Virtual Music Ensemble Technologies
Internship Coding Assignment

Overview: Your mission is to build a lightweight script for what we call the OPUS Synchronizer. The job of the OPUS Synchronizer is to read in multiple recording files and output a single virtual ensemble file. For reference, here is an example of a basic virtual ensemble, featuring Phillip Sylvester, one of our Co-Founders: https://www.youtube.com/watch?v=cPXvsadJR_k.

Logistics: We recommend Python or C++ for the script, but if there is another language you believe is more adept at accomplishing the project, feel free to use that language instead. The script can be one file or multiple files.

Timing: We expect the script to take 4-6 hours to implement. If you have reached 6 hours and not yet finished the script that is perfectly alright, just include a commented framework for how you would implement the remaining functions.

Presenting: The first part of your interview will be to demo your script over Google Meet by sharing your screen. You can use any video files you would like as inputs when you demo your script. If the script is not fully finished, do be prepared to demo the functions that are finished and verbally explain and elaborate on how the remaining portions would be implemented. We will also ask for a copy of your script at the end of the interview.

The Project

A fully implemented synchronizer script will automate the process of creating a virtual ensemble.

A flow for the script might look something like this:

1. Receive as input multiple videos.
    a. The videos can be read locally off your machine.
    b. For simplicity, the videos should all be the same length (so that it does not take too long to render, we recommend keeping the videos below 30 seconds in length).
    c. The videos should include both audio and video data (e.g., mp4 files).
    d. The videos you use do not have the be people performing music, but they should all be different and include distinct audio such that it is clear in the output final virtual ensemble that each video is contributing audio to the mixed audio.
2. Tile the videos in a grid layout.
    a. For simplicity, the tiled video should be on a grid layout with each of the videos having equal dimension.
    b. For example, if there are 9 videos input, the grid could be 3x3 with each video. As another example, if there are only 5 videos input, the grid might still be 3x3 with 4 black squares holding the place of the other 4 spots in the grid, or the grid

might be 3x2 with one black square, or the grid might have 3 videos on a top row and 2 videos on a bottom row — this is your choice how to handle uneven number of videos.
    c. For simplicity, the grid layout should be constant throughout the final virtual ensemble (i.e., no need for videos changing position or the grid structure changing throughout the final virtual ensemble).
3. Mix the audios.
    a. Audio from each and every input video should be audible in the mixed audio.
    b. For bonus points, normalize the audio channels such that all videos are of roughly the same volume in the mixed audio (i.e., louder videos become softer, softer videos become louder).
4. Combine the mixed audio with the tiled video
    a. The final output virtual ensemble file should contain the mixed audio laid atop the tiled videos. The video and audio data should be synchronized with respect to each other.
    b. The final output virtual ensemble file should be in a standard video file format.

Because this is a script, no UI is needed, you can just run it through your console. Since you will be running the script while presenting via screensharing, it is likely all this processing will be a computational burden on your computer. Therefore, we recommend keeping your input videos (and subsequently, the output final virtual ensemble) to below 30 seconds in length, the output final virtual ensemble to only 480p or 720p, and the number of input videos to 9 or fewer. Although if you have come up with a more efficient algorithm that can do all of this without slowing down, we would love to see it.

Many of the operations required in this script can be done using prebuilt video processing, audio processing, and image processing libraries. We encourage you to do research on using new libraries if you are not familiar with them and encourage you to use them as much as possible, rather than hand-building all the functions.

Evaluation: The project is written to have some ambiguity, this is where we would like to see your creativity. A good project does not have to be fully implemented. Not only are we looking for creative approaches to problem solving, but we are also looking to work with engineers who are team players, not solo coders. This means your code should have a clear coding style with useful documentation and commenting. It should be possible for another engineer with only a high level understanding of the code to pick up your script and understand what is going on.

Finally, we hope your project will highlight one of the most important traits we have on our VMET team — a winning attitude.