

Final Report

VR Learning Environment with Real-Time Brain Signal Monitoring

Brendan Jenkins, Ethan Li, Jacob Mitchell, Yumna Rizvi, Zayne Frabutt

FS: Nathalia Peixoto

CC: Peter Pachowicz

ECE 493-001

April 30th, 2021

Executive summary

As technology's grasp over society continues to strengthen, deeper applications of its use are considered for everyday life. The rise of the Coronavirus pandemic shocked the world into a socially distanced lifestyle, highlighting shortcomings of lesser-used technologies caused by the mass influx of user volume. Since many technological applications were implemented on a smaller scale or as alternative solutions, ease-of-use and efficiency suffered from this transition. One of the biggest challenges faced by society is centred around education, causing students and teachers who have only experienced traditional classrooms to be pushed into implementing different methods of instruction. Our project stems from the limitations of virtual learning, providing a solution to enhance the online education experience.

We have created a project centred on distanced learning, providing a virtual environment that allows students to dive deeper into their educational experience. The VR modules created for this project are targeted towards STEM classes, allowing for supplemental exposure to course material through module interaction. To enhance the virtual experience for the user, the Leap Motion controller was implemented, allowing for hand movements to control environmental interaction. This solution allows for an immersive experience that holds the attention of the user when learning outside of the classroom. Aside from providing benefits to the students, several features have been provided to enhance the teacher experience too.

The virtual learning environment we have created is supplemented with brain signal monitoring using the Emotiv EPOC+ headset. This headset monitors specific points on the head, providing insight into a student's perception of the module. For ease of use, the raw brain signal values are measured from the user and displayed in a format that allows for instant teacher interpretation. These values are accessible by the teacher in real-time. The learning module prompts the user to connect to a server, allowing for the teacher to monitor their entire class during a lesson. This connection allows the option for the teacher to control a student's module settings before and during module runtime. From the data collected, a teacher can tailor their lessons to meet student needs. This information can also be used to filter the struggling students out from the rest, allowing for the teacher to provide additional aid. Overall, our project provides an in-depth solution to challenges stemming from distance learning. The provided learning modules allow for an immersive student experience while allowing for teachers to obtain their students' physiological data to improve lesson scope and address struggling students.

Table of Contents

<u>Topic</u>	<u>Page Number</u>
1. Executive Summary	1
2. Table of contents	2
3. Approach	3
4. Technical Section	9
5. Gameplay Functionality	11
6. Experimentation & Validation	23
7. Results	42
8. Other Issues	45
9. Administrative Section	48
10. Lessons Learned	51
11. References	53
12. Appendix A: Draft Proposal	55
13. Appendix B: Design Document	75
14. Appendix C: Software printout	101

Approach

Project Origin

Adaptations to the Coronavirus pandemic have forced society to live socially distanced from one another. This lifestyle caused a shift from a traditional classroom environment to online learning classrooms. Many people have never experienced education using virtual methods, causing issues for both students and teachers. Many students, especially grade school students, are used to learning predominantly through teacher instruction. Learning in a classroom can limit distractions, allowing for greater attention directed towards the lessons. However, distance learning allows students to attend class from their households, creating an environment with several distractions. This can cause issues with student performance and understanding of core concepts required for higher-level education.

Teachers also face similar issues from online instruction, many of which have never experienced distance education before Covid. The methods used to teach students in a classroom environment are limited, creating a necessity for teacher adaptation. Online instruction also creates a disconnect between students and teachers, causing teachers to rely solely on grades to determine a student's necessity for additional help. Many teachers for STEM courses provide hands-on methods for learning to supplement lessons but suffer from limited options due to the nature of distance learning.

Our project aims to address the issues outlined above. For students, we have created an environment that helps limit distractions. This environment also provides a more interactive learning experience, providing custom modules that can supplement lesson plans. From a teacher's perspective, our solution provides methods to more clearly understand the physiological state of students when attempting to complete their assigned tasks. This can provide the opportunity for teachers to address struggling students before issues are reflected in their grades. Overall, our project aims to create an environment that promotes education through hands-on learning, while allowing teachers to better understand the impact their lessons have on individual students.

Solution to the Problem

There are several components that our project incorporates to address the needs discussed above. Our design incorporates a virtual reality environment viewed through a NeuTab headset. Using a virtual reality environment allows students to experience supplemental lessons in an environment

with fewer distractions. The virtual environments are created in Unity, allowing for custom modules to be created for the needs of a class. Also, to provide a more hands-on experience for students, the project incorporates a Leap Motion controller. This controller is mounted on the NeuTab, allowing students to have hands-on experience within the created modules. This helps address multiple problems that students face while learning from home.

We have incorporated several features to benefit the teaching experience as well. First, our project centres around the use of the Emotiv EPOC+ headset. This headset contains fourteen sensors that can read raw brainwave data from students. Using the Emotiv Cortex API, these numbers are processed and categorized in the following metrics: stress, engagement, interest, excitement, focus and relaxation. To provide this information to the teacher, we have created a teacher hub that links the entire class to one location. This allows the teacher the ability to monitor multiple students at once, providing the ability to determine which students are struggling the most with the material. We have also included methods to allow the teacher to modify the settings of a student's current module and talk to the students specifically through a designated voice channel. Using our project allows for a teacher to assign hands-on assignments, providing a method to view student progress during completion. This method solves issues centred on disconnect, providing insight for structuring future lessons and addressing students in need.

Alternative Designs

Google Cardboard vs Unity VR View:

When choosing an approach for displaying our VR environment, we were tasked with choosing software to split the screen for VR view. Google Cardboard view allows for splitting the screen in Unity with an SDK provided by Google. We considered this method first because we were unable to figure out how to use Unity VR view in the version of Unity our project was constructed in. The biggest issue with Google Cardboard involved compatibility issues with displaying content on a computer screen. This meant that we would need to choose whether we wanted to display our project on a phone or an LCD screen. Since we did not own a phone to test on and fully testing the project with Google Cardboard requires consistently exporting our project, we decided to use the Unity VR View.

Leap Motion vs Custom Controller:

To create a full experience for our VR environment, we needed a method for the user to interact with the modules. We were considering a custom controller that could be tailored to our projects specific needs. This controller would have used an accelerometer to control a cursor in Unity

while housing a button for selecting the object the cursor is hovering over. A Raspberry Zero would obtain the data from both input devices and use it to control the module. This controller would be housed in a 3D printed case that would protect all the components while providing a comfortable grip. The following figures demonstrate the concept for the custom controller design. The difficulty of the implementation in Unity coupled with the efficiency of the Leap Motion controller pushed us away from using a custom controller in our design.

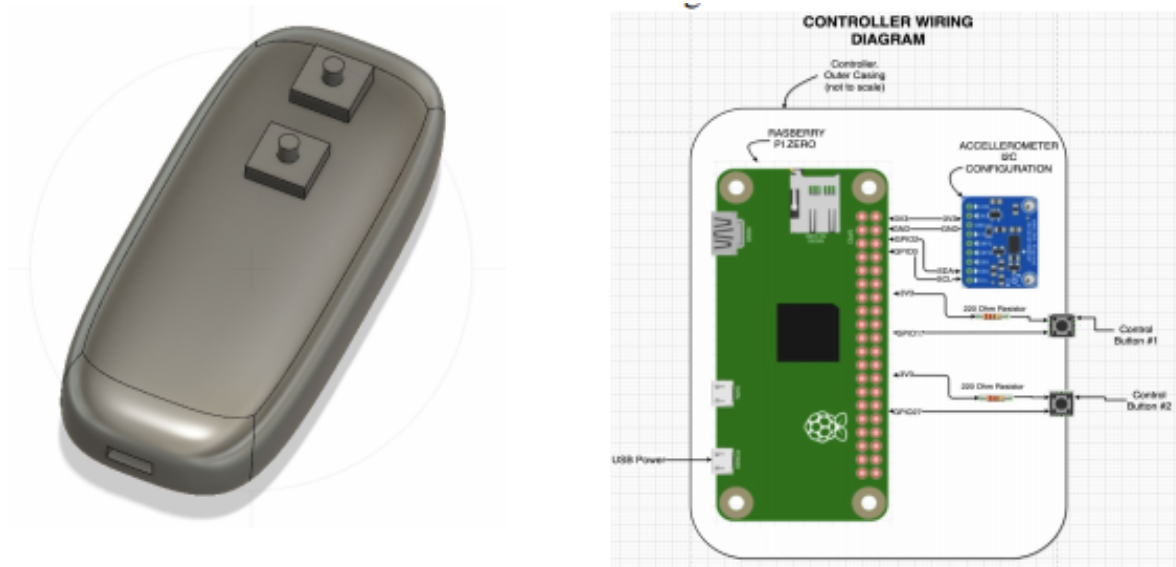


Figure 1: Design for Custom Controller

NeuTab vs 3D printed headset:

When originally discussing concepts for this project, we had had the idea to create a custom 3D printed VR headset. This idea seemed attractive, as we could create a headset that could satisfy the needs for sensor placement. Creating a VR headset also provided flexibility because the spacing between the Emotiv EPOC+ and Neutab are tight. However, there were several problems with creating a VR headset. First, methods for adding parts that weren't 3D printed were difficult. We were unsure how to properly add cushions, straps, and lenses to the headset. Another issue stemmed from a lack of experience with 3D printing. This idea sounds nice on paper, but without proper implementation, the VR experience would be diminished. These reasons led us to choose the NeuTab headset for the implementation of our project.

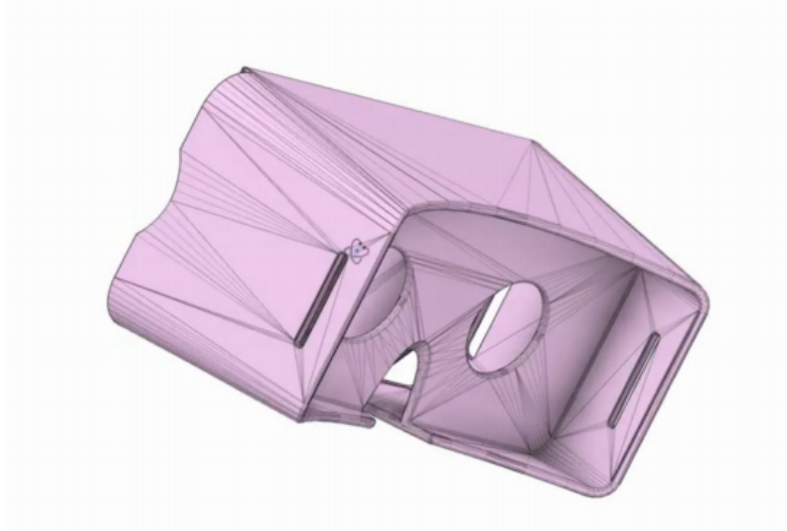


Figure 2: 3D Model for Our Custom Headset

Emotiv vs Muse:

When deciding which brain wave monitoring headset to use for our project, several things were needing to be considered. First, the amount of space needed for proper monitoring was considered. The Muse is a headband, sensing the forehead to obtain brainwave data. The Muse takes up significantly less space on the head than the Emotiv. However, since the Muse has fewer sensors than the Emotiv, the data that can be obtained is significantly less. The Muse also falls short of the Emotiv headset because Emotiv has created SDKs for use in Unity, allowing for Emotiv use for brain monitoring and tilt detection. Overall, the Muse would be a decent backup choice for the Emotiv; however, due to shortcomings in overall functionality mixed with the lesser potential for data gathering, we decided to use the Muse for our project.

Contribution of Team Members

Brendan Jenkins

From the beginning of the project, Brendan helped with the design of the system and with documentation. He also assisted with the creation of alternative solutions to the project as well. Another task was to 3D model different designs if alternative solutions should be needed, while also 3D printing at least one of the designs to experiment with sizing and visualization. A large task he was able to complete was creating an emotiv cortex app to send our data packets into the unity game software successfully. Brendan helped with being able to use the 9-axis IMU (head tracking unit) data consisting of accelerometer data, gyroscope data, magnetometer data, and quaternion data that fed into our data stream to aid with the head tracking software script in unity. Lastly, he helped validate all data streams used within the project using MatLab and Simulink to figure out bounds and noise values.

Ethan Li

Ethan assisted with the management of the team. He created the outline of the tasks throughout the year along with a timeline of completion per task. He led the meetings every week, ensuring all group members were on task and providing help to those who had long roadblocks that would disrupt the timeline of the project. Along with assisting with management, Ethan worked on creating the module designs within unity on a separate unity instance than the main one. The creation of the modules included functionality, aesthetics, and scripting. This was to guarantee that the modules were functional with and without the extra components added.

Jacob Mitchell

Jacob was the project manager for the team. He facilitated all communication with the faculty supervisor as well as the course coordinator. He managed the weekly task forms and kept up to date on any documents that needed to be submitted throughout the class. Alongside this, he created the teacher hub and set up the networking connection between the student and the teacher. This included implementing a student to teacher voice chat that allows for the teacher to speak directly to students. He was also the one to implement the VR interfacing that allows for a VR goggle view to be displayed using a regular screen without built-in VR capabilities. Lastly, he was responsible for compiling and editing the multiple video presentations throughout the course.

Yumna Rizvi

Yumna assisted with data analysis for cleaning, extracting and modelling data and performing validation strategies alongside her team members. Along with assisting with documentation and presenting alternative ideas, she assisted with tasks such as creating high-level data diagrams. She helped ensure documentation maintained a consistent flow of ideas. Additionally, she helped in problem-solving in areas such as establishing a server and virtual machine for the implementation of the project. She highlighted issues and risks and helped outline the need for deliverables and deadlines throughout the year. Additionally, she performed research on the physiological aspects and architecture of the project. She assisted with the design of the hardware components and EEG signals pattern research. Lastly, she helped analyze the various strategies of testing the project whilst training alongside VR professionals as a VR quality assurance intern in an industry-level environment to gain insight into the scope of the project.

Zayne Frabutt

Zayne was responsible for creating module designs on the main unity instance. This involved creation of several modules, scripting, and implementation of Emotiv and Leap Motion SDKs. When implementing the SDKs, he helped decipher code to allow for efficient use tailored to the project's specific needs. Zayne was also responsible for the bulk of Leap Motion testing, as well as assisting in gathering data from the Emotiv. This involved designing experimentation plans to push the project along. During the early stages of the project, he played a big role in researching potential solutions for the project, developing lists of solutions and alternative solutions. Zayne was also responsible for the majority of the written content found on the project's website. Additionally, he helped proofread and finalize documents to ensure submission quality.

Technical section

Our system incorporates three major components: the Epoc+, leap motion, and unity engine. As a top-level design, the Epoc+ and the leap motion will take in data streams from the user and feed them to the unity engine. The unity engine will host the learning modules and the hub, creating an interactive interface for both the user and the educator. This top-level overview will serve as the basis for the project.

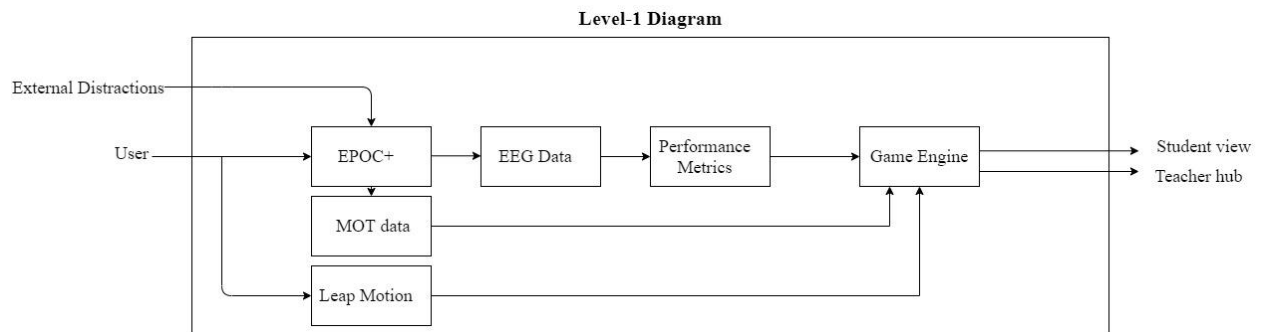


Figure 3: Level-1 diagram

This level 1 diagram displays the system overview from the top-level design with descriptions of what each component will be used for. Taking a step into it, the Epoc+ is used for motion data and the EEG data from the user. The leap motion is used for hand movement data from the user.

These data streams can be visualized in figure 4, describing the initial system architecture.

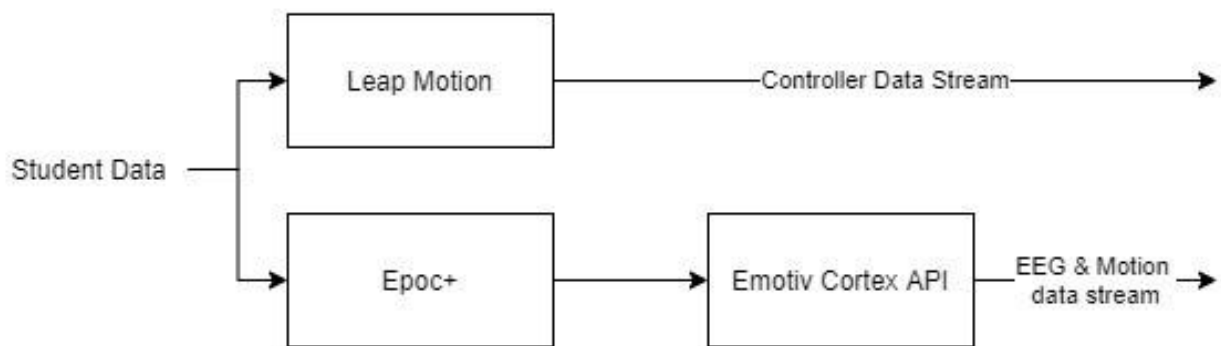


Figure 4: System architecture 1

With figure 5, the data streams are connected directly to the unity engine. The controller data will allow the user to interact with the module design within the engine. Using the motion data, the user will have control of the x-axis camera rotation, and with the EEG data, the engine will monitor the performance metrics of the user. Once the data streams are processed, the scenes are produced and outputted to both ends of the users.

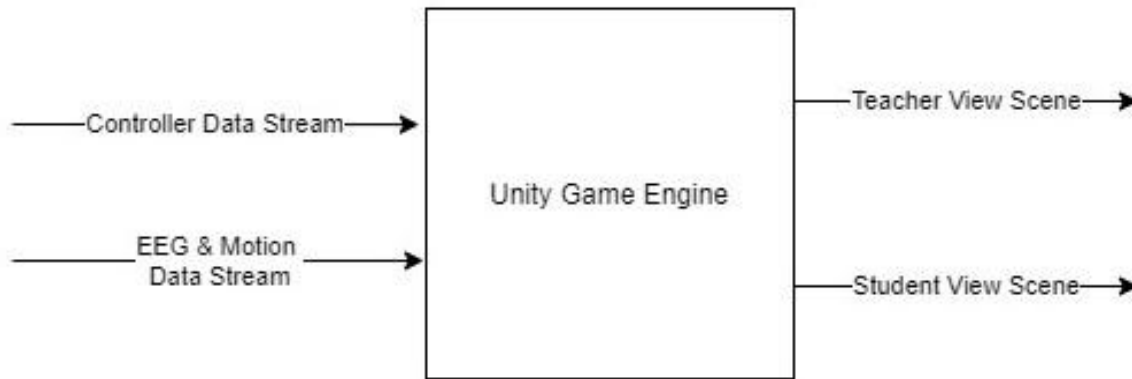


Figure 5: System architecture 2

Figure 6 portrays how the scenes will be viewed by both users.

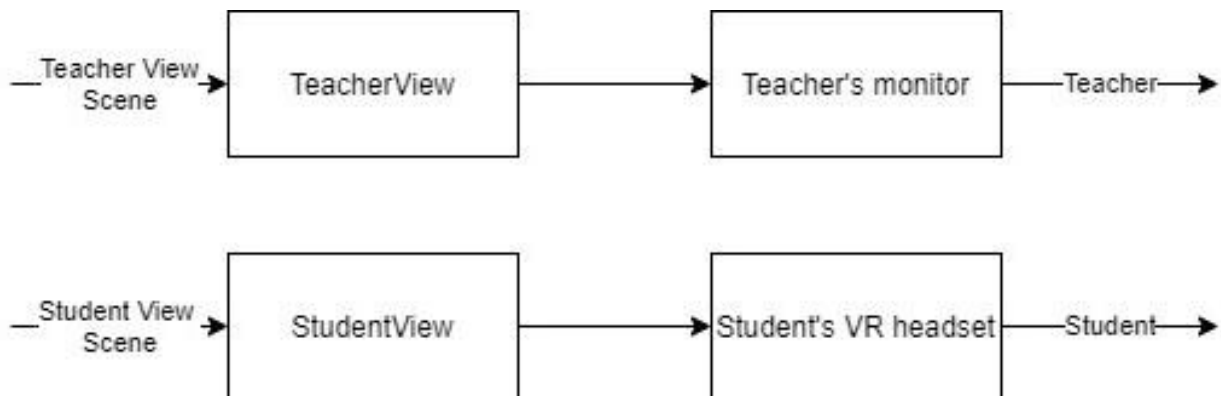


Figure 6: System architecture 3

Gameplay Functionality

Short Introduction

We currently have the Unity scenes built in the following order: initial setup, module selection, and selected module. The initial setup prompts the user to follow instructions to properly calibrate the EPOC+ and subscribe to the correct Cortex API channels. After completion, the user will jump into a classroom scene which is used for module selection. Currently, we only have one learning module, but the canvas on the back of the player's hand can be programmed to allow the user to scroll through and select existing modules. Upon module selection, the student is then connected to the teacher, giving the teacher access to the performance metrics and module settings. Currently, we have designed a module that can supplement elementary addition and subtraction. Randomly generated equations will be created, and the user will have to grab different blocks that are moving towards them to arrive at the correct answer. Upon completion of a specified number of levels, the application ends, storing the performance metrics in a file for the teacher to access at a later point in time.

