanted to disturb or influence the dream in some ways.

Input 5 and 6:

Similarly, when not monitored, its left and right channel takes 2 inputs and moves around by themselves, designed to be in opposite directions. Users are also allowed to stop and move them by themselves to explore – different behaviour when overserved or not may lead to deeper philosophical questions.

Reference:

Data:
https://physionet.org/content/sleep-edfx/1.0.0/
About sleep:
https://www.webmd.com/sleep-disorders/pink-noise-sleep
https://www.sleepfoundation.org/stages-of-sleep
https://www.ncbi.nlm.nih.gov/books/NBK526132/
Ambisonic:
https://www.matthiaskronlachner.com/?p=2015
https://www.youtube.com/watch?v=57FIZ9zr_18
ICST Ambisonics package https://www.zhdk.ch/en/research/icst
SurgeXT: https://surge-synthesizer.github.io/
All other knowledge for sleep and guidance in coding from ChatGPT

Codes:

---------- Edf to csv ----------

```
import mne
import pandas as pd
import numpy as np
```

```
def combine_psg_and_hypnogram(psg_edf_file, hypnogram_edf_file,
csv_output_file):
    # Load the PSG EDF file
    psg_raw = mne.io.read_raw_edf(psg_edf_file, preload=True, verbose=False)

    # Get the data and channel names
    data = psg_raw.get_data().T
    channel_names = psg_raw.ch_names

    # Create a DataFrame for signal data
    signal_df = pd.DataFrame(data, columns=channel_names)

    # Load the hypnogram EDF file
    hypnogram_annotations = mne.read_annotations(hypnogram_edf_file)

    # Add an empty 'annotation' column to the signal DataFrame
    signal_df['annotation'] = np.nan

    # Map the annotations to the corresponding time points in the signal
DataFrame
    for onset, duration, description in zip(hypnogram_annotations.onset,
                                            hypnogram_annotations.duration,

hypnogram_annotations.description):
        start_sample = int(onset * psg_raw.info['sfreq'])
        end_sample = int((onset + duration) * psg_raw.info['sfreq'])
        signal_df.loc[start_sample:end_sample, 'annotation'] = description

    # Save the combined DataFrame as a CSV file
    signal_df.to_csv(csv_output_file, index=False)

psg_edf_file = 'SC4021E0-PSG.edf'
hypnogram_edf_file = 'SC4021EH-Hypnogram.edf'
csv_output_file = 'sleep.csv'

combine_psg_and_hypnogram(psg_edf_file, hypnogram_edf_file, csv_output_file)
```

---------- Downsample ----------

```
import pandas as pd

# Read the data from the CSV file
csv_file = 'sleep.csv'  # Replace this with the path to your CSV file
df = pd.read_csv(csv_file)

# Delete the first 21000 rows
```

```
df = df.iloc[21000:]

# Downsample the data by a factor of 500
df_downsampled = df.iloc[::500]

# Save the downsampled data to a new CSV file
df_downsampled.to_csv('downsampled_data.csv', index=False)
```

---------- Normalize ----------

```
import pandas as pd

# Load the downsampled data
csv_file = 'downsampled_data.csv'
df = pd.read_csv(csv_file)

# Normalize each column
df_normalized = df.apply(lambda x: (x - x.min()) / (x.max() - x.min()),
axis=0)

# Save the normalized data to a new CSV file
df_normalized.to_csv('normalized_downsampled_data.csv', index=False)
```