Github link: <https://github.com/Jord159/IMDCGP206-Jordan_Carman>

12/10/18

Concept:

The concept I’m working on for this project is movement of a character in a game world using the Emotiv Epoc +. I plan on using mental commands for forwards and backwards movement. Once that works, I plan on using either the accelerometer or more mental commands to rotate the character.

19/10/18

Plan:

|  |  |
| --- | --- |
| **Week** | **Target** |
| 1 | Refamiliarise myself with API. Figure out how to use Cortex alongside Unity and set neutral state and mental commands |
| 2 | Apply mental commands to simple objects, ensure reliability in detection and execution of commands |
| 3 | Apply mental commands to forwards and backwards movement of a character in Unity, experiment with using accelerometer to rotate objects |
| 4 | If accelerometer viable: ensure accuracy in rotations, apply to character in Unity  If accelerometer not viable: apply more mental commands to character in Unity for rotations |
| 5 | Presentation of progress |
| Onwards | If work is delayed, complete incomplete work. After that, experiment with more ways to use Emotiv headset to control a character |

After reading through the Cortex documentation, the best way to get the Cortex API to work alongside Unity is by loading the required data into Unity via JSON files. I still need to find out exactly how often the data is updated but my hope is that it is frequent enough that I could load the data from the JSON files in a FixedUpdate call.

26/10/18

I’ve spent several hours today reading through the documentation and the example project on the Emotiv Github repository and am yet to figure out how the JSON requests are passed to the API. I’ll spend some more time throughout the week to try to figure this out but if I can’t find out how I can even pass requests to the API then I will be unable to do anything with the Emotiv headset and will need to change the project I’m working on.

02/11/18

I’ve finally found how to access the data via Unity and it appears to work fine, though I will need to double check this once I have access to the headset. It seems that the CortexExamples solution has methods created which automatically do a lot of the requests. Using the CortexAccess namespace within the Unity project, I should hopefully be able to access the methods I’ll need to access the data.

Once that was done, I looked for the format for the data I’m given. It appears the data is output as a block of bytes in a file stream. I’m not entirely sure where the data gets output to, it seems to just go into the build location for the project, but I won’t be able to check this until I have access to the headset. My plan was to prepare an object to receive and use the data from the headset in Unity so I could quickly check how everything works but I don’t think I’ll be able to until I can actually see what’s output and where.

In the meantime, now that I have a better understanding of how the example project works, I’ll read through the code to see how I can train the neutral state and add mental commands. This is something I’m going to need to implement very soon and so I need to understand how to do it as soon as I can.

09/11/18

I ran the cortex-example project with the Emotiv headset connected to make sure everything there works before I try to use it and to see how it works. Unfortunately, I kept getting an error on completion of Authorise. The error I was getting was an “Internal JSON-RPC error” with code -32603. The Cortex documentation doesn’t say any more on the error than that, and I was unable to find any information explaining the cause or the solution of this error.

15/11/18

Read through the introduction guides to WebSockets and JSON linked in the documentation to see if either of those would give me an idea on how to solve this error. There was nothing there. I then went back to trying to find any forum posts or anything about this error and realised that by including the hyphen in the error code, search engines won’t return any results including that error code. Removing the hyphen from my search got me some results for the error code, though none of the results I found were of much use to me. I’m going to talk to Victor later today to see if we can find a solution, or at least a workaround, so we’ll have something to present tomorrow.

20/11/18

Rereading the example project code and noticed a check to make sure the application can connect to WebSockets that I hadn’t noticed before. This could mean that the error I’m getting is less a JSON error and more a case of the system not being able to connect to WebSockets. I’ll need to check the port is open on the system and then try running the example code again to see if that fixes the error.

Did some more reading and it seems that WebSockets is connecting to localhost, rather than a website. The Cortex installer adds 127.0.0.1 emotivcortex.com to the hosts file. Since I keep all my files on my external hard drive, it’s possible that this addition was never made to the hosts file of the machines I’m working on. If not related to my current issue, this could prove to be a problem later down the line so I’m going to have to manually add that line to the hosts file of any machine I want to work on.

23/11/18

The 2 possible solutions I found Tuesday didn’t work so I’ve decided that that method was taking far too long to get working and decided to change my approach. I’m now going to try using node-red to connect to the BCI and get the data from there. Once I can access the data, I’ll look to see if my initial plan is still feasible. If it’s not, I’ll have to change what I plan on doing with the headset, otherwise I’ll continue with what I’m doing. Using node-red also allows me to work closer with Victor, which should help speed both of our work up and hopefully allow us to have something working by the end of this project.

07/12/18

Using node-red I can access the data from the headset. This can then be output to a text file which can be used in other programs. I’m currently having an issue where only 5 of the 6 metrics output are being written to the text file. I’m trying to keep the file so only the most recent output from the headset is in the file so I’m overwriting the file with the first stream then adding all subsequent ones into the file afterwards. For some reason, the overwrite is clearing the file but not adding the metric to it. Once I figure out a way around this, I should be able to easily get the data from the headset into Unity. The only downside I can see to this method is that unless I am able to automate it on runtime, the user will have to manually start node-red and create the flow.

13/12/18

I’ve decided on using separate text files for each metric, rather than having them all in the same file. I’ve not noticed any missing data this way so it seems a far safer way to go than having them all output to a single file. Also, if a bit of data does fail to get updated for whatever reason, the file is still there with the previous value that can then be used instead, preventing errors occurring that way.

The metric data imports nicely into Unity, and is updated every 10 seconds, to match the update of the metrics. At the moment I’m just printing it to the debug log, I’m not really sure what I could do to with that data in the short amount of time I have left before the deadline. It’s clear that with the 10 second update time, character movement is unfeasible using the performance metrics. I think that mental command checks are more frequent, and probably better suited to moving a character, but that would make the program dependent on the BCI, which I’d prefer to avoid. I’m doubtful that I would be able to make the required checks to minimise the chance of errors in the game by trying to use a command that hasn’t been trained.

14/12/18

I’ve been trying to think of a way I can use the metric data in a gamey way that I could implement before submission but the only thing I can think of is an idea I had at the start of the semester: using the metric data to trigger certain events in a game. I think this sort of technology could prove useful to games which rely on emotion. You could use the metric data to trigger events, for example, when stress gets below a set threshold in a horror game, an event is triggered to scare the player or when they’re starting to get disinterested, spawning in some lore.

This technology could also pave the way for inclusivity for people with physical disabilities. Mental commands don’t require any physical action so they could be used as an alternative to using a controller or a keyboard and mouse.