Initial Set-Up

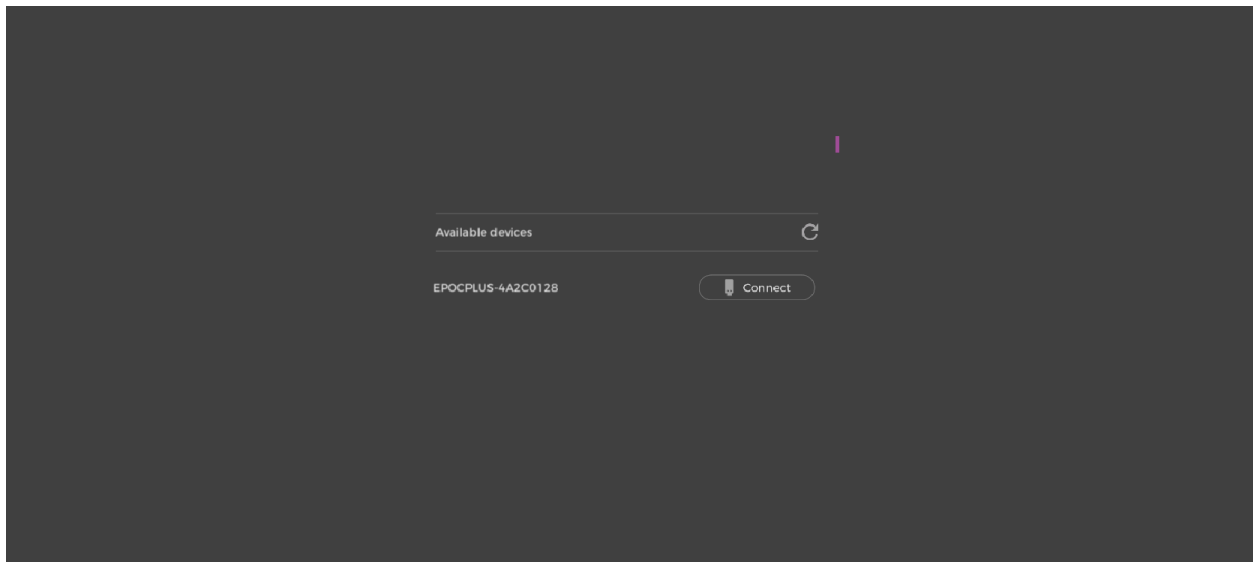


Figure 7: Bluetooth Connection for Emotiv in Initial Setup

The first thing the student will do before entering the learning module selection will be connecting the EPOC+ and Cortex data via bluetooth dongle as shown above in figure 7. After a successful connection through bluetooth, the scene will automatically take the student to the next stage, prompting the user to put the EPOC+ on their head and ensure all contact points are properly calibrated. This is denoted using the colors green, yellow, and red. Green represents the strong connection, while red connection signifies a weak sensor connection. Shown in figures 8 and 9 is an example of contact connections versus bad contact connections.

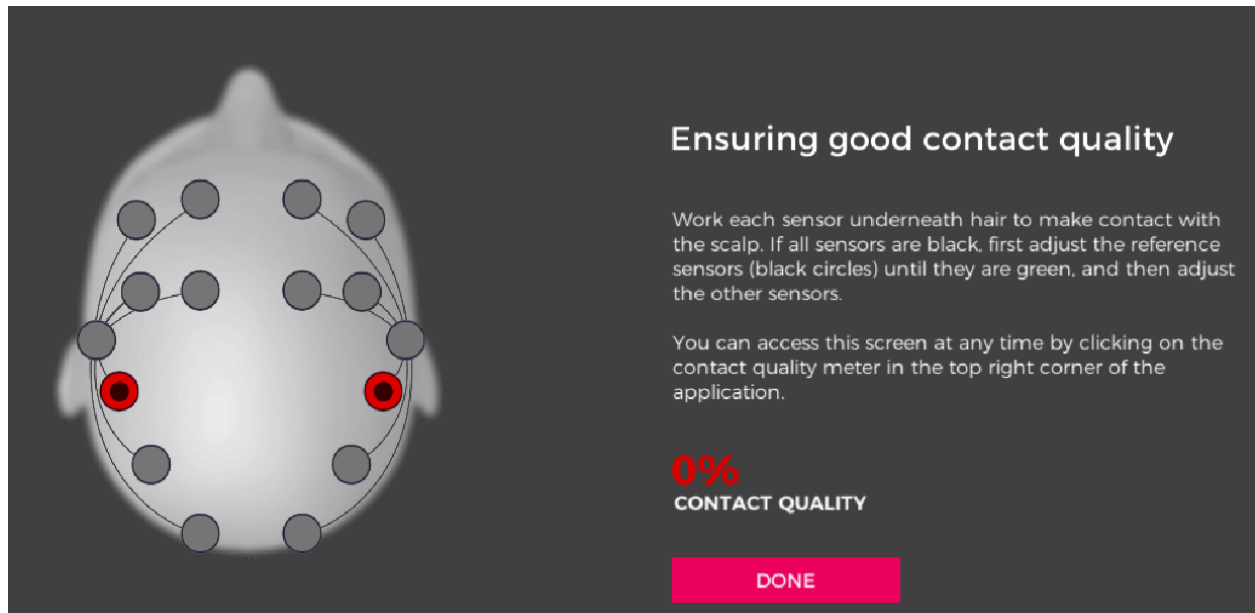


Figure 8: Showing bad contact quality

(Next Page)

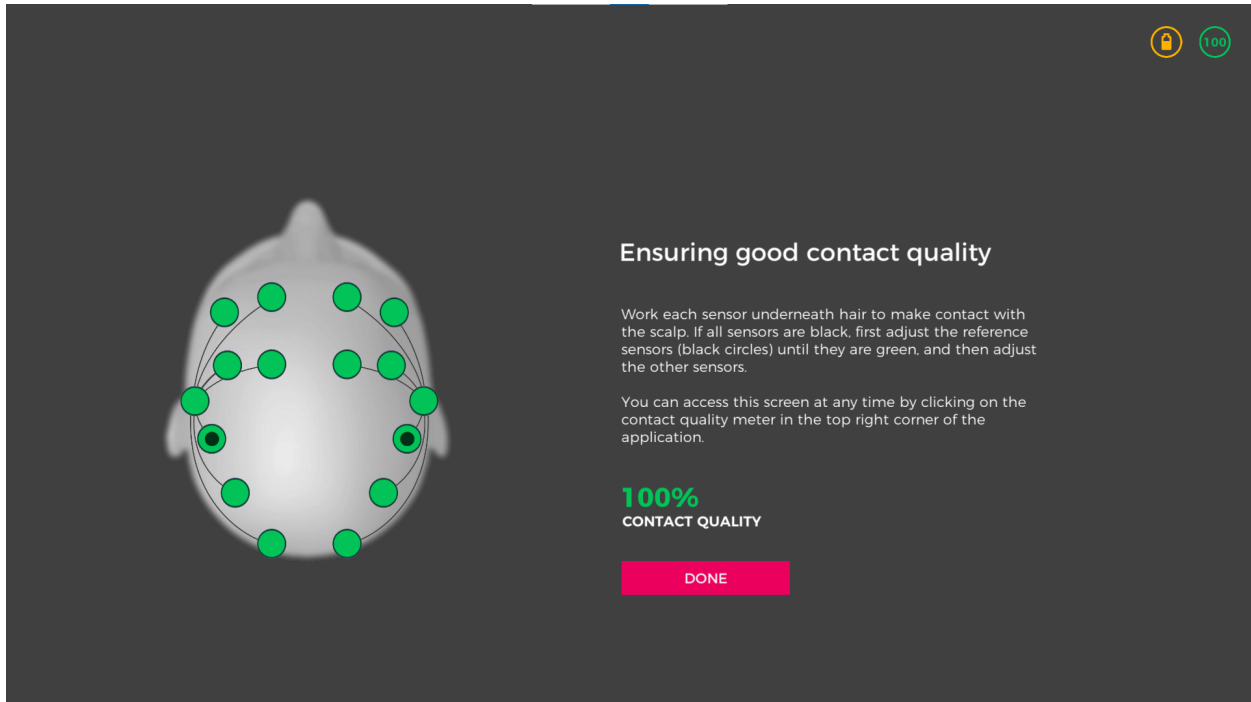


Figure 9: Showing good contact quality

There will also be a percentage indicating how well your connections are on a scale of 0-100 percent with 0 percent being no connection and 100 percent indicating perfect connection for picking up brain signals. After all contacts are connected the user will click “Done” and will be taken into the scene in which the user will subscribe to all the data streams. The user will click three buttons for subscription of EEG data, motion data, and performance metrics shown as “Subscribe EEG”, “Subscribe MOT”, and “Subscribe PM” respectively shown in figure 10 below.

(Next Page)



Figure 10: All data metric subscriptions

Lastly, the user will click “start game” and be put into the classroom environment shown below in figure 11. The user will put on the VR headset and then they will select a learning module to begin their game using the settings menu attached to the users hand



Figure 11: Classroom starts room

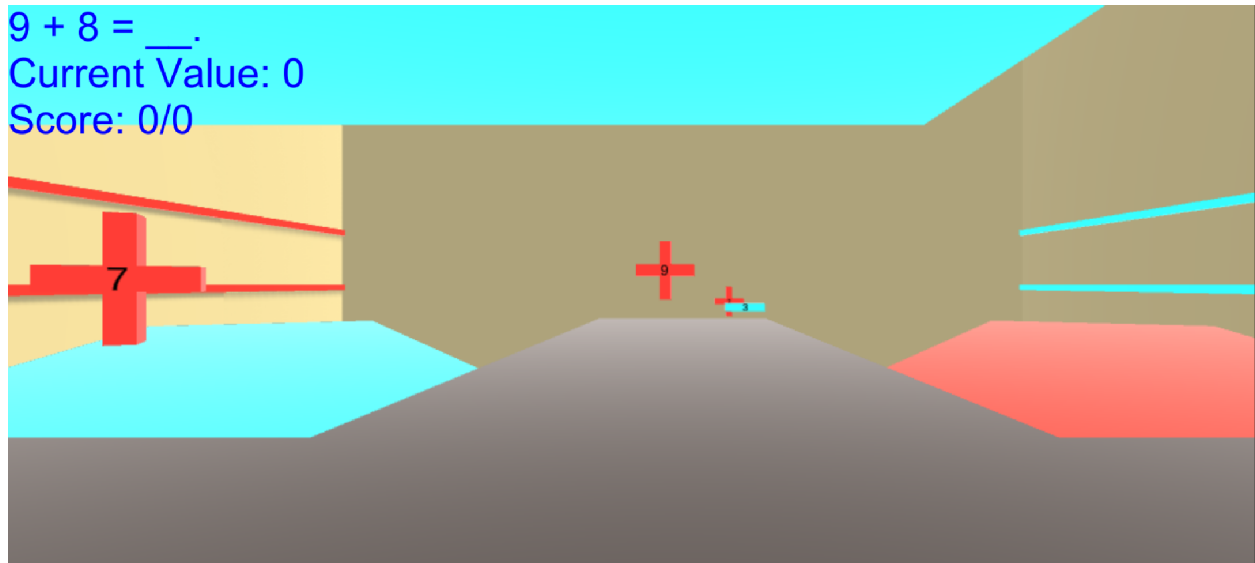
Learning environment

Figure 12: Module starting

The learning environment is where the student and the teacher will perform most of the module interaction. After choosing the module in the set up phase, the student will be spawned onto a platform with hand models connected via the leap motion data stream. The platform will extend into three lanes beyond the student. At the ends of the lanes, a “plus” or a “minus” model, with a randomized numerical value in a range correlating to the equation on the GUI of the student’s VR view, will be spawned in and sent towards the student. The student will then be able to interact with the game objects to solve the equation by grabbing the models with the leap motion hand models, as can be seen in figure 12.

(Next Page)

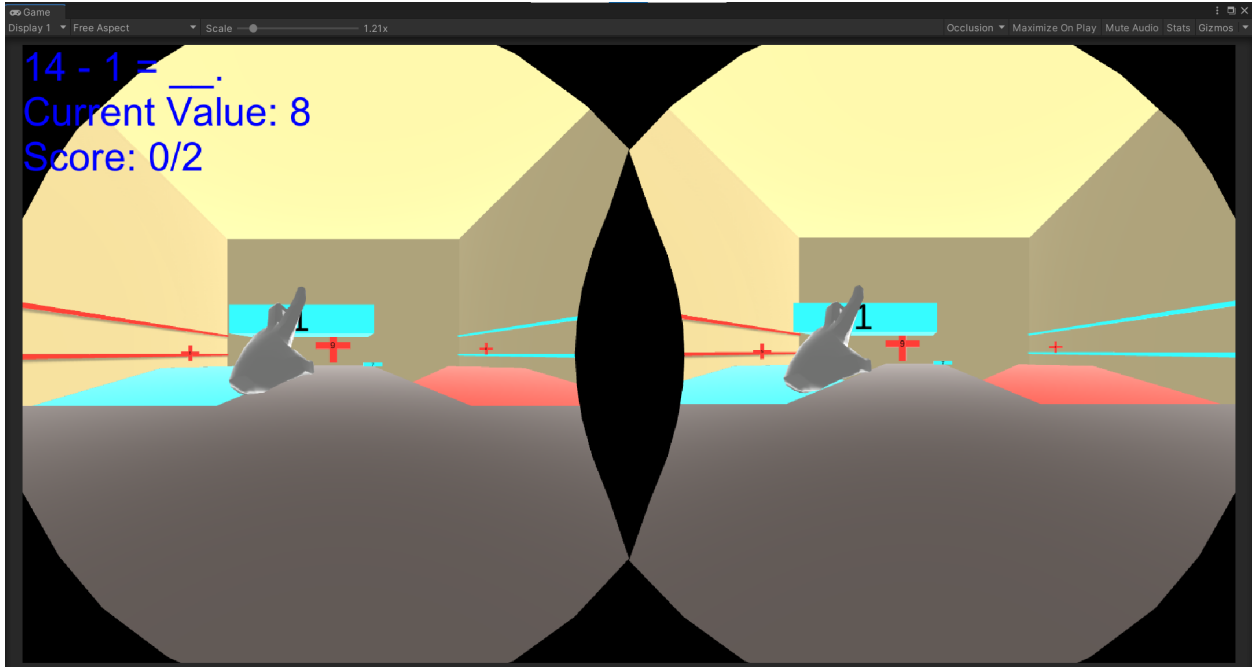


Figure 13: Interaction of models

In figure 13, the models are currently displayed as their symbol of operation to the equation. If the player grabs a “plus” model, the value will have an operation of addition and be added to the total count. Similarly, the “minus” model will set the operation to subtraction and decrement the counter. The counter will be used to solve the equation set on the GUI and can be modified as many times as the student grabs blocks, unless the value reaches the out of bounds area (Adding above the equation’s result or subtracting below 0). Once the student reaches an exact match, the module will notify the student of their accomplishments and move on to a new equation. Whether the student correctly reaches the answer or stumbles upon an out of bounds count, the total amount of correct equations will be incremented or remain the same, respectively. This count will be proportional to the total number of attempts made and will be displayed on the GUI of the student and the teacher.

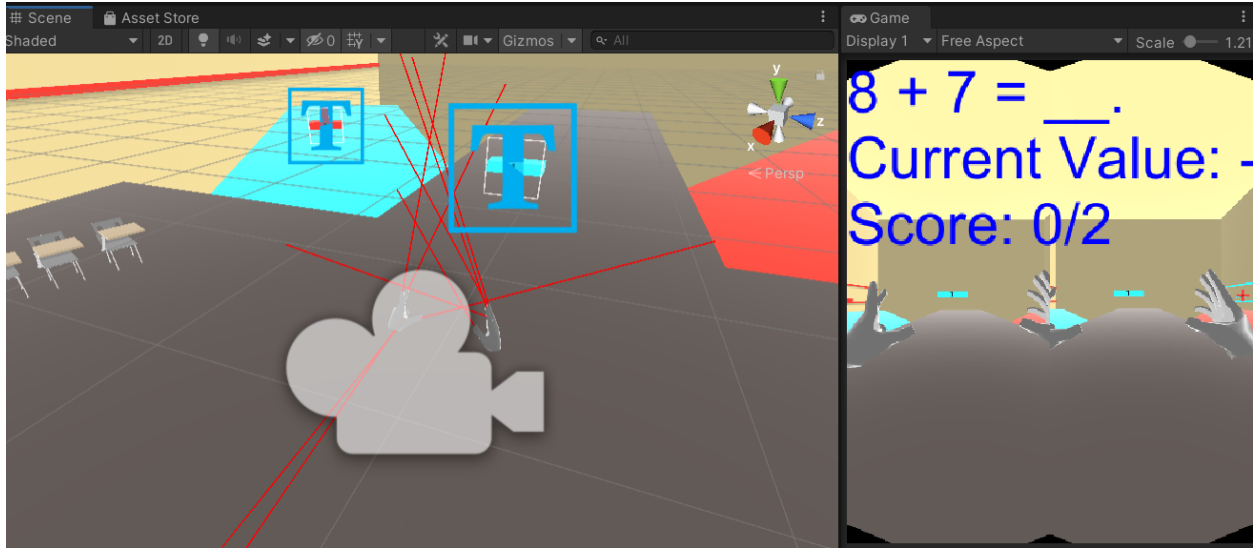


Figure 14: Raycasting fingers

To create the module, we connected several scripts to multiple game objects. With the operation models, we incorporated a spawning script to randomize the location of the model and set the settings for the model (i.e. value, operation, speed, etc). The models will then have their position updated to move forward each frame, at a rate proportional to the speed and their rotational location. For the student model, the leap motion SDK was imported, allowing us to visualize the hand data stream within unity. The hand models were then modified to include ray casting, as seen in figure 14, a method of intersecting with any collision model and storing their logistics data. To bring all of the scripts together, we included a “game manager” that interacted with all of the script’s functionality like setting the model logistics and collision data.

The module will concurrently monitor the student’s performance metrics during the time it takes to solve an equation. This data will be sent directly to the teacher hub where the settings can be interacted with by the teacher using the “game manager” to aid in the student’s experience.

Teacher hub

Talk about the data being sent to a buffer, and how it will be writing to a csv file for storage of how the student was doing over time

The teacher hub is the program that the teacher will use to monitor the students during their usage of the modules. It will provide data for the teacher on each of the performance metrics, including stress, engagement, focus, interest, relaxation, and excitement. In order to start up the teacher hub, you simply click “Host Button” and then the students’ information will appear.

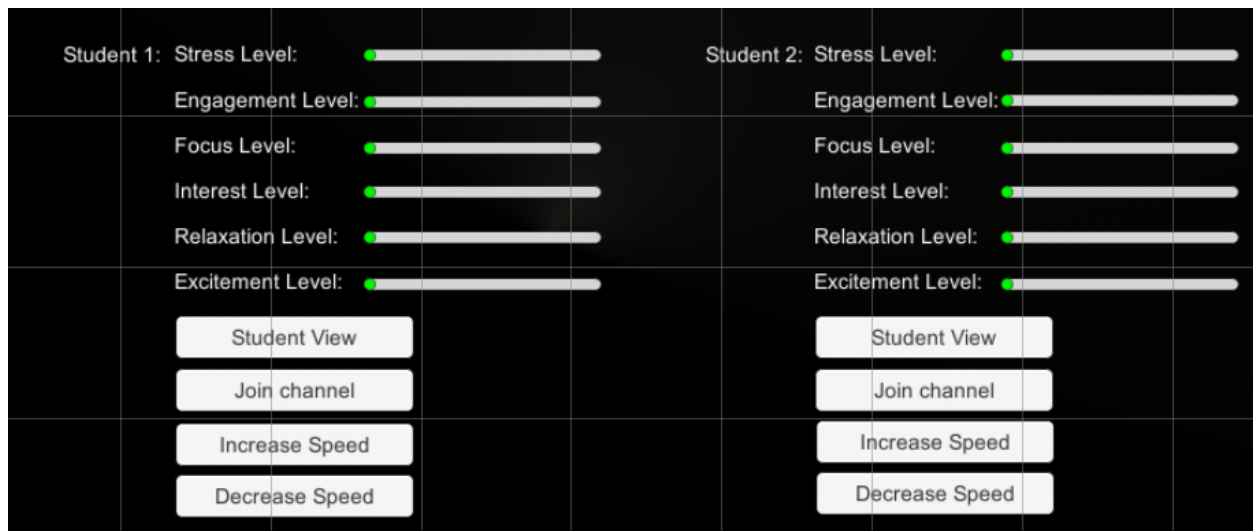


Figure 15:Teacher hub view

When viewing the student view, the teacher is able to join a voice chat with the student. This allows the teacher to directly interact with the student and see how what they are saying is affecting them in real time. The teacher is also able to directly increase and decrease the speed of the boxes on the students’ side. This is also adjustable from the students’ side using a slider on their hand models. All these functionalities are shown above in figure 15.

