Framework Setup

1 About this document:

This document provides an overview of the contents of our software package (provided on the GitHub[1] repo). More specifically, it explains the different software projects that you can find, but also the steps that you need to take to compile and run them yourself. Please note that all development was done in Windows 10 & 11, then these are the only platforms we currently support.
Our software package assumes that you have a working setup (i.e. Acoustophoretic boards, firmware and calibration), as most of our examples require this to run. Also, most of our current examples are specifically designed for top-bottom board setups. However, other setups can be used.

2 Folder Overview:

Our content is structured in the following sub-folders, each of them focussing on a key aspect related to our software package:

0. **Libraries (zipped)**: In order to avoid compatibility issues, we decided to build a compilation of pre-compiled binaries for all the libraries we use. This avoids pointing you towards dozens of GIT repositories to build your libraries, wasting weeks in compiling them, finding out some of your builds are not compatible, etc. This folder contains all you need to run our software straight away, and the process to set this up is explained below.
1. **OpenMPD**: This folder contains the core algorithms supporting our development. More specifically, it contains our board controller and a set of supported high-performance multi-point solvers (*GS-PAT*, IBP and naive). These components are developed in C++ and their source code, together with documentation and example applications (in C++) can be found in this subfolder.
2. **OpenMPD_Client**: In an effort to make development easier, we integrated our core algorithms into Unity 3D game engine. This sub-folder contains the Unity implementation, as well as a set of application examples about the use of the main components of the framework (main structure, primitives and tools).

These files can be found in our online repository (GitHub[2]) together with the full *OpenMPD* framework tutorial.

3 Installation procedure:

3.1 Tools required:

Running our software requires the following software tools to be installed on your home computer.

1. **Microsoft Visual Studio (Toolset v141 [vs2017] or v142 [vs2019])**: Any version of Visual Studio will do, as long as you install and use the right C++ toolset (**v141 or v142**) and install C# if you want to use our Unity integration. In any case, we used either Visual Studio 2017 and Visual Studio 2019 during development, and these are the versions recommended.
2. **OpenCL SDK:** Our solver uses OpenCL and you will need to get a suitable developer SDK, which depends on your GPU manufacturer (Nvidia, AMD, Intel). Although OpenCL implementations should work on any GPU platform, all our computers have been using NVIDIA GPUs and their CUDA Toolkit.
3. **Unity (optional)**: You will need this if you want to create applications in C#. We used version 2019.3.4f1, but it should be compatible with any version.
4. **7-zip(optional):** You need to download and unzip our **0.libraries** folder but otherwise, you are good to go!

[1] Link removed due to anonymity
[2] Link removed due to anonymity

3.2 System requirements:

- **Windows 10 or 11**: Our software was developed for these platforms. Using a different version of Windows might be possible, but will require you to change the "*Software SDKs*" in the Properties of your Visual Studio C++ project.
- **Graphics Card:** Our OpenCL solver requires workgroup sizes of, at least, 512 compute units. This is well within the capabilities of current GPUs, but you can check these running applications such as GPU Caps Viewer. Due to this requirement, it is highly recommended to use an external graphics card with up to date drivers (e.g., NVIDIA and Cuda SDK) when running these examples.

3.3 Installation steps:

Once you have downloaded and installed the tools above, this is all that you need to do. Please, make sure that Visual Studio is closed before starting.

- **Download "*0.libraries.zip*"** (from our GitHub[3]) and unzip it in a local folder in your computer. You will need to use the path to this local folder later (step 4).
- **Declare a LIBS_HOME environment variable**, pointing to your local folder:
    1. Go to "*Control Panel*"→ "*System and Security*"→ "*System*".
    2. On the left side panel, click on "*Advanced System Settings*" (requires admin/elevated rights).
    3. The *System Properties* dialogue box will open (see Figure 1, left), showing the tab "*Advanced*". Click the "*Environment Variables*" button to the bottom right of the dialogue box.
    4. The *Environment Variables* dialogue box will open (see Figure 1, right). In the bottom panel (System variables), select "*New*".
        - Enter LIBS_HOME as your "*Variable Name*".
        - Copy the path to the folder as your "*Variable Value*". The path should point inside the unzipped folder (e.g. *"<path_to_your_folder>\0.libraries*").
        - Click "*OK*"
    5. If you are using a NVIDIA GPU, please install the CUDA SDK and the NVIDIA Experience to keep your drivers updated. Once installed, you can check if your CUDA SDK installation created its environment variable correctly (variable CUDA_PATH should appear in your System variables), otherwise you will need to do it manually by repeating the step 4 (using CUDA_PATH as a variable name and the path pointing to the CUDA installation as the variable value). To avoid driver issues, please verify the latest version for the NVIDIA graphics card drivers are installed before running the *OpenMPD* framework.
    6. Make sure that in the Visual Studio installation, the module "desktop development with c++" is included to avoid some possible compilation issues. If don't, just open the Visual Studio Installer and click on modify, then add this module by ticking the box on it and pressing the modify button on the bottom-left of the window.

[3] Link removed due to anonymity

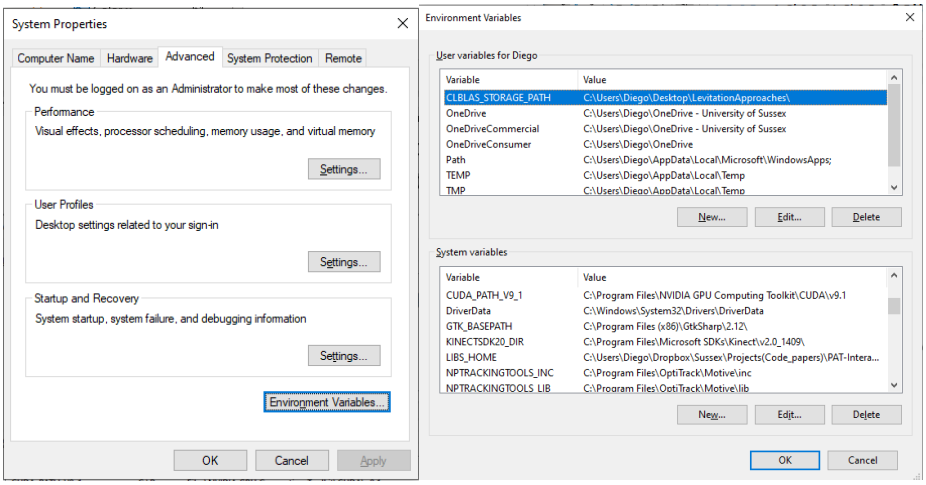

Figure 1: Creating your LIBS_HOME environment variable

- **Compiling C++ project:** If the steps above were successful, you should be able to open our C++ project compile them and run them. Please refer to "The *OpenMPD* Tutorial Guide" for instruction about the framework itself, the main components and examples.