**MSc BCI-VR Prototype**

**Overview & Setup Instructions**

**v2.0**

**Equipment**

We used an g.tec EEG cap, electrodes, Open BCI, Openvibe, OSVR headset and the Unreal Game Engine.

**Repository**

https://github.com/michaelmcmahon/MSc\_BCI-VR\_Prototype

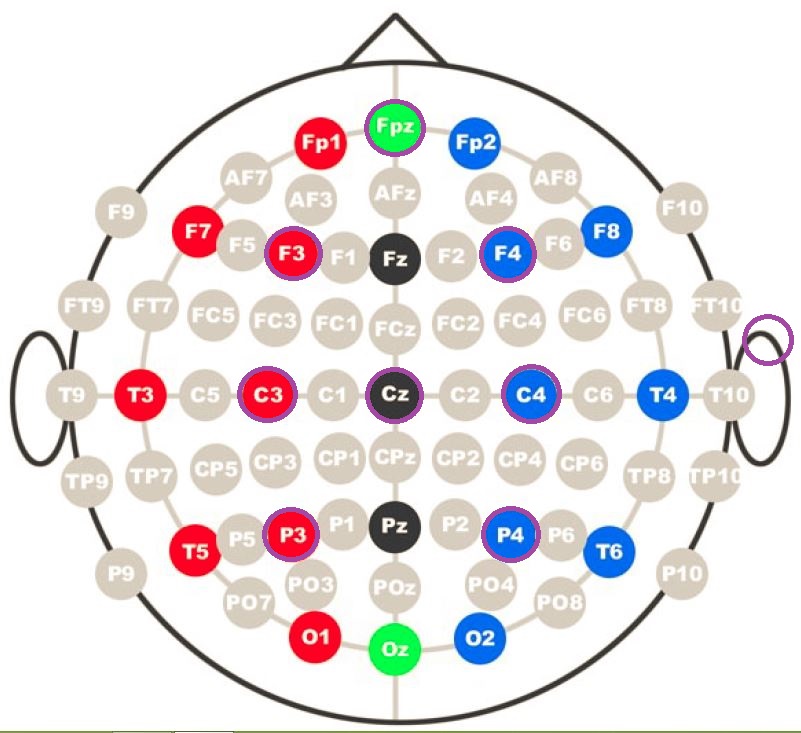
**Instructions**

1. Install the OpenVibe software from: <http://openvibe.inria.fr/>
2. Download the latest BCI-VR folder from Git and save to a folder on your system <https://github.com/HackTheBrain/MindAnamorphosis/tree/master/BCI>
3. Setup your OpenBCI board as per the instructions in the OpenBCI tutorial: <http://docs.openbci.com/Tutorials/01-Cyton_Getting%20Started_Guide>
4. To connect OpenBCI to OpenVibe follow the instructions here: <http://docs.openbci.com/3rd%20Party%20Software/03-OpenViBE>
5. Make sure you read the additional Driver setup information here: <http://openvibe.inria.fr/drivers-openbci/>
6. Setup the electrodes on your headset.

Electrodes were positioned at: FC3, C3, CP3, Cz, FC4, C4, CP4

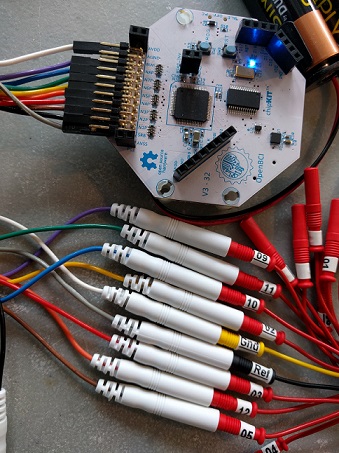
The Ground Electrode was positioned at: Fpz

The Reference and Ground Electrodes were positioned on the Right and Left Ear Lobes