PM.Engagement.Scal	PM.Excitement.Scal	PM.Stress.Scal	PM.Relaxation.Scal	PM.Interest.Scal	PM.Focus.Scal
0.917205	0.277902	0.676064	0.327334	0.350316	0.369501
0.800716	0.322548	0.866307	0.460014	0.539285	0.424824
0.644445	0.322788	0.806286	0.680994	0.644459	0.334679
0.590174	0.395787	0.669144	0.650107	0.657602	0.339635
0.463476	0.440332	0.509192	0.536042	0.620387	0.400597
0.661912	0.397843	0.400189	0.424939	0.454383	0.411995
0.816405	0.425506	0.238114	0.259946	0.353901	0.431899
0.6905	0.484022	0.250412	0.252236	0.386651	0.459963
0.662549	0.647653	0.261251	0.254774	0.411395	0.492689
0.856763	0.456325	0.468636	0.253511	0.493941	0.607009
0.872325	0.457065	0.441473	0.251914	0.472444	0.582382
0.803406	0.55625	0.450398	0.255424	0.479727	0.592593
0.794142	0.563633	0.388288	0.25639	0.4889	0.566458
0.735547	0.511184	0.299302	0.24085	0.464583	0.540391
0.662199	0.493248	0.458142	0.34712	0.549534	0.527987
0.837185	0.989158	0.541178	0.239378	0.734749	0.465175
0.812131	0.954104	0.746358	0.292744	0.59495	0.794103
0.85344	0.685225	0.417912	0.353512	0.465615	0.667593
0.810197	0.359382	0.275734	0.214269	0.464385	0.634734
0.723741	0.558763	0.324246	0.227433	0.515536	0.576406
0.639417	0.329753	0.39257	0.349998	0.460485	0.502039
0.661249	0.287214	0.354637	0.249468	0.50896	0.481648
0.70276	0.409877	0.302544	0.232247	0.520109	0.457008
0.741844	0.688583	0.452365	0.230511	0.568365	0.550612
0.836027	0.656119	0.522057	0.297744	0.576274	0.622767

Figure 16: csv file of performance metrics data

The performance metrics will also be exported to a separate csv file as shown in figure 16. This will allow the teacher to go back and review the students' performance metrics values throughout the session.

Right now, there are only two students on the menu, but this could be easily increased as the need comes along. We only need two students for our demonstration. Currently, the networking protocols we implemented do not have the capability to allow each student to have their own unique rooms and have the teacher join them, but this is going to be implemented soon within the MLAPI networking protocol, so it could be added in the future.

Software Datapath

The software datapath of our system requires the collaboration of several application programming interfaces (API's), software development kits (SDK's), and scripts created by the group. There are four main data paths that we needed to create: head tracking, cube interaction, game management, and the teacher hub. Figure 17 below shows our implementation for mapping real-world quaternion values into Unity quaternion values. The data subscriber reads motion data from the Cortex API and transfers it into the camera movement script. This script uses Euler Angles to change the quaternion values from the data stream to Quaternion objects. These Quaternions are then applied to transforms and attached to the leap rig, providing a camera rotation effect.

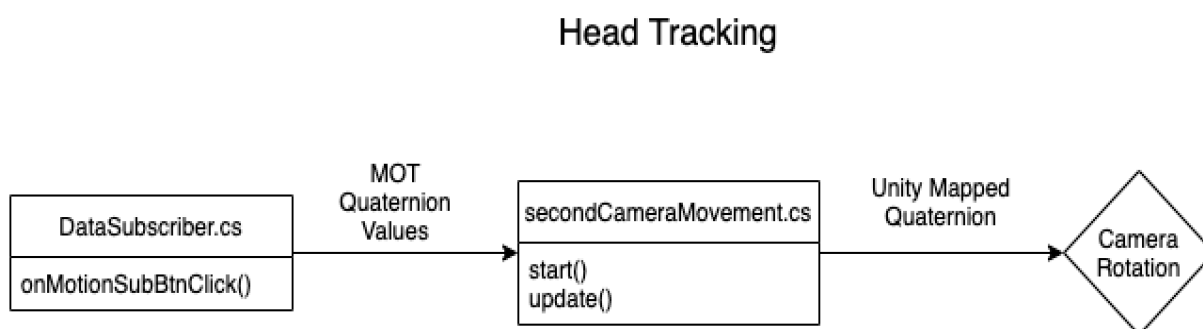


Figure 17: Head tracking software functionality

The next major data path we needed to address followed game management. This requires both cube interaction (figure 18) and game management (figure 19) scripts. The difficulty for the module revolves around the speed blocks move towards the player. Our module has two methods to modify this speed. The teacher has buttons mapped to each student for incrementing and decrementing block speed. The values increment the speed by .2 and are bounded between .5 and 2.5. These values are passed through the settings menu into the box movement scripts. After the boxes are spawned, they can be destroyed in two ways. Boxes are automatically destroyed after a certain amount of time to prevent lag from large quantities of Objects in the environment at one time. The other method of destruction revolves around ray casting. Each finger has a ray extending from it, detecting when it touches another object. In rayCastTest.cs, an object contacting a ray is destroyed, outputting the value associated with the block.

This value is passed along and sent into the game manager. The value is then added or subtracted from the current total depending on the type of block that is contacted. This value is displayed in a GUI for the user, allowing the user to determine the next block to grab. If the total value is less than the correct answer, the game allows the user to continue playing. A value equal to or above the correct answer will cause the game to restart. If the value is incorrect, only the completed number of games will increase, denoting an incorrect answer in the total score. On the other hand, if the player collects a value equal to the correct answer, the game will increment both the correct and completed values. The GUI allows for the user to track the equations attempting to be solved, the current value they have collected, and the current ratio of correct to completed iterations of the module.

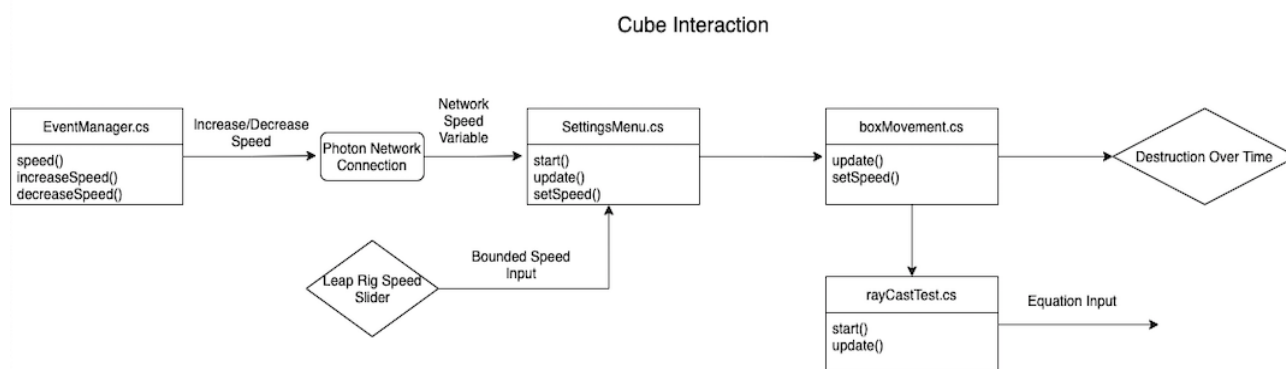


Figure 18: Detailed cube software functionality

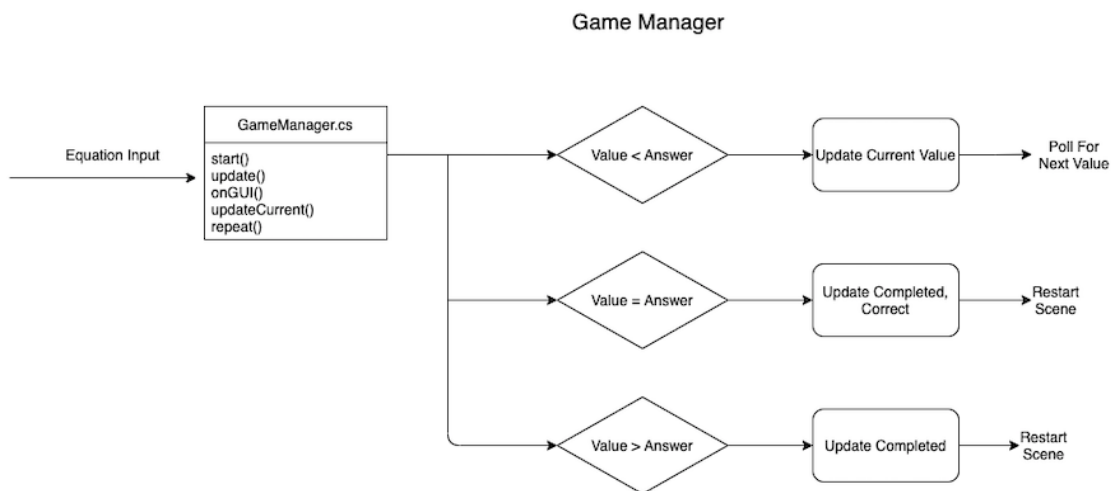


Figure 19: Game manager software functionality

The last data path we needed to build centered around the teacher hub as shown in figure 20. The performance metric data stream is sent through the settings menu into the player prefab. To use network transports a default player prefab needed to be created. This prefab is the same on all Unity instances, containing shared scripts that pass around variables. EventPlayer.cs passes the performance metric data from the Event Player to the Event Manager. The values are then displayed in sliders for the teacher to view and saved in a CSV file to allow for accessing data at a later point in time.

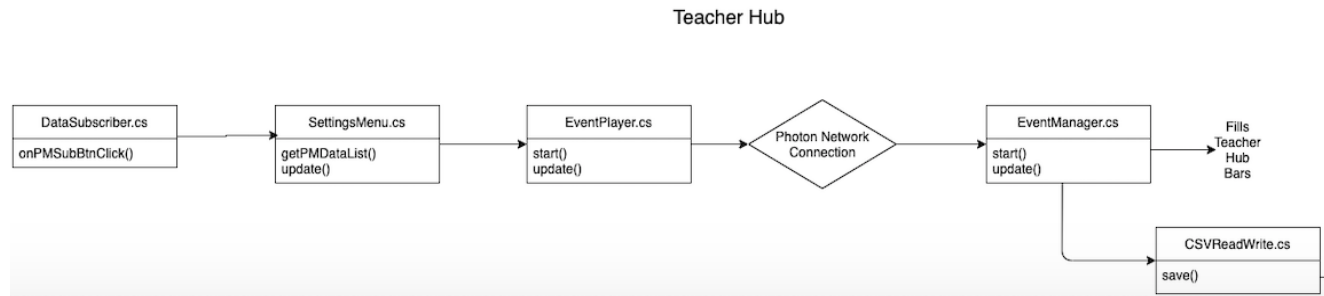


Figure 20: Teacher hub software functionality

(Next Page)

Experimentation & Validation

Test #1 - Head Tracking

Discussion

One of our experiments was to use the data stream coming from our EPOC+ headset that consisted of quaternion data (q0-q3), gyroscope data, accelerometer data, and magnetometer data and mapping it into our 3D environment for head tracking. The gyroscope data, accelerometer data, and magnetometer data all combined to create the quaternions. The quaternions are just representations of a vector in 3D space of the form $w + xi + yj + zk$, which will define the head movement. The variables w, x, y, z represent $q_0, q_1, q_2,$ and q_3 in a range of $[-1,1]$ respectively. The purpose of this experiment was to use these 3D quaternion vectors and convert them to Euler angles of pitch, roll and yaw for the rotation of head movement in 3D Space. This experiment also helped us solve an issue called gimbal lock which will be discussed below.

Apparatus from Experimentation

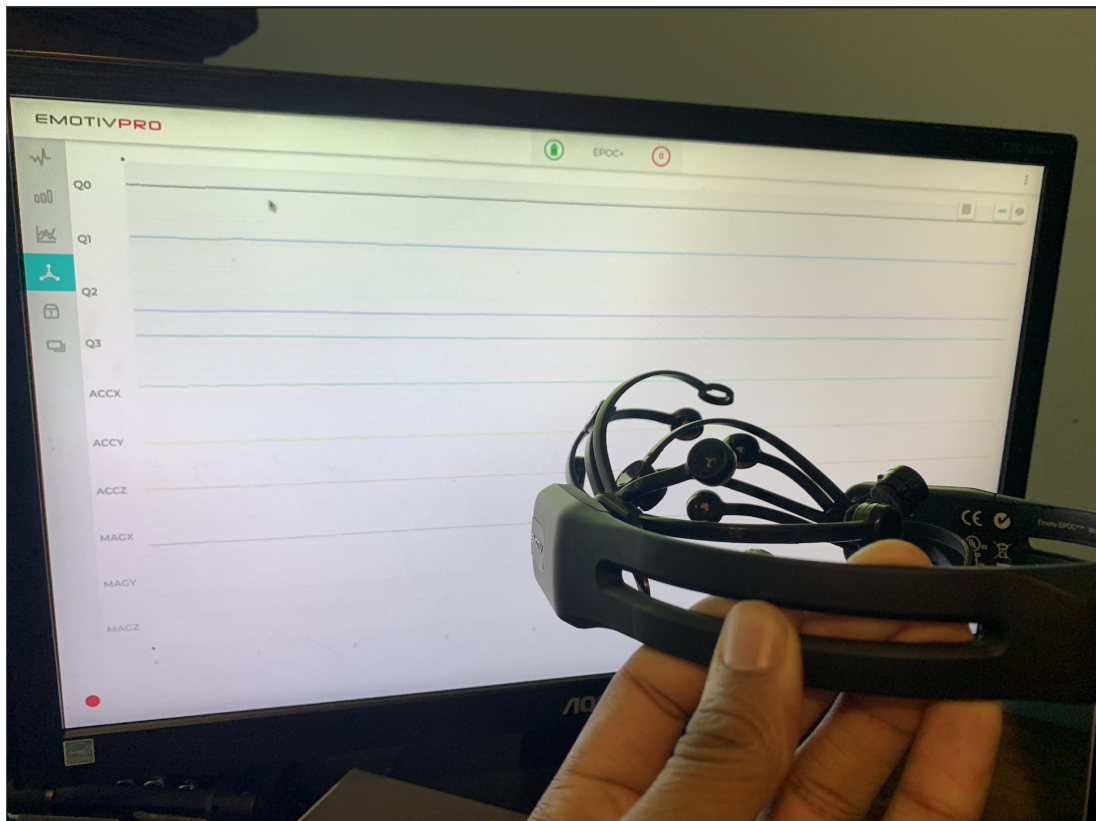


Figure 21: Apparatus from Experimentation

Data collection

Here is a snippet of the quaternion data streams from a CSV file.

A snippet of datastream as column vectors:

q0	q1	q2	q3
0.552257	0.424988	-0.625244	0.351379
0.552232	0.42511	-0.625183	0.351379
0.552328	0.425232	-0.624878	0.351624
0.552518	0.425232	-0.624573	0.351868
0.552779	0.425049	-0.624329	0.352112
0.552946	0.424438	-0.62439	0.352478
0.553064	0.423889	-0.624451	0.352844
0.553426	0.423523	-0.624207	0.353149
0.553698	0.423401	-0.62384	0.353516
0.553064	0.423889	-0.624451	0.352844
0.55397	0.423279	-0.623474	0.353882
0.553699	0.423096	-0.623596	0.354309
0.553492	0.422791	-0.623779	0.354675

Table 1: CSV file data for quaternion data

Validation

For validation, we used MatLab to plot the pitch, roll, and yaw angles over time. This MATLAB code extracts quaternion data from a CSV file, calculates the pitch roll and yaw angles as so. The x-axis (yaw) is limited from -180 to 180 degrees, the y axis (pitch) from -90 to 90 degrees, and the roll (z-axis) from -180 to 180 degrees. There are instances where *gimbal lock* causes the angles to wrap back around. For example, yaw is limited from -180 to 180 degrees, but when it the euler angle goes below -180 degrees it will wrap back up at 180 degrees in order to show 360 degrees of motion. Our criteria for validation is that angles produced in the EmotivPro software and Unity match the angles within MATLAB to validate our rotation angles are correct. Within

the first validation test we are going to make sure all rotations are correct without any gimbal lock. Then do another validation test and interpret a graph with gimbal lock.

Code Representation functionality of validation:

```
%%ECE-493-001
%VR learning environment with real time brain signal monitoring
% 2/21/2021
opengl software
quat = importdata('quattest.xlsx'); %insert filename of data
q0 = quat(:,1); % grabbing column vectors for quaternion data
q1 = quat(:,2); %column 2
q2 = quat(:,3); %column 3
q3 = quat(:,4); %column 3
%creating time vector
T = 1/64; % sampling rate of 64 hertz
t = (0:1261-1)*T; %creates time vector based off of EPOC+ sampling rate
% yaw, pitch, roll angle Vector
[yaw, pitch, roll] = quat2angle([q0 q1 q2 q3]);%converts quaternion data to yaw
pitch and roll
yawangle = (180/pi)*yaw;
pitchangle = (180/pi)*pitch;
rollangle = (180/pi)*roll;
% Rotation angle Vector
eul = quat2eul([q0 q1 q2 q3]); % This is another method to get the rotation
angles.
XAngle = eul(:,1)*(180/pi);
YAngle = eul(:,2)*(180/pi);
ZAngle = eul(:,3)*(180/pi);
%plotting data
figure;
plot(t,rollangle, t, pitchangle, t, yawangle);
title('Quaternion Angles Overtime');
xlabel('time(seconds)');
ylabel('Angle(degrees)');
legend('Roll Angle', 'Pitch Angle', 'Yaw Angle');
```

Validation test - No gimbal lock

This separate run shows us rotating the EPOC+ headset for 12 seconds.

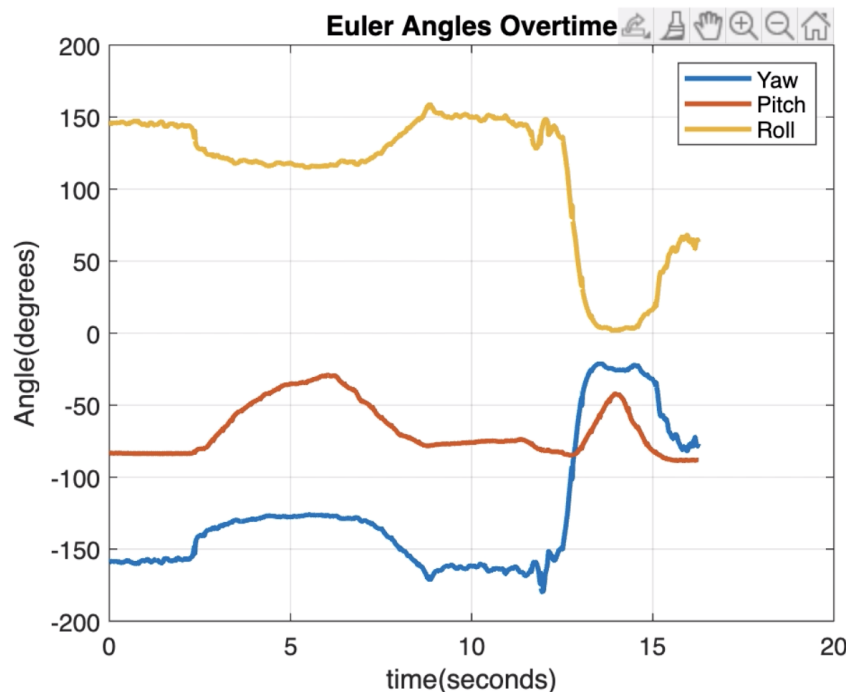


Figure 22: Pitch, Roll and Yaw Angles for validation test #2

Within this test we experienced *completely no gimbal lock* when rotating the headset. We were able to compare these rotation angles to how we were rotating the headset within our experiment. In this experiment we rotated upwards (pitch), rolled right (roll), and turned right (yaw). We rotated the headset around 75 degrees depicted above with the pitch angle in figure 22, meaning we rotated the headset upward toward the ceiling 75 degrees as we did in the experiment. Next, we rolled the headset to the right about 30 degrees back to the initial position which is what is shown from about 2 seconds to 8 seconds in figure 22. We then, again turned the headset right about 30 degrees and that is validated in the figure above. This experiment again, helps us validate our rotation data in order to use it for our application in unity.

Validation test #2 - With gimbal lock

This run is showing us rotating the headset in order to trigger gimbal lock across some rotation axes.

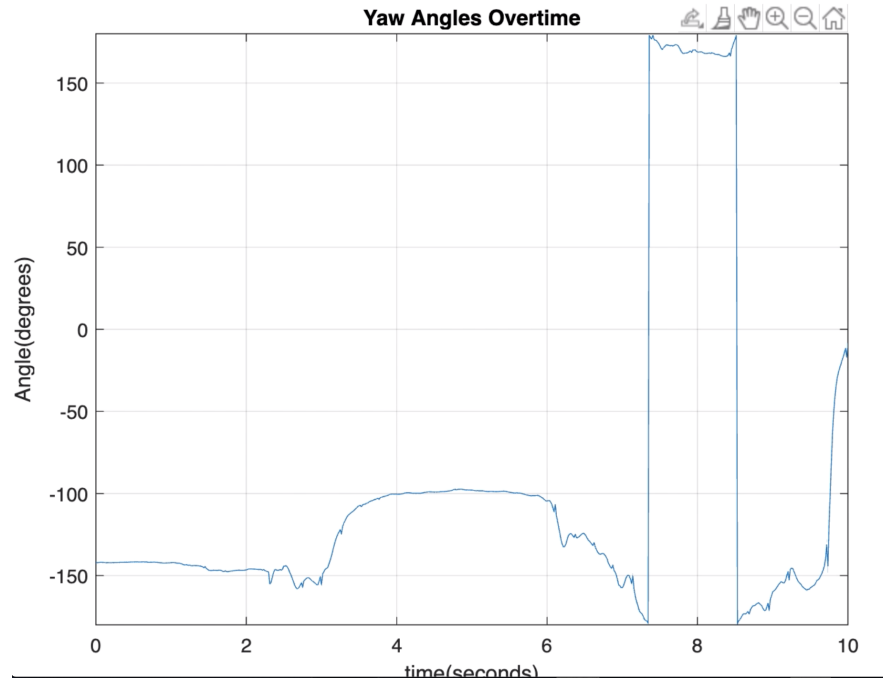


Figure 23: Yaw Angle with gimbal lock

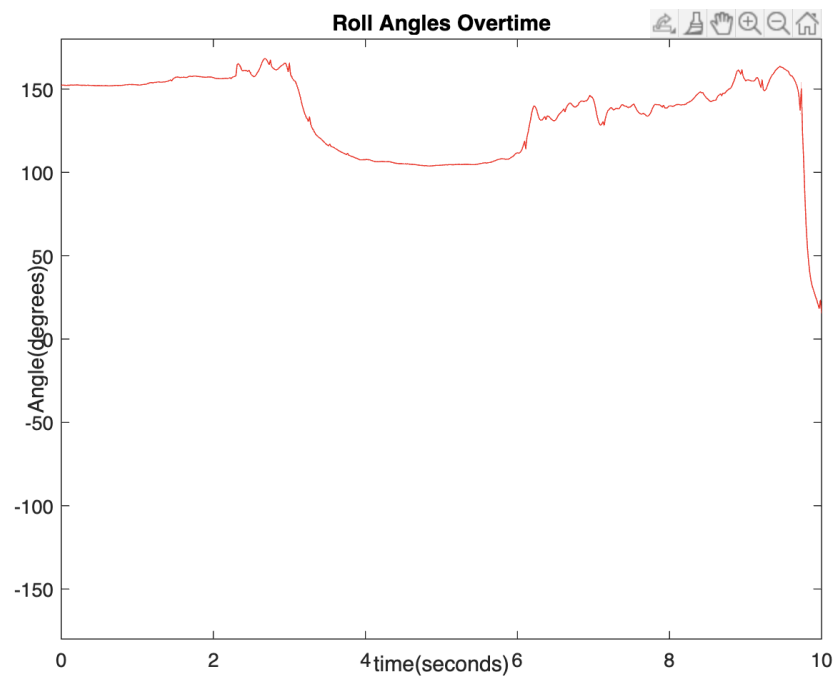


Figure 24: Roll Angle

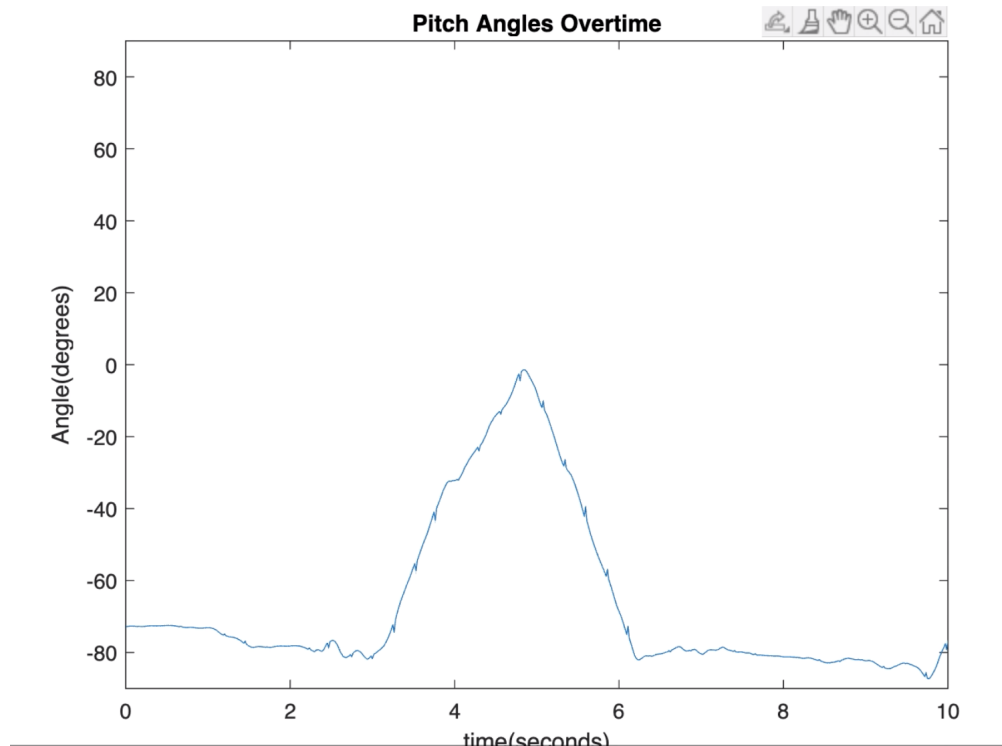


Figure 25: Pitch Angle

This validation test was similar to the previous except this time instead of turning right a second time we turned left, meaning that our Yaw angle would be less than -180 as shown in figure 23. Figures 24 & 25 showing pitch and roll and angles being similar to the previous validation experiment since the roll and upward movement was unchanged. Around 7.5s we see what gimbal lock causes as it makes the angle warp back around from -180 and puts the continuation of the angles starting at 180. This is due to a limitation of representing 3D angles using euler angles. With this gimbal lock example it helped us validate that we were correctly converting and representing the angles in 3D space.

Conclusion

The experiment with head tracking and validation was successful. We were able to compare the matlab angles to the in EmotivPro and the angles printed in unity with 100% certainty. This validation also helped us understand the issue of gimbal lock, in which we used methods within unity to help us solve these issues.

Test #2 - Performance metrics

Discussion

The objective of this experiment is to test performance metrics versus actual metrics. This consists of reading data at a sampling rate of 0.1 Hz in a range of $[-1,1]$ with any value below 0 meaning the performance metric is not active with above 0 being an active performance metric. When any of the performance metrics of Stress (FRU), Engagement (ENG), Interest (VAL), Excitement (EXC), Focus (FOC), and lastly Relaxation (MED) are read, we will verify by asking the user if they are feeling what the performance metric is outputting. E.g., the stress performance metric is outputting a 0.5 indicating a user is moderately stressed. This experiment is to help incorporate this data into our game and know what noise needs to be filtered out.

Pictures of Experiment (using emotiv pro app)

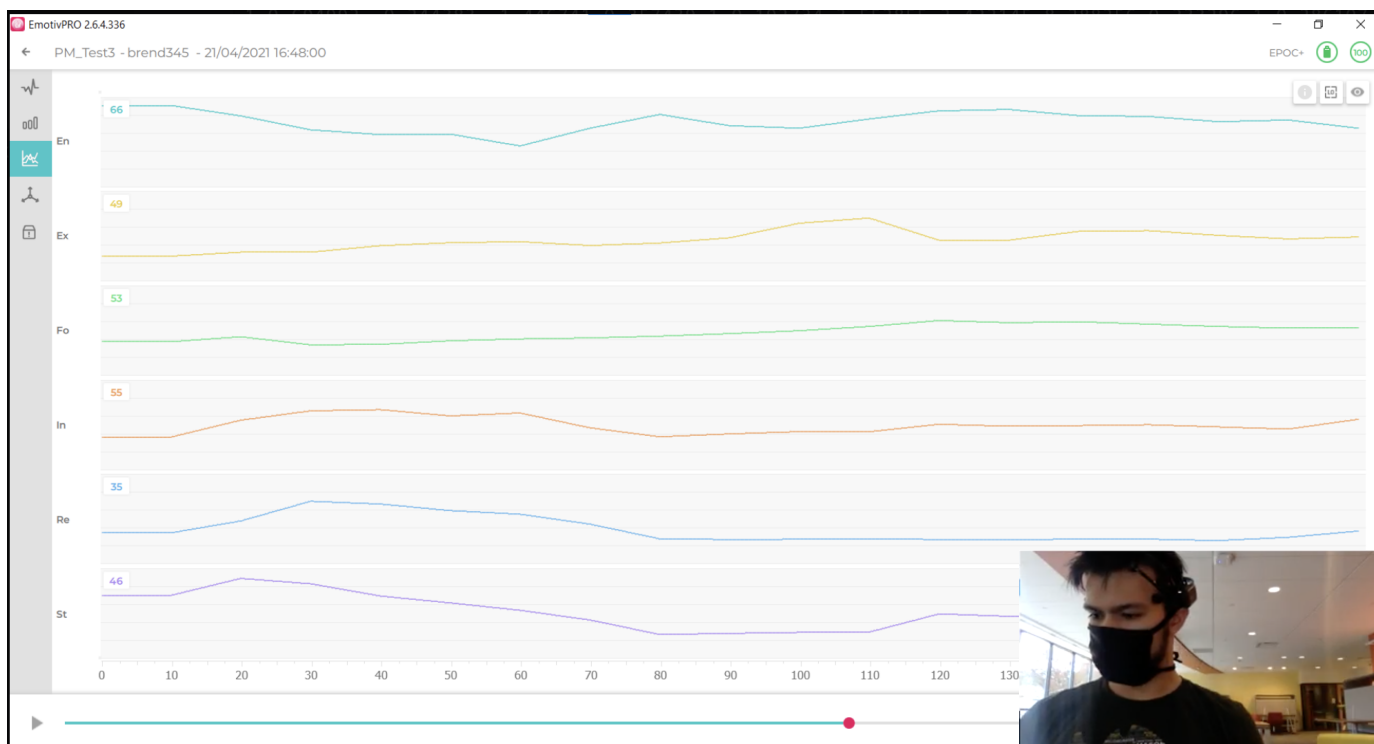


Figure 26: Experimental Apparatus

Data collection

Below is raw data pulled straight from our emotiv cortex app. As you can see in table 1, the performance metrics shown below have a range from 0 to 1 unscaled with “-1” indicating the performance metric is not active. The closer the value is to 0 the lower the performance metric is and the higher the number the more of that performance metric is active with a value of 1 being the max. For simplicity, for user-specified we scaled these values by multiplying by 100 for a range of [0 to 100] for actual displayed data outside of experimentation.

<u>PM.Engagemen</u> <u>t.Scaled</u>	<u>PM.Excitemen</u> <u>t.Scaled</u>	<u>PM.Stress.</u> <u>Scaled</u>	<u>PM.Relaxation</u> <u>.Scaled</u>	<u>PM.Interest.</u> <u>Scaled</u>	<u>PM.Focus.</u> <u>Scaled</u>
<u>0.66</u>	<u>0.40</u>	<u>0.17</u>	<u>0.33</u>	<u>0.63</u>	<u>0.21</u>
<u>0.69</u>	<u>0.19</u>	<u>0.28</u>	<u>0.27</u>	<u>0.53</u>	<u>0.33</u>
<u>0.70</u>	<u>0.12</u>	<u>0.43</u>	<u>0.20</u>	<u>0.52</u>	<u>0.40</u>
<u>0.75</u>	<u>0.05</u>	<u>0.56</u>	<u>0.23</u>	<u>0.58</u>	<u>0.37</u>
<u>0.35</u>	<u>0.01</u>	<u>0.38</u>	<u>0.11</u>	<u>0.26</u>	<u>0.21</u>
<u>0.80</u>	<u>0.18</u>	<u>0.88</u>	<u>0.41</u>	<u>0.68</u>	<u>0.41</u>
<u>0.81</u>	<u>0.22</u>	<u>0.91</u>	<u>0.34</u>	<u>0.62</u>	<u>0.46</u>
<u>0.73</u>	<u>0.12</u>	<u>0.57</u>	<u>0.24</u>	<u>0.55</u>	<u>0.40</u>
<u>0.70</u>	<u>0.11</u>	<u>0.91</u>	<u>0.42</u>	<u>0.63</u>	<u>0.43</u>
<u>0.50</u>	<u>0.24</u>	<u>0.92</u>	<u>0.69</u>	<u>0.74</u>	<u>0.32</u>
<u>0.30</u>	<u>0.11</u>	<u>0.37</u>	<u>0.15</u>	<u>0.27</u>	<u>0.21</u>
<u>0.15</u>	<u>0.31</u>	<u>-1</u>	<u>0.05</u>	<u>0.68</u>	<u>-1</u>
<u>0.77</u>	<u>0.62</u>	<u>0.37</u>	<u>0.26</u>	<u>0.64</u>	<u>0.30</u>
<u>0.58</u>	<u>0.45</u>	<u>0.96</u>	<u>0.57</u>	<u>0.76</u>	<u>0.54</u>
<u>0.40</u>	<u>0.31</u>	<u>0.88</u>	<u>0.69</u>	<u>0.67</u>	<u>0.47</u>
<u>0.470</u>	<u>0.10</u>	<u>0.53</u>	<u>0.56</u>	<u>0.53</u>	<u>0.36</u>
<u>0.287</u>	<u>0.12</u>	<u>-1</u>	<u>0.12</u>	<u>0.24</u>	<u>-1</u>

Table 2: CSV Data for Excitement (EXC)

Validation

Below we have our validations for all 6 performance metrics all plotted for at least 3 minutes. All tests standard deviations and averages were calculated for all performance metrics.

Code Representation functionality of validation:

```
clear
opengl software
pm = importdata('performance metric.csv'); %insert filename of data
engage = pm.data(:,1)*100; % grabbing column vectors for pm data
excite= pm.data(:,2)*100; %column 2
stress = pm.data(:,3)*100; %column 3
relax = pm.data(:,4)*100; %column 4
interest = pm.data(:,5)*100; %column 5
focus= pm.data(:,6)*100; %column 6
T = 1/0.1;
t = (0:17-1)*T;
figure;
grid on
plot(t, engage)
legend('Engagement');
plot(t,excite)
plot(t, stress, t, relax,'LineWidth',2)
plot(t, engage, 'LineWidth',2)
plot(t, interest)
plot(t, focus)
xlabel('time(seconds)');
ylabel('Metric');
```


Tests

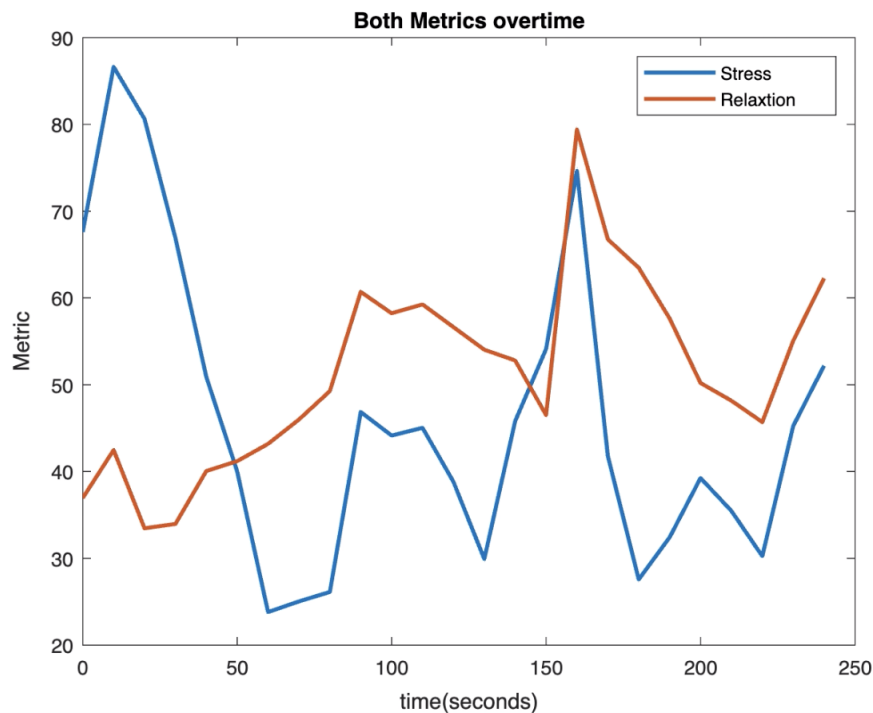


Figure 27: Stress and Relaxation graphed overtime

During this validation test, we were expecting stress (FRU) to go down and for relaxation (MED) to go up by asking the user to do deep breathing exercises. At the start we told the user to do deep breathing for 120 seconds. Shown in figure 27, after about 50 seconds we began to see the data trend in the right direction. With the average of stress after 50 seconds being 38.75 and the average of relaxation being 59.4. Although we did see a spike in stress around 160 seconds as sometimes there are outliers output from the data stream. In conclusion, our theoretical predictions matched our experiment validating our experiment.

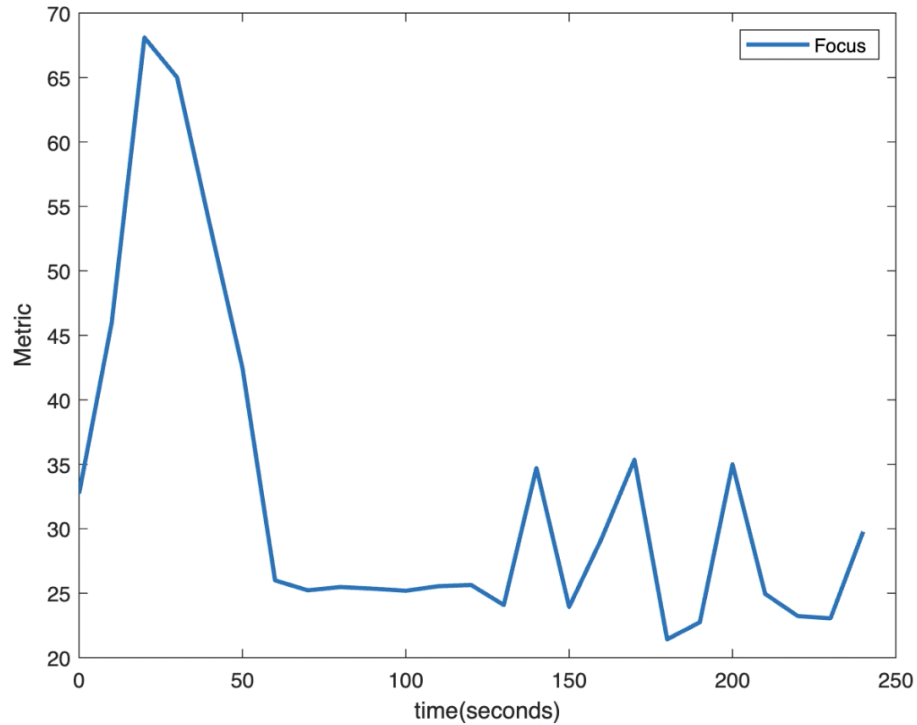


Figure 28: Focus plotted overtime

Next we wanted to validate our focus (FOC) performance metric. We assumed that if we could get the user extremely focused on something, then distract the user we should see a dramatic drop in focus. We made the user focused at about 20-30 seconds as in shown figure 28. After distracting the user the focus metric dropped considerably. In conclusion, our theoretical predictions matched our experiment validating our experiment.

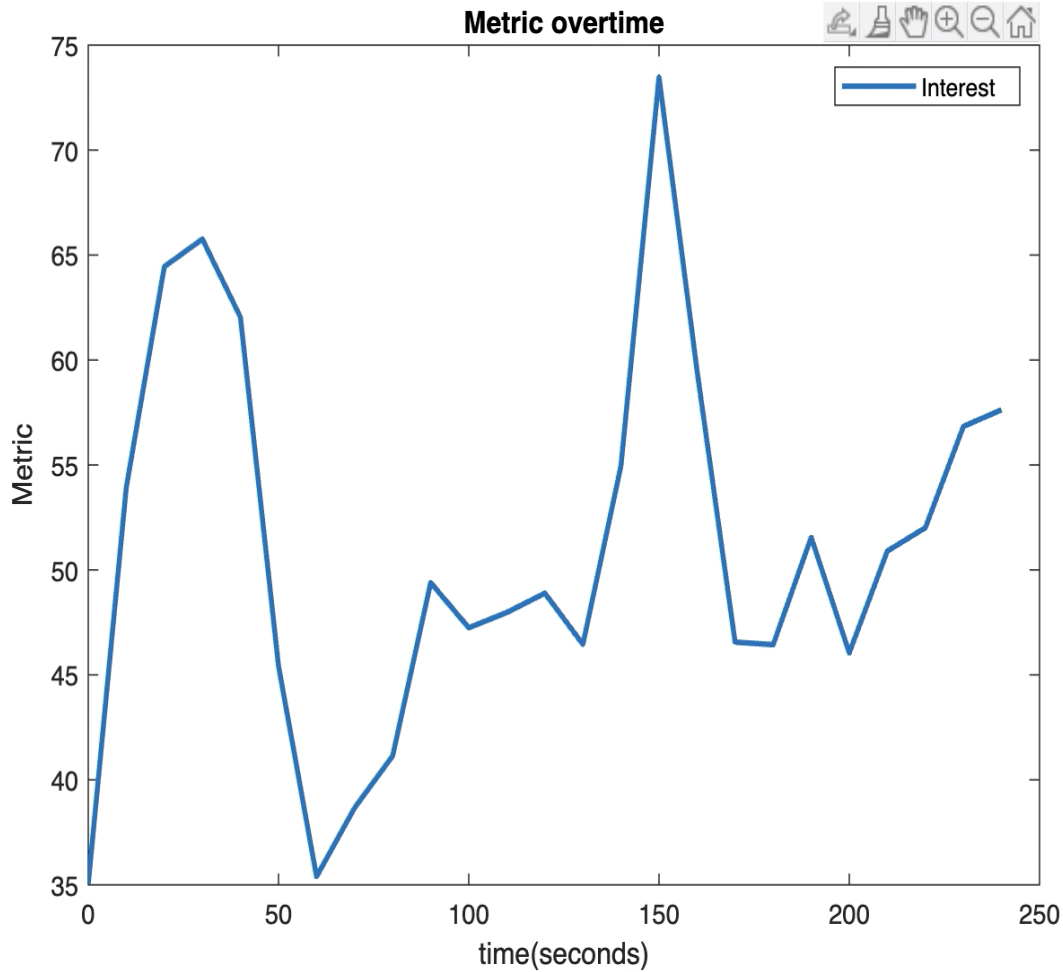


Figure 29: Interest plotted overtime

Within this test we were looking to validate our interest (VAL) metric. Interest is a hard metric to validate because users' interests vary. So we finally found something that triggered the users interest twice by playing a youtube video containing their favorite creator. In figure 29, around 55 seconds and 150 seconds respectively we saw the users interest peak when watching the videos. To verify these spikes were true, we had the user look away for 10 seconds around 50 seconds in order to see if the metric went lower. In conclusion, our theoretical predictions matched our experiment validating our experiment.