1. We connected the Electrodes to the following inputs on out OpenBCI board.

|  |  |  |  |
| --- | --- | --- | --- |
| Electrode | g.tec Ch Number | Colour | BCI Input |
| **FC3** | **9** | **Purple** | **8** |
| **FC4** | **10** | **Blue** | **7** |
| **C3** | **11** | **Green** | **6** |
| **Cz** | **2** | **Yellow** | **5** |
| **C4** | **12** | **Orange** | **4** |
| **CP3** | **3** | **Red** | **3** |
| **CP4** | **5** | **Brown** | **2** |
| Ref |  | Grey | SRB |
| Gnd |  | White | AGND6 |



1. Follow the instructions here to setup the Channel Acquisition for your specific headset electrodes <http://openvibe.inria.fr/channel-selection-in-acquisition-server/>
2. This set of scenarios implements the Graz BCI, based on motor imagery of the hands. It computes the spatial filters that efficiently discriminate the signal using CSP, for significantly better performances. The scenario is based on the example scenario provided with OpenVibe which you will find it in [Your Location]:/openvibe/share/openvibe/scenarios/bci-examples/motor-imagery-CSP

**mmcm-mi-csp-0-signal-monitoring-v1.xml:** This scenario should be always used prior to anything and in background to check the signal quality of the acquisition device. Once you are sure that the EEG acquisition runs correctly, you can go on to the next step.

**mmcm-mi-csp-1-acquisition-v1.xml:** First step is to acquire some data in order to train the classifier that will discriminate Right and Left-hand movements. The training session can be configured in the LUA stimulator (number of trials, timings, etc.). This will collect the EEG signals and write to a file in [Your Location]:/MSc\_BCI-VR\_Prototype/signals/motor-imagery-csp-1-acquisition-{date}-{time}.ov.

**mmcm-mi-csp-2-train-csp-v1.xml: This** scenario computes a Common Spatial Pattern to produce a spatial filter csp-spatial-filter.cfg that maximizes the difference between the signal of the two classes. NOTE: Remember to change the “Generic Stream Reader” box to point to the location of file in the signals folder that you recorded in the previous step using mmcm-mi-csp-1-acquisition-v1.xml

**mmcm- mi-csp-3-classifier-trainer-v1.xml:** This scenario trains an LDA classifier motor-imagery-bci-config-classifier.cfg using the csp-spatial-filter.cfg created in the previous scenario. NOTE: Remember to again change the “Generic Stream Reader” box to point to the location of file in the signals folder that you recorded in the previous step using mmcm-mi-csp-1-acquisition-v1.xml

**mmcm-mi-csp-4-bci-controller-online.xml:** This scenario allowsreal-time feedback through Unreal VR. Now that we have created and trained the csp-spatial-filter.cfg and the motor-imagery-bci-config-classifier.cfg files we can use these as part of the online scenario which allows live control of the Anamorph-VR Unreal 3D Game by the user. Just run the scenario and launch the Anamorph.exe from within the Anamorph-VR folder – the user should be able to control the 3D symbol by thinking left or right.

**Bci-Controller-Input.xml:** This scenario allows a user to both test their setup and rerun previous sessions. To test your setup simply open the file in OpenVibe and configure the Generic Stream Reader to point to the file demo-motor-imagery-csp-1-acquisition-[2017.06.11-17.02.52].ov. Once the scenario is running launch Anamorph.exe and if the signals are being picked up the symbol will react to the right/left input. Any previous session can be rerun in the same fashion.

**Developer Notes**

VRPN (Virtual-Reality Peripheral Network - https://github.com/vrpn/vrpn/wiki): was used to connect the BCI controller within the Unreal Project to the OpenVIBE scenario. As far as we could see only 32bit versions of the VRPN client were available for download; and so you need to download the source code and build a 64bit version of this, as required by Unreal. For this task we used CMake: https://cmake.org/. VRPN is a device-independent and network-transparent system for accessing virtual reality peripherals in VR applications. You can find more information on configuration of VRPN with OpenVibe here: http://openvibe.inria.fr/vrpn-tutorial-sending-data-from-openvibe-to-an-external-application/