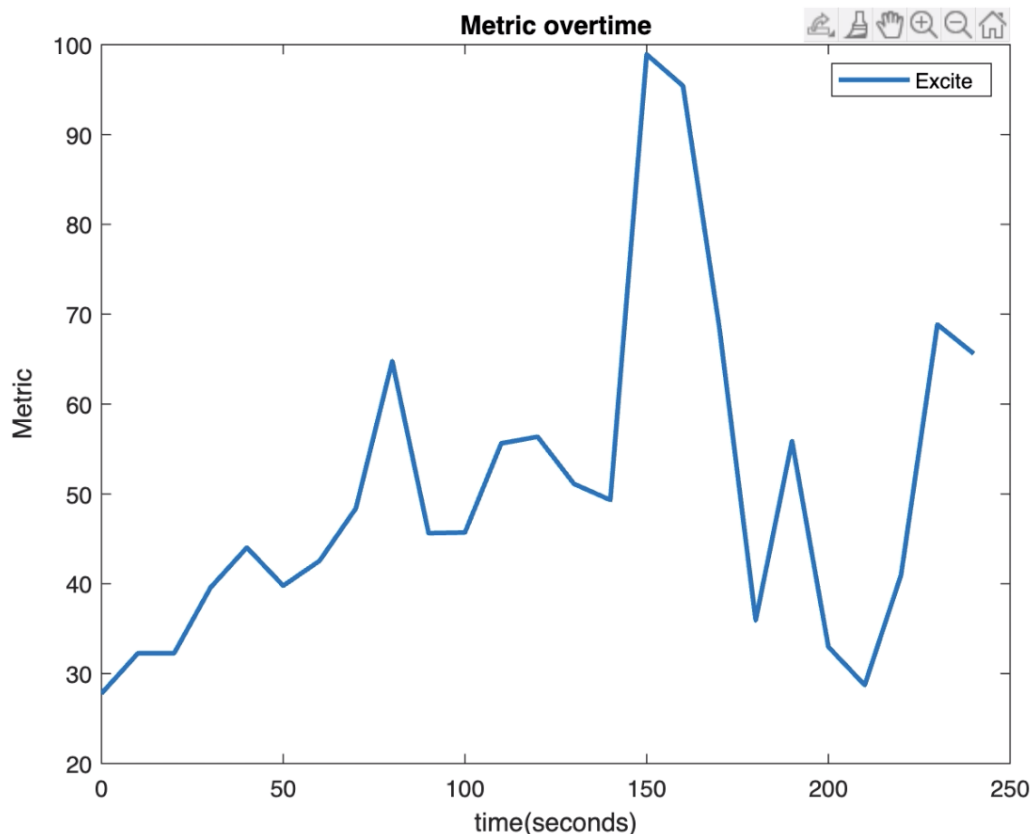


Figure 30: Engagement plotted overtime

Lastly, we wanted to test the engagement (ENG) metric. We did this easily by just telling the user to play our game for about 2 minutes and 30 seconds. We saw an upward trend in excitement the longer the user played the game as shown in figure 30. After we told the user to stop playing there was a considerable drop in the engagement metric for these tests. In conclusion, our theoretical predictions matched our experiment validating our experiment.

Conclusion

The criteria for validation of performance metrics consists of asking the user if what they are experiencing is reflected by what the EPOC+ headset performance metric is indicating with a success rate in comparison of at least 90%. We were able to verify our theoretical results while having a 100% success rate in our brain signal testing. We also calculate the average and standard deviation of our data. Our data was within 3 standard deviations of the mean also helping validate our data as shown below in table 3.

Statistic:	Engagement	Excitement	Stress	Relaxation	Interest	Focus
Average	74.3	50.6	46.1	32.6	51.1	51.338748
Standard Dev.	0.105	0.184	0.174	0.129	0.09	0.109

Table 3: Average and standard deviation of metrics

(Next Page)

Test #3 - Unity Gameplay

Discussion

This experiment consists of testing the implementation of Leap Motion and Emotiv SDKs, behavior of created C# scripts, and behavior of Unity objects with our learning modules. The Leap Motion SDK should map the users hands into the virtual environment and allow for interaction with designated objects. This is done by changing component values in the Unity inspector to create a mesh that the hand models can interact with. The Emotiv SDK requires two different tests. First, we needed to ensure that the values that are being read for the Cortex API correspond to the values obtained from both previously mentioned experiments. This ensures correctness of data before attempting to map values for scene interactions. After ensuring correct data, we need to link motion data from the Emotiv to camera rotation in Unity. The data taken from the Leap Motion and the Emotiv are used to control the C# scripts. We must test that all components within our environment behave as expected with and without interactions from other objects.

Pictures of Experimentation

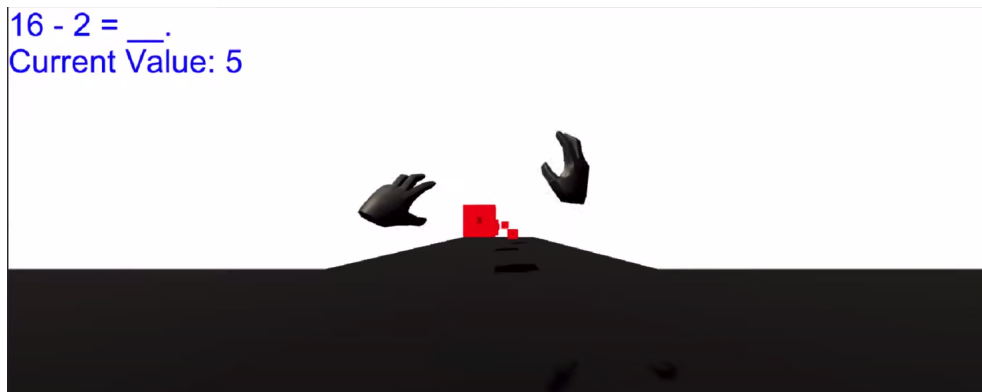


Figure 31: Screenshot of unity gameplay

Validation

Due to the nature of this experiment, there will be no significant numerical data obtained. For the Leap Motion, we tested the hand mapping in example and the group created Unity scenes. The hand models were correctly mapped in both, properly displaying the correct hand and position in the environment. We also tested the hand models on Unity objects containing an interaction mesh and objects without it. The user can grasp and move items containing the mesh, while hand models pass through objects not containing this mesh. With these two constraints working correctly, we can validate that the leap motions behave as we expected within the Unity environment.

The first Emotiv Test requires comparison of data from the Emotiv Pro application and data from the Cortex API. In order to test this, these two things must be run concurrently and values should change simultaneously. This test is pretty straight forward, demonstrating correctness upon matching data. Testing the implementation of motion data requires several conversions. Real world quaternion values are different from Unity's. Because of this, we needed to write a script that converts between these two. During testing we experienced many issues due to gimbal lock, causing the camera to behave in unexpected ways. After researching work arounds, we determined that there was no straightforward method to work around this using transforms for camera rotations. To combat this, we narrowed our head tracking to one axis of rotation. After doing this, we still experienced some form of gimbal lock until we properly calibrated the headset. After experimentation, we were successful in mapping head movements to camera rotation without the user experiencing erratic camera movements.

The last part of this experiment centered on behaviors of C# scripts within the virtual environment. First, we needed to ensure that our scenes were displayed in the correct order and executed until completion. This required building the project and testing it from beginning to end. Our startup scenes displayed first, being followed by the classroom for module selection. After selecting the correct module, we were able to access our learning module, in which the majority of our created scripts are found. Our project contains a game manager and spawning script which provide that majority of functionality of the module. In order to determine if the spawning script worked correctly, block prefabs must spawn from designated locations. The spawned blocks must maintain a specified movement speed and both addition and subtraction prefabs must be created. Our module correctly displayed this behavior, proving proper implementation. The game manager script needed to properly generate equations, display the GUI for module status, and allow for repetition of levels. Upon loading in, the GUI properly displays the equation, the correct count and the completed count. Interacting with the prefabs

updates the equation, moving to the next repetition of scene on an end condition. After the designated number of repetitions were completed the application closed as intended.

Conclusion

Implementation of Leap Motion behaved exactly as expected, showing proper mapping and implementation with specified Unity objects. The Cortex API displayed the correct data, allowing for matching values with the Emotiv App. This ensures that the data imported into Unity is the same data you would get using Emotiv's software. After making a few changes to our head tracking script, we were properly able to map leap motion movement to camera rotation. Although not as in depth as we planned, gimbal lock prevented smooth Unity transformations. Lastly, we validated that our created scripts work as intended. Our game runs from start to end, allowing for intended behavior throughout the duration of execution. Testing each component of our modules separately and together prove that our project properly implements our intended actions.

(Next Page)

Test #4 - Teacher Hub

Discussion

The purpose of this experiment is to test variable mapping on two different Unity instances. This testing will be done on the local host using the IP address 127.0.0.1 as the main method for data transfer. Network variables must be created to move data, allowing write permissions from both sides of the data transmission to properly modify variables. Also, referencing network variables on one Unity instance requires properly locating the script containing the necessary variables and mapping them to unique network variables on the other. Using one Unity instance to modify the settings of another requires multiple functions to be linked together because a common player prefab is required for network transmission. Proper implementation of the teacher hub will allow the teacher to send commands to modify the student's module settings, while allowing the performance metric data from the student side to be accessed by the teacher.

Validation

Just like experiment 3, this test will not require obtaining any new numerical data. This test will be considered in multiple parts. The first experiment will involve sending data from the teacher hub to our learning module. For each student in the teacher hub there will be buttons that can modify the movement speed of the blocks in the student's module. To properly implement these buttons, each button must modify a speed variable on the teacher side and then send the updated value to the student. This value will be read by a function on the student's side, updating the value in their settings. After linking the buttons, we were able to increase the speed of the block movement to a bounded 2.5 and decrease the speed of the block movement to .5. Shown in figure 32 below shows the buttons referenced and experimented with.

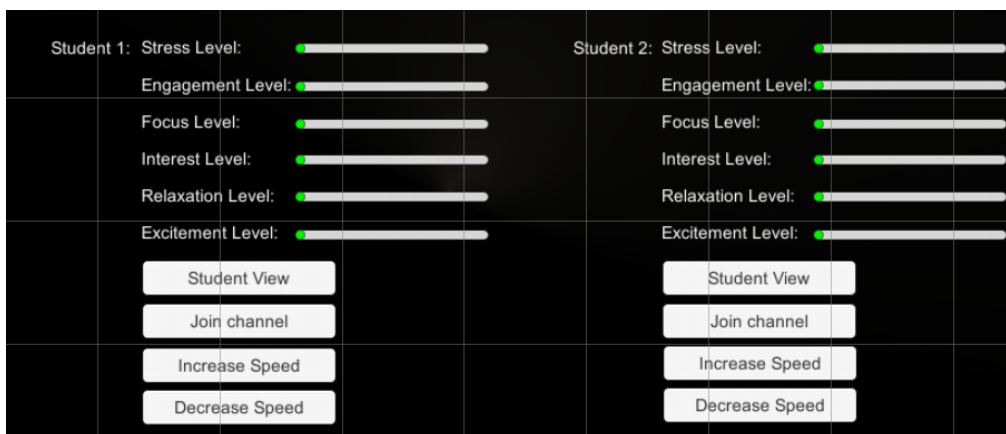


Figure 32: Testing of teacher hub

The second experiment tests receiving performance value metrics from the student's module and displaying them for the teacher. We first started by using static variables to send across the network because the values of the performance metrics update slowly in Unity. The teacher hub contains sliders for performance metrics, increasing the amount they are filled as the performance metrics grow closer to 100. For the initial tests we set each metric data sent to .5, intending each slider to fill halfway. This was not working for the longest time, but modifying the network settings using trial and error eventually linked the two instances together. After we were able to obtain the static variables, we tested the code to send the performance metric data. We compared the values from the Cortex data stream to the values being displayed on the teacher hub to determine if the values were consistent. Doing this resulted in matching data, validating the implementation of the teacher hub.

Conclusion

From this test we were able to determine the data being sent and received by the teacher hub were aligned with the intended values. This section of the testing took a long time because of the network requirements. The scripts on both the teacher hub and student side needed to be the same to ensure proper connection. We also had errors with inconsistency in the network settings across unity instances. After properly configuring this data we had full access to all network variables. The values of the blocks speed change almost instantaneously and the performance metric data updates on the teacher side at the pm refresh rate from the Cortex API. Overall, we were able to validate that the behavior of our teacher hub behaves as intended.

(Next Page)

Results



Figure 33: Final Project Setup

Our system design successfully incorporates the Epoc+, Leap motion controller, Unity game engine, and the virtual reality headset. The outcome came with many redesigns of the project, hours of debugging code, and validating EEG signals.

With the Epoc+, our system had to slowly process thousands of EEG signals per second along with computing the performance metrics of the user. This led to the system slowing down dramatically when running in parallel with the Unity game engine along with storing and exporting the signals as buffers to the teacher hub. Our work around for this problem was to average in the data and slow down the rates being stored on the teacher hub's computer. With that, the data still lead to accurate results while the performance of the system not degrading nearly as much as it would have.

Our first design of the module was our group's first scene ever created in Unity. Knowing that, the scene had to be scrapped and our second module would build off the foundations of the first design. After many trials, this led to our final design of the module iteration names "Equation Saber," modeled after the game "beat saber" with designs and functionalities to incorporate

learning measures. The aesthetics of the module were changed in combination with the module redesign, as it needed to be visually pleasing while also not distracting during a lesson. We decided on sending the objects towards the students as operators of the equations for greater visual appeals than a box and greater focus towards the actual purpose of the module.

The system also allows for the expansion of modules to be easily incorporated into the design. Any educator or developer can create and add a jump scene button that allows the student to connect to it. Once connected, that scene will have full access to all of the Epoc+ data streams, leap motion controller stream, and the teacher hub connection. We expect that the expansion of modules will allow for higher educational lessons to be incorporated into the system, making this a universally used environment between schools.

The teacher hub currently works as a local host connection to the student's device. This allows for the connection between a teacher's computer to the student's activity during any module. For expansion of the system, we expect to create and host a server that the teacher hub can run on, connecting to the student's device from any network rather than the same computer.

The core of our system is the connection between APIs, SDKs, and the scripts that run the module. Using the APIs, our system has a direct connection between the physical components of our build with the software that runs it. This is critical for the Epoc+ where the EmotivPro software receives data and the only functionality without the API is to export the data and then read the data once written. With SDKs, our development of the leap motion controller and the Epoc+ did not have to start from scratch. We could import the core functionality of the product and build off of the ever so many functions that the SDK provides. This gave us ample room to design the functionality of modules interacting with the SDKs as we did not have to design the SDKs.

The experiments and validation was a large part of the final product of our project. The experiments to test functionality of headtracking, performance metrics, gameplay, and the teacher all served valuable lessons of what would and wouldn't work in our project. Within headtracking testing, we came across maybe issues with mapping the head tracking in 3D space. Gimbal lock was a huge problem for us as it limited our range of motion although we were able to find workarounds to fix it for our use using unities scripting API. Performance metric experimentation was a complete success as we were able to validate all metrics and implement it into our Teacher hub. The only issue was the speed performance metrics were updated in the teacher hub. With a refresh rate of 0.1 Hz, this is not ideal, but it's the fastest speed we were able to obtain. Within gameplay testing, everything meshed well with all of our components and we

were able to figure out the limitations to our game and its functionality. Within our teacher hub testing had very few errors that were quickly debugged. Teacher hub testing was a success by verifying our performance metrics data stream was sent correctly. In conclusion, all of our testing was an overall success.

Overall, our system is only able to function after the testing and validation of data, connecting of software designs, and hardware processing. We designed our system to include many separate components that were not specifically created to be used in unison. This made it an incredibly challenging task but well worth it. Our system is fully capable of expanding into education beyond our reach and is only limited by the creativity of the educator designing the module.

A YouTube link to our final design: <https://www.youtube.com/watch?v=QiAAxHF4gEo>

(Next Page)

Other issues

Reason for the project

This project aims to provide a way to estimate cognitive workload in a virtual reality environment using EEG based analysis for learning. The purpose of the project is to provide a personalized virtual learning environment to students with marginalized education opportunities due to their environment, this project's purpose is reinforced during the Covid-19 pandemic where there has been a distinctive global rise in online learning. Although remote learning may present a barrier for interactivity and instructor control that would otherwise be present in a traditional classroom environment, this project insists that a virtual learning environment grants the opportunity for even further personalized lessons by allowing analysis into student's retention rates by monitoring brain signal data [10].

Potential use of the project

Not only can this project have significant use in remote-learning situations and homeschooling, but the scope of this project can also be extended within in-person classrooms. Personalized learning has the potential to globally change opportunities for students living in low-educational access environments. The end-user for this project is not only students in classes ranging from kindergarten to higher-level education, but also in various other training situations such as driving institutes, military and police, and also private tutors and parents. This can also benefit students needing special accommodations for testing and learning as well as medical professionals interested in neurofeedback for diagnosing cases such as Alzheimers and attention deficit hyperactivity disorder (ADHD). Neurorehabilitation is a promising use for this project. Research done by Duke University used VR and brain-computer interfaces to treat physically impaired patients. The 12-month study with 8 patients concluded that VR can help aid in restoring ability with patients who suffer from a chronic spinal cord injury [11].

Cost figures

The money spent for the complete build of the system, referenced in the funds spent section, was \$141.86, not including the monthly EmotivPro fee. Including the Epoc+, discontinued and rebranded as Epoc x, the cost of the build is increased to \$991.86. With that, the base price of the system is \$991.86 with a recurring monthly fee of \$29.99 for the student version of the EmotivPro software. Incorporating the engineering hours spent (Roughly 1150 hours) at a rate of \$33 an hour (The national average of entry-level electrical and computer engineers), the product

must make enough to cover \$37,950 in hours worked, leading to a product value of \$180.00 without the Epoc and \$1030.00 with the Epoc (Assuming a figure of 1000 products sold) [12].

We plan to sell our product as a service without the Epoc, allowing our primary audience to buy Epoc's as a separate entity of the product. This allows us to value our product at \$18 a month, netting a 20% operating margin by the end of year 1, and a 100% operating margin years after assuming no maintenance is required. However, including maintenance, a development team of 3 engineers will operate and maintain the product at the national average rate of entry-level electrical and computer engineers, \$33 an hour, leading to an extra \$17 a month with the 1000 figure. Including maintenance of the product, the value is increased to \$38 a month with a 20% profit by the end of year 1, and a 55% margin afterwards.

Design Alternatives

There were several components we considered implementing, but we found that the used components fit our needs more closely. As discussed above, we were considering using the Muse headset for brainwave monitoring. Although this method does provide less insight into the physiological state, the price is more friendly. The Muse costs \$250.00, lowering the cost of the headset by \$600.00. Using the muse would also require changing the monitoring method in Unity, changing the necessary SDKs if there is one currently available for the Muse. Another implementation considered was the creation of a custom controller. This would cost \$10 for a Raspberry Pi Zero, \$8 for an ADXL345, and the cost for 3D printing a casing for the headset. This is significantly cheaper than the Leap Motion; however, implementation and maintenance on a custom controller would be a lot more difficult. Lastly, we had two backup plans for the implementation of the virtual environment. The first idea was the creation of a custom 3D printed headset. Headset lenses range from \$3 to upwards of \$30. The lower the quality of the lens, the less enjoyable the VR experience. Straps for VR goggles are also inexpensive, allowing for a low total cost for the headset. After considering our options, we believed that sacrificing quality for a low costing NeuTab headset was not worth it. Lastly, we considered using an Oculus VR headset. This headset is a minimum of \$300, making its use in the classroom environment expensive. This would provide a higher-level experience for the user, but for our purpose, spending the money on the Oculus vs. using the NeuTab wasn't worth the tradeoff.

Maintainability/Maintenance

One thing we would need to do to maintain our design is to update files for the Cortex API connection as the version for API changes. Other updates include adding new Unity modules to

meet the specific needs of the customer, being able to expand the target audience of the product, and updating unity versions. Important firmware updates need to be administered for the Epoc+ as the Cortex API is updated. New VR headsets and pieces of headsets can easily be 3D printed in case of any breaks or malfunctions of the design.

Retirement, Replacement, and Disposal of Project

Components such as the Epoc+ can be dismantled. The gold plated contacts can be recycled and reused while the battery of the Epoc+ should be taken to permitted treatment facilities for recycling or disposal. Other electrical components such as wires should be taken to nonprofits that recycle electronic material. Lastly, all plastics and/or glass lenses can be normally recycled at a general recycling plant. For replacement, old gold-plated contacts can be readministered and plastic can be moulded for the design of newer modelled products.

(Next Page)

Administrative part

Discuss project progress

The start of the project was mostly figuring out how to implement our multiple different components together. There were several different APIs that we had to read the documentation for and implement into our unity game. We originally set out to create a VR game controlled by brain signals but decided we wanted to do something a little more applicable to the current times. This is when we came up with the idea for a virtual learning environment.

With the rise of COVID, we all decided that it would be a more relatable project if we were to create something that would enhance distance learning. We found that we could use brain signals to create an environment that would allow teachers to monitor their students' attention span and responsiveness to their teachings. The first semester, 492, was mostly spent creating a very basic module and ensuring correct implementation of our different components, including the EPOC+ and the leap motion.

When we moved into 493, our focus was on refining our project and getting it all to work cleanly together.

Did you complete all tasks successfully?

We were able to implement most of our tasks successfully. One thing that we weren't able to finish in time and ended up having to drop was to have our teacher and students connect from separate servers. This means that the student and the teacher would have to be on the same network, but the cost of renting a fully dedicated server would be significantly too high for this project., and there weren't many other elegant solutions that didn't inhibit the ability for the rest of our project to work correctly.

Did you have to change the design, tasks, schedule? What were the changes and why?

We did have to change the design slightly throughout the semester. We originally started out creating a new 3D modelled headset for our project to be housed in but ended up scrapping that and just using the neutab instead. It had the full functionality that we were looking for in our 3D headset, and we felt that we could have a more well-rounded project if we focused more on the virtual modules rather than creating something that we had cheap and easy access to that would give the same result.

Another early design change we implemented was to change over from raspberry pi controlled accelerometers and handheld controllers to leap motion. Our original plan was to use two small handheld controllers with accelerometers and basic buttons that would be used to interact with the modules. We found that it was much more elegant and refined if we used the leap motion API. It allows the user to have full real-time models of their hands in the modules and it makes them much more interactive.

We were able to stick pretty well to our schedule set out at the beginning of the semester. We had a fully working, yet somewhat basic model to display in late March. Then we began to make a more visually pleasing and intellectually challenging module. The validation part of our project began a little late, but we were still able to get it done on time.

Any extra (not-planned) activities you had to carry out?

An unplanned activity was having to redesign our learning module after feedback from our Faculty Supervisor. No other unexpected activities came up apart from this instance.

Funds spent + discussion

The majority of our funds were spent on the EMOTIV pro software subscription. This subscription is necessary to get readings from the EMOTIV EPOC+ in real-time. This means that we had to maintain this subscription for a long amount of time to do our experimentation with the brain signals. The rest of our budget was spent on small things such as the LCD screen, resin for 3D printing, and USB/HDMI connections. The budget would have been significantly higher, but the EPOC+ and leap motion were given to us by our faculty supervisor Nathalia Peixoto.

Budget - \$500	Links	Price
Emotive pro software subscription - \$30/mo.	Emotiv	\$193.95
LCD Screen	Amazon	\$46.99
Resin for 3D printing	Amazon	\$50
USB-C hub	USBCHUB	24.99
USB C extension	Extension	13.99
Short HDMI cable	HDMI	5.89
	Total	\$335.81

Table 4: Budget Breakdown

Man-hours devoted to the project and Discussion

Team Member:	Fall 2020 (hours):	Spring 2021 (hours):	Totals:
Brendan Jenkins	105	135	240
Jacob Mitchell	100	130	230
Yumna Rizvi	95	115	210
Zayne Frabutt	102	136	238
Ethan Li	105	138	243
Team Totals(per sem):	507	647	1154

Table 5: Man hours for all members

All man-hours are rough calculations based strictly on individuals project working times, meetings, documentation, and discussions. These hours were logged in google sheets for an accurate representation of working times. All individuals contributed over 200 man-hours for a team total of over ~1100 man-hours worked.

(Next Page)

Lessons learned

General knowledge/Research:

- ❖ Learning up to date knowledge about brain signals
 - Learning about various brain signal waves such as gamma, beta, alpha, delta, and theta, which all have different meanings in terms of what a person is feeling which is classified as emotional metrics when factoring in different channels.
- ❖ Research on students during online learning
 - During our project, we were able to research issues students were having during online learning to help aid which emotional metrics we should keep track of when the user was in the learning environment.

Technical Skills:

- ❖ JavaScript Object Notation for the WebSocket
 - Web Sockets are true concurrency and optimization of performance, resulting in more responsive and rich web applications. We learned how to use websockets in order to develop our Cortex App which used websockets in order to send data packets to our Unity game.
- ❖ Learning C# scripts
 - Learning new C# syntax in order to help program different functionality of our game.
- ❖ Learning how to use Application Programming Interfaces (API's)
 - Creation and subscription to Emotiv Cortex to obtain data from Emotiv SDK
 - Accessing Vivox API to create voice channels between teacher hub and learning module
- ❖ Learning to validate data and experiments to assist drawing conclusions about
 - Validation of data was important for us as we needed to know what the data was actually representing as well as figuring out the noise thresholds which would interfere with our actual data.
- ❖ Reading reference code and learning how to implement it for our own purposes
 - Basic technical skill that is seen very often in industry especially geared toward programming and/or microprocessor programming.

Managerial Skills (for every individual):

- ❖ Manage time and tasks
 - Every group member was able to be given tasks in which they were responsible for. Every member was responsible for completing their tasks by a specific deadline to ensure project success.

Teaming Experience:

- ❖ Group work online
 - From a group consensus we agreed that working on a large scale project during online learning was more difficult than expected. More miniscule issues arose that required some sort of mitigation as it was hard to find places to do group work in person that was suitable for everyone.
- ❖ Consistent meeting times
 - Meeting 2 or more times a week helped us resolve issues of individuals going off task and making sure everyone was keeping up to date in completing everything necessary to stay on track with our completed schedule.

(Next Page)

References

- [1] K. Mukhtar *et al*, "Advantages, limitations and recommendations for online learning during COVID-19 pandemic era," *Pak. J. of Med. Sci. Quarterly*, vol. 36, 2020. DOI: [10.12669/pjms.36.COVID19-S4.2785](https://doi.org/10.12669/pjms.36.COVID19-S4.2785)
- [2] P. A. Abhang, B. W. Gawali, S. C. Mehrotra, "Chapter 2 - Technological basics of EEG recording and operation of apparatus," in *Introduction to EEG- and Speech-Based Emotion Recognition*, P. A. Abhang, B. W. Gawali, and S. C. Mehrotra, Eds. *Amsterdam: Academic Press*, 2016, pp. 19-50. DOI: <https://doi.org/10.1016/B978-0-12-804490-2.00002-6>
- [3] Z. Scavotto, N. Peixoto, "Video games and brain wave frequencies," 2020. [Online]. Available: <http://gamingwaves.onmason.com/>
- [4] Brainworks, "What Are Brain Waves?," 2020. [Online] Available: <https://brainworksneurotherapy.com/what-are-brainwaves>
- [5] Emotiv, "How Does an EEG Work?," 2020. [Online] Available: <https://www.emotiv.com/eeg-guide/>
- [6] S. Pathirana, M'd Gapar M'd Johar, D. Asirvatham, "Applicability of multi-agent systems for electroencephalographic data classification," *Procedia Comp. Sci.*, vol. 152, pg. 36-43, 2019. [10.1016/j.procs.2019.05.024](https://doi.org/10.1016/j.procs.2019.05.024)
- [7] Dartmouth College, Reading Brains Lab: THE ERP Technique. "Reading brains lab: the ERP technique" [Online]. Accessed: 05-Nov-2020. Available: <https://www.dartmouth.edu/~readingbrains/ResearchFiles/ERPTechnique.html>
- [8] N. Badcock, P. Mousikou, Y. Mahajan, P. De Lissa, J. Thie, G. McArthur, . . . KE, S. "Validation of the Emotiv EPOC® EEG gaming system for measuring research quality auditory ERPs," *PeerJ.*, Feb. 2013, [10.7717/peerj.38](https://doi.org/10.7717/peerj.38)
- [9] Y. Roy, H. Banville, I. Albuquerque, A. Gramfort, T. H Falk, J. Faubert. "Deep learning-based electroencephalography analysis: a systematic review," *J. Neural Eng.*, vol 16, pg.5, 2019, [10.1088/1741-2552/ab260c](https://doi.org/10.1088/1741-2552/ab260c)
- [10] S. A. Alwedaie, H. A. Khabbaz, S. R. Hadi, and R. A. Hakim, "EEG-Based Analysis for

Learning through Virtual Reality Environment,” *Journal of Biosensors & Bioelectronics*, vol. 09, no. 01, 2018.

[11] Neurotech@Berkeley, “Harnessing the Power of Brain Waves in Virtual Reality: Applications for Gaming and Medicine,” *Medium*, 25-Oct-2019. [Online]. Available: <https://medium.com/neurotech-berkeley/harnessing-the-power-of-brain-waves-in-virtual-reality-applications-for-gaming-and-medicine-ea307a19c8db>. [Accessed: 22-Apr-2021].

[12] “2021 Engineering Salary Statistics: College of Engineering,” *Michigan Technological University*. [Online]. Available: <https://www.mtu.edu/engineering/outreach/welcome/salary/#:~:text=According%20to%20the%20U.S.%20Bureau,takes%20to%20become%20an%20engineer>. [Accessed: 22-Apr-2021].

Appendix

Appendix A: Proposal (ECE-492):

Draft Proposal

VR Learning Environment with Real Time Brain Signal Monitoring

Brendan Jenkins, Ethan Li, Jacob Mitchell, Yumna Rizvi, Zayne Frabutt

FS: Nathalia Piexoto

ECE 492-001

September 20th, 2020

Executive Summary

With the rise of the Covid 19 pandemic, the world has been forced to quickly adapt to unfamiliar circumstances. Public safety guidelines have spurred a mass transition from face-to-face meetings into online workspaces. This migration to online workspaces has had a huge impact on the normal workday, but it also caused many changes to the traditional learning environment. Although there have been incredible efforts from schools and teachers, some students are struggling to adapt to new teaching techniques.

Distance learning introduces many new challenges and distractions to the average school day. The usage of collaborative environments aids teachers with relaying information, but there are many subjects that are taught more effectively through hands-on learning. Some ways of simulating this hands on experience is through virtual labs and videos. This method can be lacking, however, and it doesn't give the immersive feeling of being in a lab. One way to improve this method is to implement virtual reality to aid with learning. Virtual reality can give a fully immersive experience that makes you feel like you're actually there. It also allows you to experience things that would never be possible otherwise, such as walking through an old civil war museum or look at the individual parts of a cell.

While virtual reality by itself is a powerful tool, we can take it one step further to make it even more useful for schools. By implementing a brainwave headband reader, such as the EMOTIV's EPOC+, we can monitor students' attentiveness and understanding of a certain topic. In doing so, teachers can take an active role in the students' learning. This allows teachers to challenge a student who is thriving, or help along one who is falling behind. Brain waves can be rather complex, though, so we will implement an easy to read centralized interface for the teacher to monitor a large group of students at once. To do so, we will use information gathered from the EPOC+ and compare them to generalized brainwave patterns for concentration, understanding, and more.

A customized game incorporating both VR and brain wave pattern analysis will allow for a deeply immersive experience tailored to a course's needs. Almost every course could benefit from this, but hands-on intensive courses in the STEM and medical fields could have incredible potential. The specially fit headset we will design alongside the EPOC+ will be a perfect tool for implementing these kinds of games. Technology in virtual learning environments could be a huge market in the future.

Problem Statement

Introduction/Motivation

During the COVID -19 pandemic, learning can be extremely difficult due to the situation it puts students and teachers in. With most schools and universities transitioning to virtual learning, it can be especially tough for those students who benefit the most from in-person teachings. As this has been a pressing issue over the past seven months, the issue needs a solution to help both students and teachers in this situation. Most students learn in different ways and most can't focus in these virtual learning environments as external factors can take over that would otherwise not be there in a classroom environment. This creates major limitations in virtual learning environments. Shown in figure 1, is a study which teachers and students identified the limitations of virtual learning as a whole.

		<i>Limitations</i>
Inefficiency	Unable to teach skills	"In anatomy, the study through models was good. But hands on training is not possible, the student will not be able to understand properly. Skills needs actual hands on training".
	Lack of student feedback	"I find it annoying that during lectures you don't have students feedback whether they are getting the point or not".
	Limited attention span	"There is no continuity of lecture. We lose our concentration and the syllabus is so lengthy."
	Lack of attentiveness	"As the students know that they will get the recordings, they don't listen the lecture properly".
	Resource intensive	"Lots of people might not be having these gadgets. Buying these gadgets comes an extra burden on them in such stressful situation".
Maintaining academic integrity	Lack of discipline	"There is some problem coming with discipline, some students use to misbehave during lectures".
	Plagiarism	As this system is new to everyone, it is difficult to have individual assessment. During assignment, they easily copy paste stuff from web."

Figure 1: Limitations with online learning environments.

Source: Adapted from [1]

This has been an obvious issue since march 2020 to present (September 2020) and there haven't been many major ways this issue has attempted to be mitigated. Our virtual reality learning environment could help not only students but teachers as well. Creating an affordable learning environment through monitoring different aspects of brain signals can help identify what help is really needed for every individual. Overall, many studies such as the one shown in figure 1, identify a multitude of issues with online learning and this project is a great opportunity to help identify and solve these issues.

Identification of need

In this project, our VR environment would need to help monitor the different aspects of learning in an educational environment such as attention span, stress, and other variables through an EEG. Through this data our VR environment would need to provide feedback to the user in order to help each individual with specific issues. This would need to be a low cost solution so it can be accessed by multiple students/teachers.

Market/Application

This product can directly be used by learning institutions specifically for students and teachers who need it. A prevalent problem in the education system amidst a pandemic is the lack of a personalized digital educational experience. A tool to assist learning retention in online education environments via an extended reality platform that is low-cost and accessible for those underrepresented in educational environments and for educators to monitor the effectiveness of their lessons as the goal of the project is to monitor brain signals.

Approach

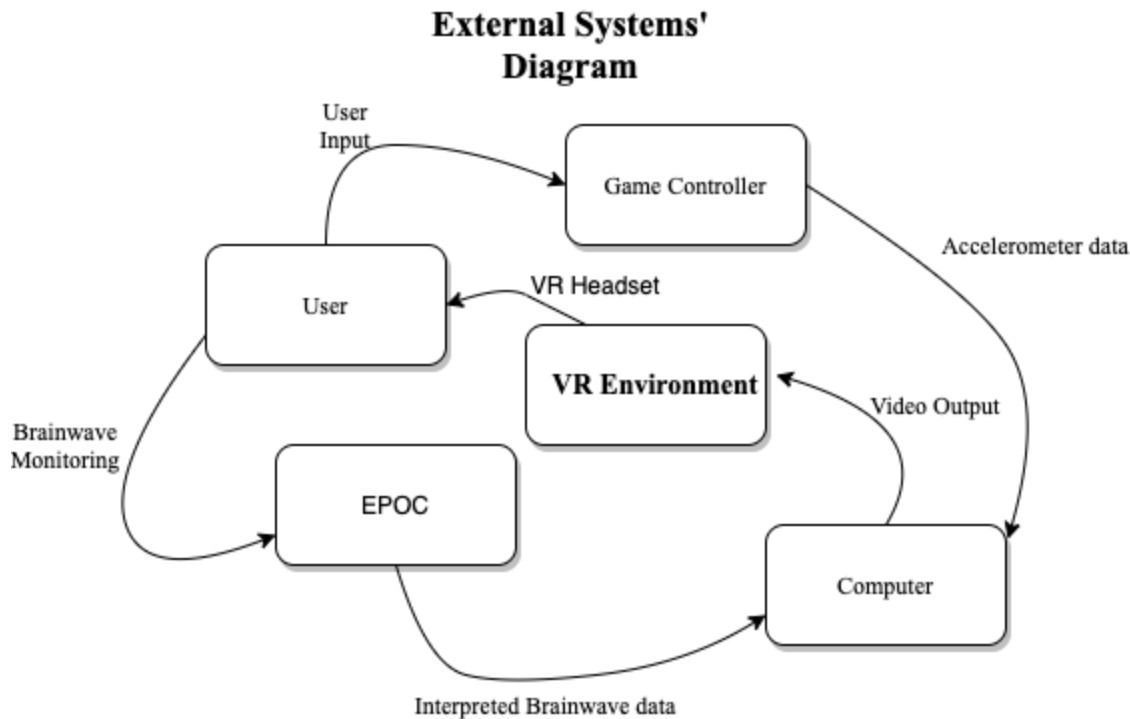
Problem analysis:

The main problems that this project intends to address are issues associated with learning outside of the classroom during the COVID-19 pandemic. Evaluating this solution revolves around several aspects. The first aspect is the safety of students from the pandemic. One of the main purposes of this project is to make distance learning more efficient in order to maintain safety. Assuming that students will obtain access to this hardware through the school, similarly to borrowing a laptop, there are a few safety precautions that need to be considered. Proper precautions such as social distancing, the wearing of masks, and consistent sanitation of areas should be taken when obtaining the hardware. Doing this should preserve the safety of the students and faculty and provide a very safe way to enhance their distance learning experience.

The second issue that needs to be addressed in this project is the implementation of the virtual environment. This environment needs to be intellectually stimulating and provide accurate information that the student can learn from. Many classes such as STEM and medical courses will be ideal subjects for implementation of this project. In order to properly implement a functioning module in a particular subject, the information from the module must be properly researched. These modules can also be made to go along with specific lessons, allowing them to be a supplemental teaching aid to what is discussed in the class meeting.

The third issue that needs to be addressed is the comfortability of the student. This is extremely important if the student will be using the virtual environment for an extended period of time. The proposed solution revolves around modeling a VR headset that fits well with the EMOTIV EPOC+, the brainwave monitoring headset we will be using. 3D modeling a custom headset allows the design to have maximum comfortability. The first issue that is addressed is the orientation of the headset of the student. The headset will be designed to work similarly to other popular VR headsets. The implementation of proper padding and adjustable head straps will allow the user to have the most comfortable experience. The VR headset also needs to allow the EPOC+ to sit in the proper place on the students head. This is the reason for the creation of the custom headset. Modeling the VR headset with the EPOC+ in mind allows for both to fit properly.

The last issue that needs to be addressed in this project is ensuring the usability of the data obtained from the EPOC+. The brain signals obtained would be difficult to interpret for someone who does not fully understand each of the readings. However, this project will have a central hub for the teachers to monitor students to see their concentration and understanding of the information currently being presented. In order to do this properly, we must find a way to extrapolate the data and make it easily readable for people who don't fully understand brain waves.



Approach:

When approaching the solution for building a virtual reality learning environment monitored by brain signals, there are many things that need to be considered. The first thing that needs to be considered is the environment itself. The learning environment needs to be able to hold the attention of the student. A big advantage to using virtual reality is that it cuts out distractions. When wearing a VR headset, the student will be submerged in the virtual environment. This helps mitigate distractions not normally present in a normal classroom. Another important factor to consider in this design is the creation of interesting and relevant modules. These modules can be tailored to fit the needs of specific subjects. Some potential applications could be its use in STEM classes, science labs, and medical applications. A major benefit to using virtual reality for teaching is that it can provide a hands-on experience. Knowing that all students learn in different ways, this can address the problem that many students face when learning outside of the classroom. The creation of a stimulating virtual environment with relevant course material will provide a deeper learning experience to students who are forced to learn in an at home setting.

The second problem that our solution addressed is the methods used to obtain brain signals from students. For this project design, the EMOTIV EPOC+ will be used. While using this headset to monitor brainwaves, different factors in a student's physiological state can be obtained. Among the many things that can be monitored, a student's concentration and relaxation levels can serve as an indication for their real-time experience while doing a module. This can show which

students are most interested in a topic, as well as the students who are struggling. The second issue that our solution addresses is the use of this information. Unless experienced with brain waves, most teachers will not be able to determine brain state from the signals alone. Using the EPOC+, the brain signals can be exported from each student. The monitored patterns will be displayed in an easily readable format through a central location provided for the teachers. This will allow for the teacher to monitor how engaged a student is in real time. Lastly, the modules are designed to give the students lessons of many durations. This means that the hardware for this design must be comfortable for a student to wear for extended periods of time. The creation of a VR headset that works in accordance with the EPOC+ will be important for addressing this. A custom design will allow for maximum comfort when using both the EPOC+ and the VR headset in conjunction. This custom headset will also allow for every contact point of the EPOC+ to sit in the correct location on the students head for proper brain wave monitoring. Implementing all of these factors will allow for the student to get the most from their learning experience in an at-home learning environment.

Alternative Approach:

Due to the nature of this project, there are several possible issues that may arise when attempting to implement this solution. The first potential issue revolves around the creation of the virtual reality headset. Our senior design team has limited experience with 3D modeling. To fully utilize the created virtual environment, a functioning headset is necessary. The consumer market is filled with many virtual reality headsets that are optimized to create the best virtual experience. If the group cannot create a functioning model, we plan to use the Oculus headset. Switching to the approach would cause a significant increase in price, but would provide an ideal experience for using the created modules.

Another major issue that surfaces when implementing our solution is the placements of the VR headset and EMOTIV EPOC+. When using the EPOC+, it is crucial that the placements of the sensors fall in the correct location. As shown in Figure 2 below, the location of the sensors fall in the same space that the VR headset is positioned on the head. If both pieces cannot be positioned correctly, this would compromise either comfort of the VR headset or accuracy of the EPOC+. Also, the EPOC+ covers a large area of the head, potentially leading to difficulties surrounding its use. As a backup, we have intentions on using the MUSE headset to brain wave monitoring. This headset is positioned exclusively on the forehead, allowing for easier use of the two items conjointly. Modeling a VR headset around a headband is also an easier task than modeling one around a whole head covering. The MUSE is our alternative approach because it contains less sensors than the EPOC+, meaning that less data about physiological state can be obtained from it.

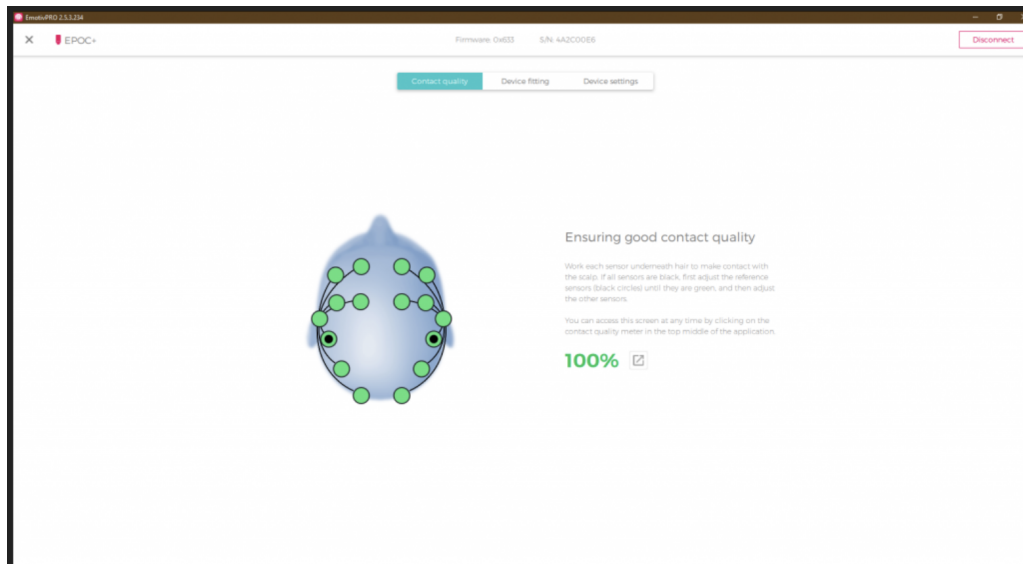


Figure 2: Placement of EPOC+
Adapted from [3]

The last potential issue doesn't pose as big of a problem but it still needs to be considered. For this project we intend to use an LCD screen to display the virtual environment. Implementing a screen into a VR headset would require a design for a larger housing. This would also require an HDMI cord to run into the headset. This was our primary idea for the headset because it allows the code to be written as a computer application instead of a mobile app. If using the LCD is not practical, using a phone for the display in the headset is our alternative plan. This would require mobile app development for the modules. This could be problematic because our team has limited experience in this area. Switching to this alternative approach would require more time to produce a finalized product, but the functionality should remain the same

Intro to Background Knowledge:

When attempting to understand readings from EEG, it is essential to understand the different types of brainwaves that readings are obtained from. Distinctions between different categories of brain waves lie in their variation in frequency. Different brain wave categories are responsible for different brain functions. When using EEG, the primary wave types that are measured consist of Beta, Alpha, Theta and Delta [5]. Beta waves, typically ranging from 12 to 38 hertz, “dominate our normal waking state of consciousness when attention is directed towards cognitive tasks and the outside world” [4]. These types of waves are ideal for studying brain function, making them an ideal method to determine brain state during our designed modules. Alpha waves generally signal thoughts, aiding in, “mental coordination, calmness, alertness, mind/body integration and learning” [4]. Alpha waves will also be very important when monitoring the physiological state of a student. Tracking these waves provides a good indication of active thought process and potentially attention span. Theta waves act as a gateway to cognition, demonstrating when the

user is in the process of falling asleep or waking up [4]. Delta waves are generally only present when in a state of deep sleep. These two types of waves probably will not be as useful for application in this lab. Figure 3 below provides differences in frequency and potential uses for each of the brainwave categories. It should be noted that this chart includes Gamma waves. The EPOC+ has the potential to read gamma waves, but being the highest frequency waves, it may be harder to interpret them. Understanding the distinction between brain wave types and frequencies will allow for a better understanding of the information gathered from the EPOC+.

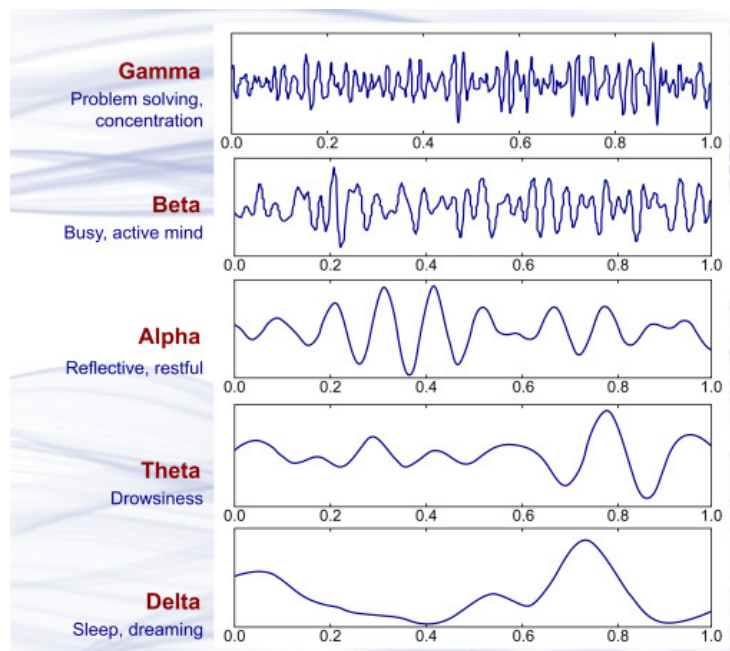


Figure 3: Brain Wave Samples Usable to Determine Concentration
Adapted from [2]

Project Requirements Specification:

Mission Requirements

- We will use the Emotiv headband to collect brain signal data as inputs for an educational virtual reality environment.

Input/Output Requirements

- The Emotiv shall output brain signal data packets.
- The computer shall accept an input from a user through brain signal data via data exported.
- The computer shall provide power to the VR headset and its components.
- The computer will use the data and output the results into the VR environment.
- The computer shall accept an input from the user through the VR controllers.
- VR controllers will receive power from 2-4 1.5V AA batteries.

Functional Requirements

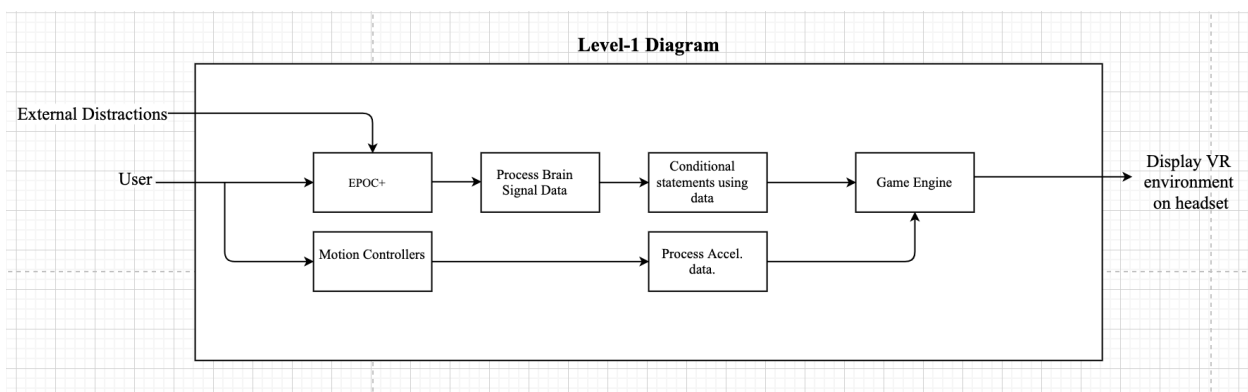
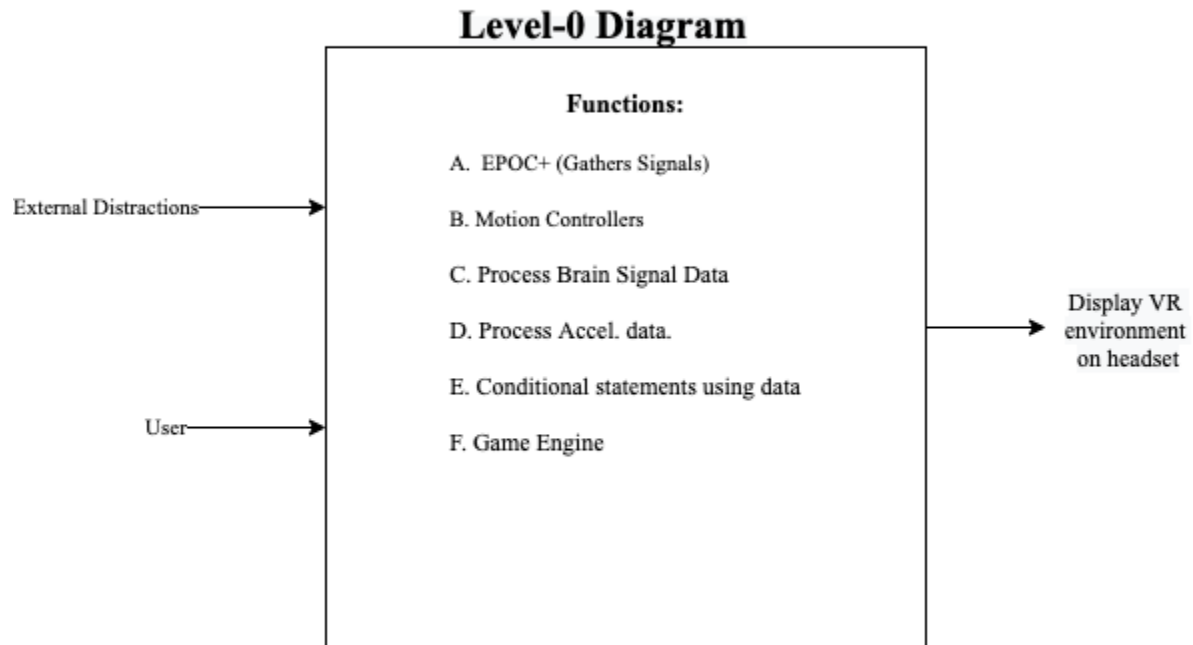
- The Emotiv will record brain signals at a rate of 128 samples per second.
- The Emotiv shall send data over to the computer with no packet loss.
- The software will detect variations in brain signals while interacting with a virtual reality environment.
- The program will be written efficiently to prevent data loss and reduce latency.

Technology and System-Wide Requirements

- A dedicated series graphics card with HDMI output ports.
- two USB 2.0 ports (at least one them is powered)
- Bluetooth will be required for use of Emotiv and controllers.
- Windows 7 or higher operating system
- Our VR headset will be cost efficient as possible to fit the needs of affordability for students.
- Controllers will use an accelerometer with I2C interface to help locate placement of the user in game.
- Hardware will be simplified and straightforward for easy integration with software.

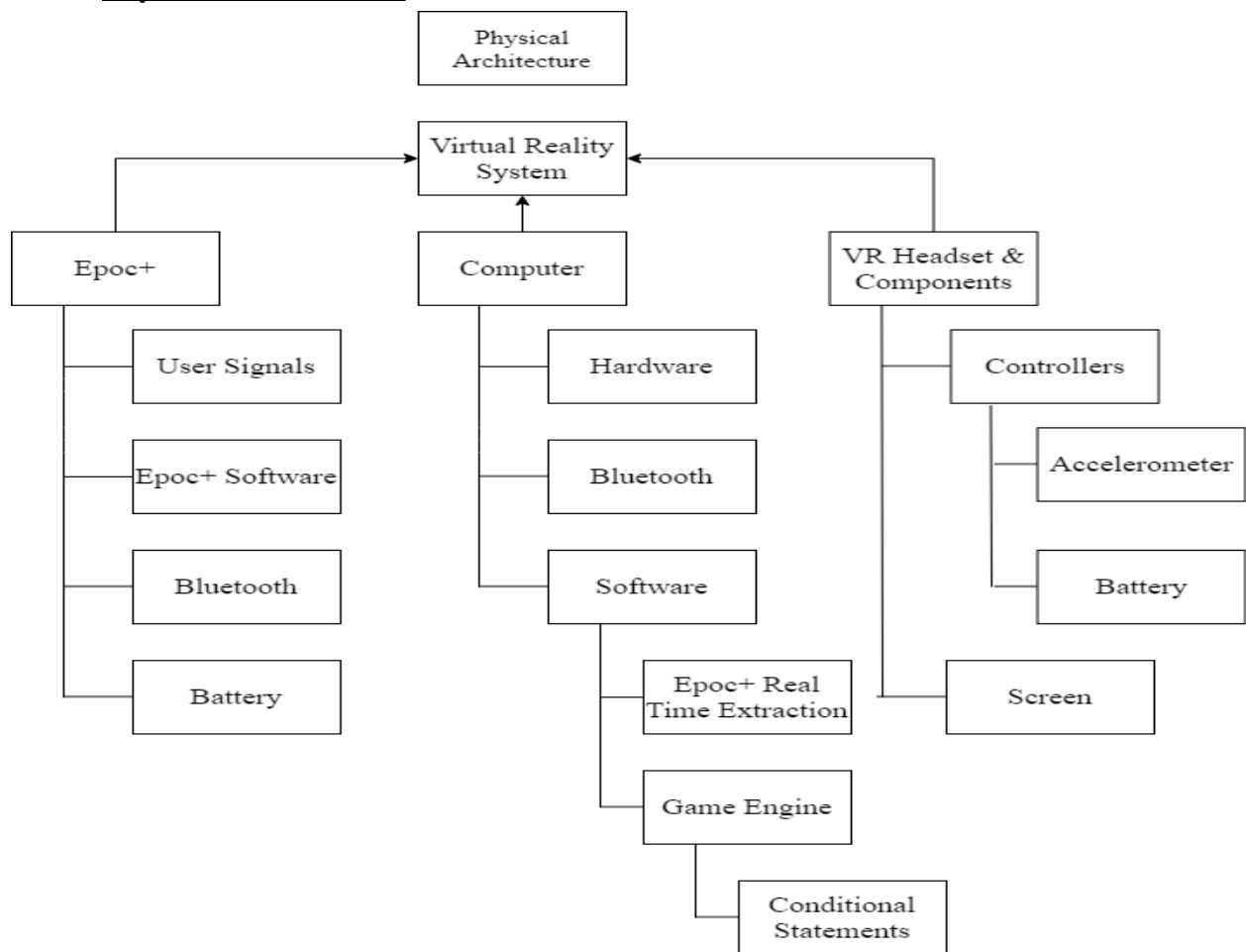
Preliminary Design

Functional decomposition



System Architecture

- Physical Architecture:

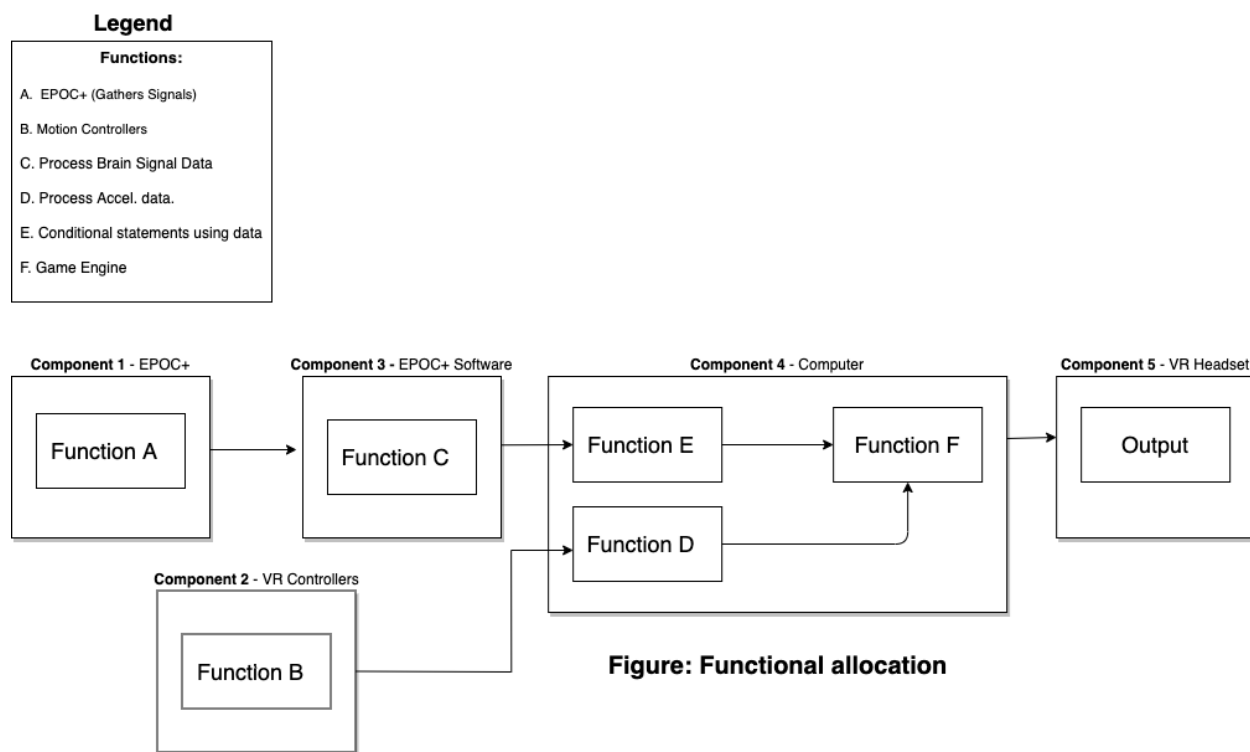


Component selection

- Component 1 - Epoc +
 - The Epoc + is a mobile EEG testing headset that will allow us to track brain signals that we will use as inputs to our design.
 - Provided to us by Nathalia Peixoto. The Epoc + has been discontinued and replaced with a newer version, the Epoc X, which is \$849. This version will be compatible with our design as well.
 - Allows for 12 hours of battery life tracking 14 different channels of brain signals.
 - Link - <https://www.emotiv.com/epoc/>
- Component 2 - VR Controller
 - We will make our own controllers for arm movement tracking and button selection. This will serve as inputs to our design.
 - The controllers will have accelerometers, buttons, pcb design, batteries, and a cover to hold everything. In total, it will cost around \$15 to make each controller.

- Alternatively, if it becomes too inefficient to create our own, we will use controllers that come with a commercial VR headset.
- Component 3 - Epoc Software
 - Emotiv created commercial software to bundle with their EEG testing headset. We will be buying their student version of the software.
 - The software will include real time playback, import/ export, converting to recordings, and the raw EEG API software. We chose the student version for the raw EEG API software, as it allows for us to connect our data and our virtual environment design together.
 - The software is billed monthly at the cost of \$29.
 - Link - <https://www.emotiv.com/emotivpro/>
 - Alternatively, we will be using the lite version and grab the data ourselves through our own software incorporated into the lite version.
- Component 4 - Computer
 - A computer is required to run the software for the game and for the Epoc software. This will be, at first, through the use of personal computers from our group members.
 - Alternatively, we will be going to one of the school's lab computers to work on our design.
- Component 5 - VR Headset
 - We will be buying the commercial Oculus Virtual Reality headset. This will allow for us to view our design.
 - Alternatively, we will create our own headset through a combination of LCD screens and accelerometers to track head movement.

System Architecture diagram



Preliminary Experimentation Plan

Selection of requirements for experimental validation:

1. Requires use of EPOC+ sampling at a rate of 128 samples per second.
2. Software has a success rate in processing data *at least* 95 percent of the time.
3. Accelerometers are always calibrated correctly with a recalibration method.
4. Hardware has no shorts and is functioning correctly.
5. EPOC+ signals are able to be read and exchanged with the software with minimal amount of latency.
6. Brain signals are able to interact with the game and are reflected through a heads up display or action within the gameplay.

Experiment 1 - *Collecting Brain Signal Data and Data Transfer, along with getting the virtual environment set up.*

In this experiment, we plan to collect electroencephalogram (EEG) brain waves from our Emotiv device and transfer the data to our software. While conducting these experiments we will test what brain waves are associated with playing certain kinds of educational games and being in different scenarios and see and document changes in brain waves associated with those changes. After documenting different changes associated with different brain signals, we will introduce real time extraction of the data into the transfer of signals. This will allow us to incorporate conditional statements within our code to help the signals interact with the virtual reality environment. We will also be testing to see if our basic configuration of a virtual environment simulates correctly. This environment will have little to no inputs connected and has a main purpose of checking if it displays.

Experiment 2 - *Testing how VR software interacts with brain signal data*

Experiment 2 has a goal of transferring the brain signal data at real time to the virtual reality environment. This will be a test of if the signals are strong enough to be detected and if the conditional statements are being enacted. We will go about this by setting up a simple environment with 14 different conditional statements, one for each of the EEG nodes, and have the user try to interact with any of those statements. The goal will be to utilize all 14 conditional statements in one way or another. This will also test if our game is structured and running properly.

Experiment 3 - *Hardware/Structural Verification/Testing*

Within experiment 3, this is where we would test our hardware's performance to make sure the system works properly in conjunction with our software. Part one would be our hardware verification tests. In which, we would be testing hardware under conditions simulating expected real-life conditions, including storage, transportation, operation and maintenance environments. Secondly for this experiment we would test the hardware. We would begin by developing a set of test criteria while applying functional tests to determine whether the test criteria have been met, then applying qualitative assessments to determine whether the test criteria have been met. Lastly, we would find any weaknesses with our hardware and create a way to fix/contain the problems post experiment.

Preliminary Project Plan

List of Tasks for 493

1. 3D designing and printing our VR headset housing.
2. Begin fabrication and assembly of hardware.
3. Begin writing software for our VR game.
4. Start experimenting with collection of brain signals and draw conclusions from those tests.
5. Request access and get familiar with Emotivpro software (student version).
 → These tasks will be completed in parallel.
6. Begin testing Hardware
7. Begin testing Software.
8. Test hardware/software compatibility.

Allocation of Responsibility

Tasks:

3D designing printing our VR headset housing - Ethan

Begin fabrication and assembly of hardware - Yumna

Begin writing software for our VR game - Jacob

Start experimenting with collection of brain signals and draw conclusions from those tests
- Brendan

Request access and get familiar with Emotivpro software (student version) - Zayne

Begin testing Hardware - Brendan / Yumna / Ethan

Begin testing Software - Jacob / Zayne / Ethan

Test hardware/software compatibility - Whole Team

Potential Problems -

Risk Analysis & Knowledge

Project Objective	Risk Description	Effect	Risk Severity	Risk Likelihood	Risk Response Strategy
Financial Cost	Underestimation of budget - insufficient to carry out designing tasks	Deterioration of project quality	Medium	0-4%	Limit scope of design to a necessary, manageable, minimum
Skills & Information	Unavailability of core skills affecting designing & building processes. Unavailability of information & lack of understanding of new software required, including access to open source software & expertise	Delays & errors in design & implementation - verification increases cost & time due to developments of revisions	Low	50-90%	Team strengthens skills by consulting with experts & transparency about skills needed to be learnt. Information required is obtained in advance of requirement
Durability & Compatibility	Problems due to the lack of foresight in the durability of 3D constructed products as well as the compatibility with other materials and software	Deterioration of quality. Delays & errors. Increases in costs if materials aren't durable or compatible.	High	30-60%	Various phases of testing & troubleshooting for compatibility of hardware & software in a timely manner as a project priority. Important tools & components are stored securely.
Opinions & Permission	Delays in obtaining opinions & permissions from faculty advisors regarding scope of the project, completion of tasks and access to resources	Disturbed project process & suspension of work	Medium	30-60%	Ordering & reserving resources. Alternatives are considered ahead of time. Earlier diagnosis of the situations is handled in meetings preceding communication with advisors.

Internal Conflicts	Conflicts amongst members due to insufficient information & communication.	Delays & disruption in completion	High	0-4%	Response of all team members to communicate & mediate conflicts in the team
Completion	Unrealistic goals & deadlines	Deterioration of design quality & failure to meet deadlines	High	0-4%	Response of all team members to prioritize completing goals ahead of deadlines. Meetings preceding deadlines to review progress.

Bibliography

- [1] K. Mukhtar *et al*, "Advantages, Limitations and Recommendations for online learning during COVID-19 pandemic era," *Pakistan Journal of Medical Sciences Quarterly*, vol. 36, 2020.
- [2] Priyanka A.Abhang, Bharti W.Gawali, Suresh C.Mehrotra, "Technological Basics of EEG Recording and Operation of Apparatus," Dr. Babasaheb Ambedkar, Aurangabad, India, 2016, ch. 2, pp. 19-50. <https://doi.org/10.1016/B978-0-12-804490-2.00002-6>
- [3] Z. Scavotto, N. Peixoto, "Video Games and Brain Wave Frequencies," 2020. [Online]. Available: <http://gamingwaves.onmason.com/>
- [4] Brainworks, "What Are Brain Waves?," 2020. [Online] Available: <https://brainworksneurotherapy.com/what-are-brainwaves>
- [5] Emotiv, "How Does an EEG Work?," 2020. [Online] <https://www.emotiv.com/eeg-guide/>
- [6]

Appendix B: Design Document (ECE-492):

Final Design Document

VR Learning Environment with Real Time Brain Signal Monitoring

Brendan Jenkins, Ethan Li, Jacob Mitchell, Yumna Rizvi, Zayne Frabutt

FS: Nathalia Piexoto

CC: Peter Pachowiz

ECE 493-001

April 30th, 2021

Problem Statement

Introduction/Motivation

During the COVID -19 pandemic, learning can be extremely difficult due to the situation it puts students and teachers in. With most schools and universities transitioning to virtual learning, it can be especially tough for those students who benefit the most from in-person teachings. As this has been a pressing issue over the past seven months, the issue needs a solution to help both students and teachers in this situation. Most students learn in different ways and most can't focus in these virtual learning environments as external factors can take over that would otherwise not be there in a classroom environment. This creates major limitations in virtual learning environments. Shown in figure 1, is a study which teachers and students identified the limitations of virtual learning as a whole.

		<i>Limitations</i>
Inefficiency	Unable to teach skills	"In anatomy, the study through models was good. But hands on training is not possible, the student will not be able to understand properly. Skills needs actual hands on training".
	Lack of student feedback	"I find it annoying that during lectures you don't have students feedback whether they are getting the point or not".
	Limited attention span	"There is no continuity of lecture. We lose our concentration and the syllabus is so lengthy."
	Lack of attentiveness	"As the students know that they will get the recordings, they don't listen the lecture properly".
	Resource intensive	"Lots of people might not be having these gadgets. Buying these gadgets comes an extra burden on them in such stressful situation".
Maintaining academic integrity	Lack of discipline	"There is some problem coming with discipline, some students use to misbehave during lectures".
	Plagiarism	As this system is new to everyone, it is difficult to have individual assessment. During assignment, they easily copy paste stuff from web."

Figure 1: Limitations with online learning environments.

Source: Adapted from [1]

This has been an obvious issue since march 2020 to present (September 2020) and there haven't been many major ways this issue has attempted to be mitigated. Our virtual reality learning environment could help not only students but teachers as well. Creating an affordable learning environment through monitoring different aspects of brain signals can help identify what help is really needed for every individual. Overall, many studies such as the one shown in figure 1, identify a multitude of issues with online learning and this project is a great opportunity to help identify and solve these issues.

Identification of need

In this project, our VR environment would need to help monitor the different aspects of learning in an educational environment such as attention span, stress, and other variables through an EEG. Through this data our VR environment would need to provide feedback to the user in order to help each individual with specific issues. This would need to be a low cost solution so it can be accessed by multiple students/teachers.

Market/Application

This product can directly be used by learning institutions specifically for students and teachers who need it. A prevalent problem in the education system amidst a pandemic is the lack of a personalized digital educational experience. A tool to assist learning retention in online education environments via an extended reality platform that is low-cost and accessible for those underrepresented in educational environments and for educators to monitor the effectiveness of their lessons as the goal of the project is to monitor brain signals.

Project Requirements Specification

Mission Requirements

- We will use the Emotiv headband to collect brain signal data as inputs for an educational virtual reality environment.

Input/Output Requirements

- The Emotiv shall output brain signal data packets.
- The computer shall accept an input from a user through brain signal data via data exported.
- The computer shall provide power to the VR headset and its components.
- The computer will use the data and output the results into the VR environment.
- The computer shall accept an input from the user through the VR controllers.
- VR controllers will receive power from a 5V micro-usb input.

Functional Requirements

- The Emotiv will record brain signals at a rate of 128 samples per second.
- The Emotiv shall send data over to the computer with no packet loss.
- The software will detect variations in brain signals while interacting with a virtual reality environment.
- The program will be written efficiently to prevent data loss and reduce latency.

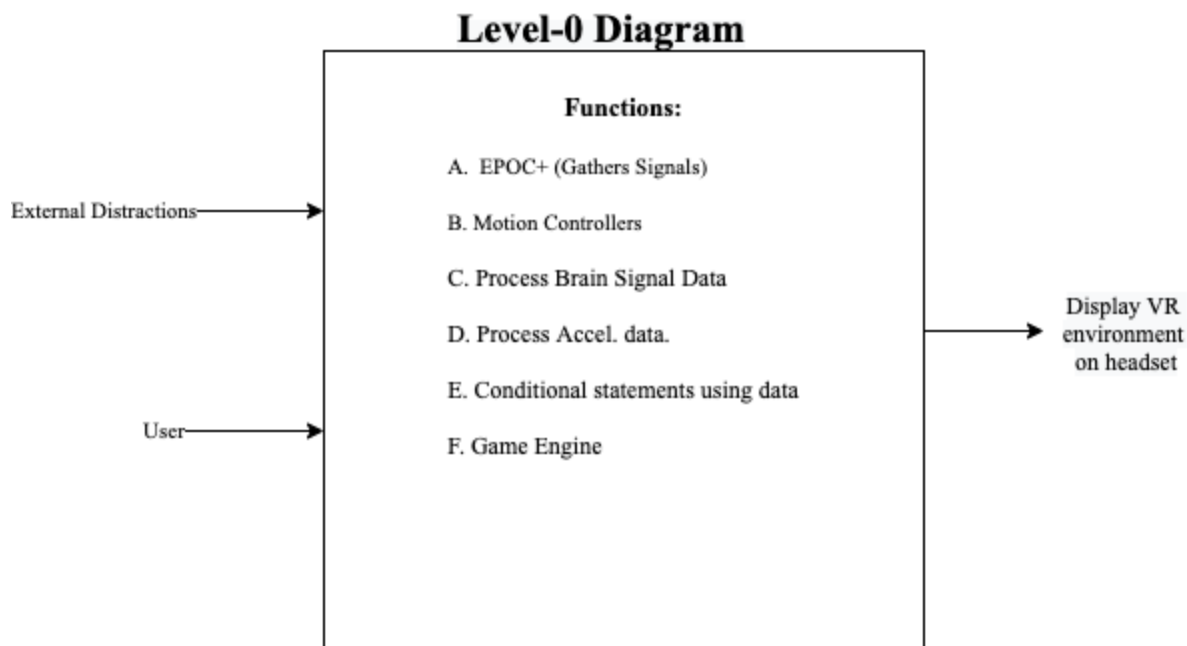
Technology and System-Wide Requirements

- A dedicated series graphics card with HDMI output ports.
- two USB 2.0 ports (at least one them is powered)
- Bluetooth will be required for use of Emotiv and controllers for data transmission.
- Windows 7 or higher operating system.
- Our VR headset will be cost efficient as possible to fit the needs of affordability for students.
- Controllers will use an accelerometer with I2C interface to help locate placement of the user in game.
- Hardware will be simplified and straightforward for easy integration with software.

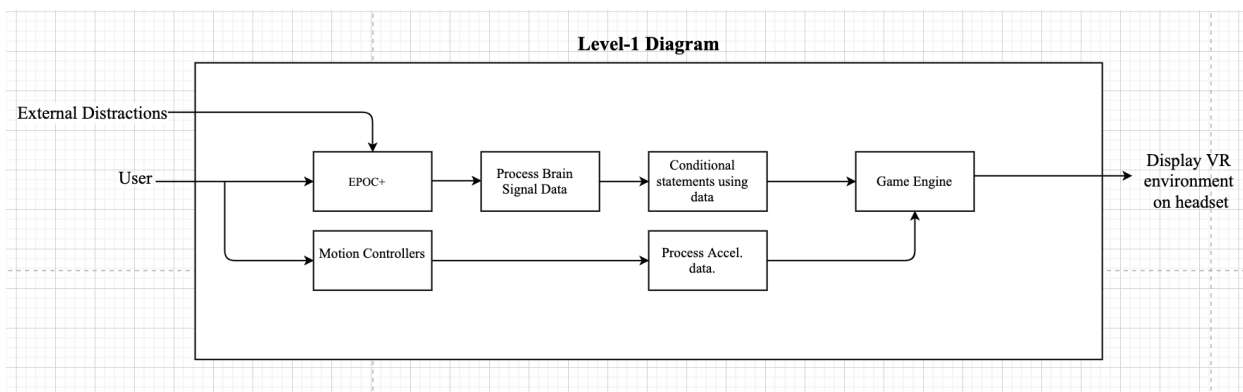
System Design/Architecture

Functional decomposition

Level-0 System Decomposition

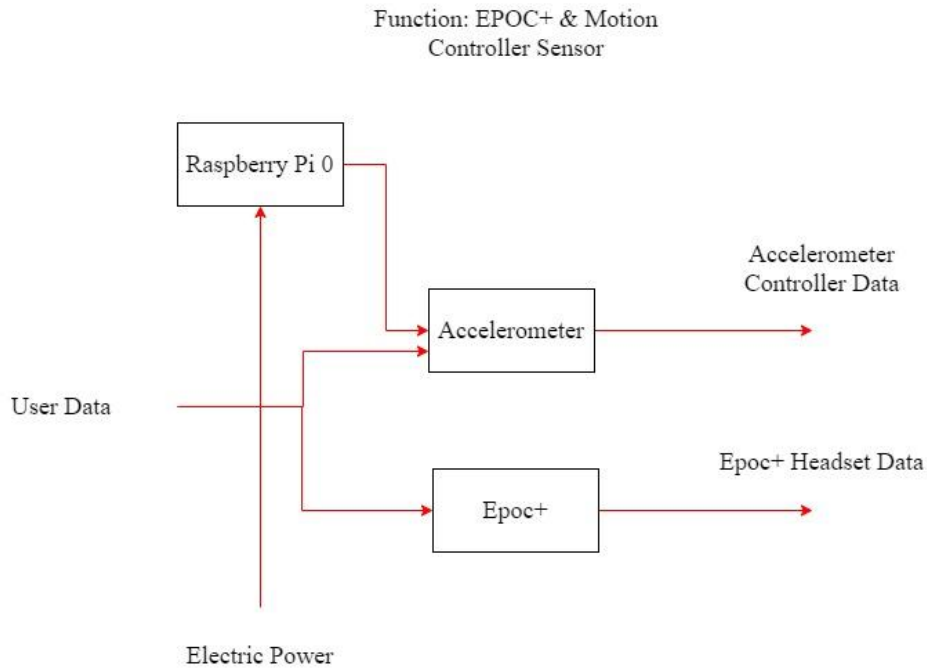


Level-1 System Decomposition

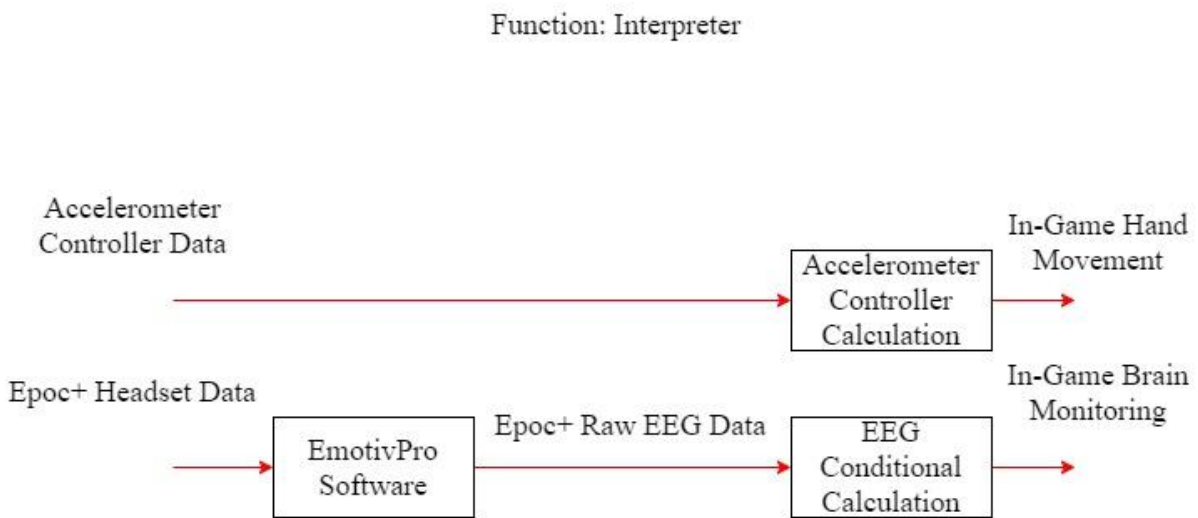


Level-2 System Decomposition

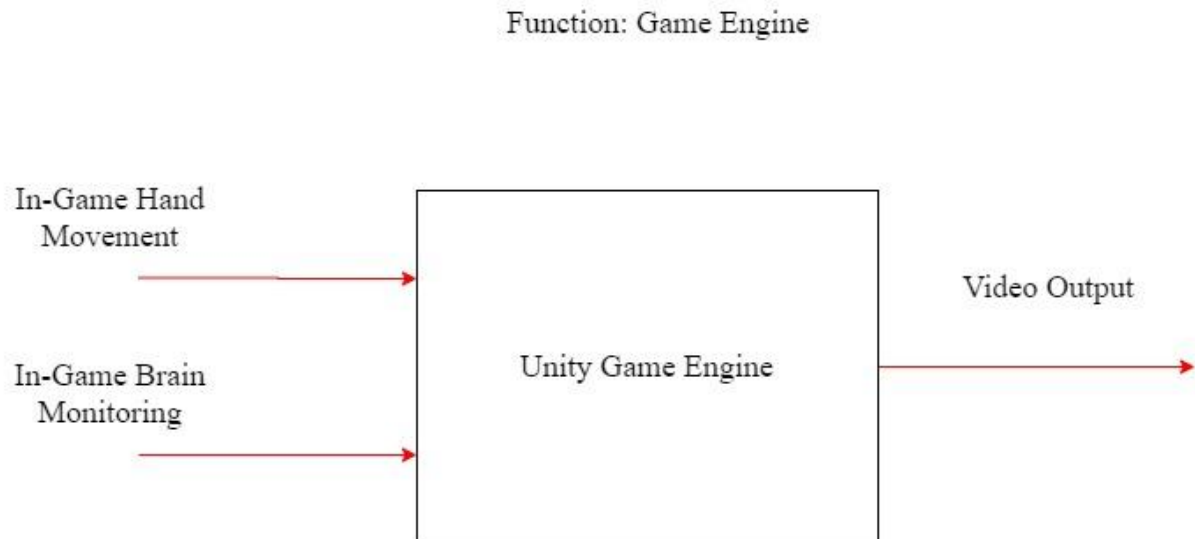
❖ Breakdown Of Functions A & B (EPOC+ & Motion Controllers)



❖ Breakdown Of Functions C, D, & E (Processing of Data)

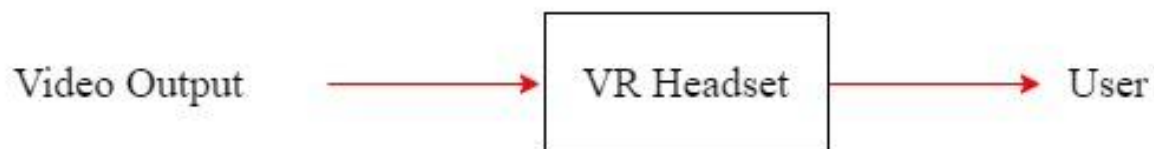


❖ Breakdown Of Functions F (Game Engine)



❖ Breakdown Of Output

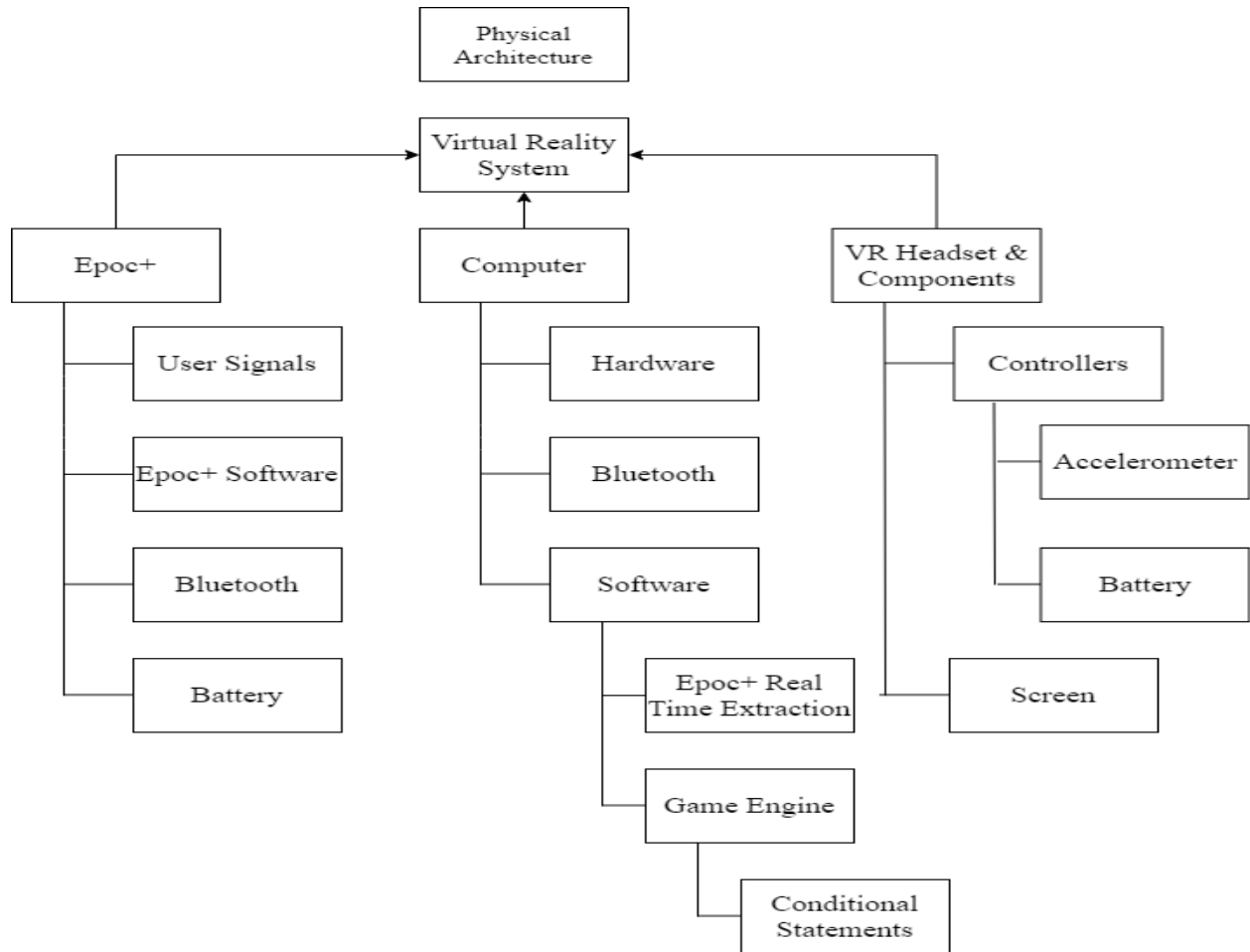
Function: Virtual Reality Display



System Architecture

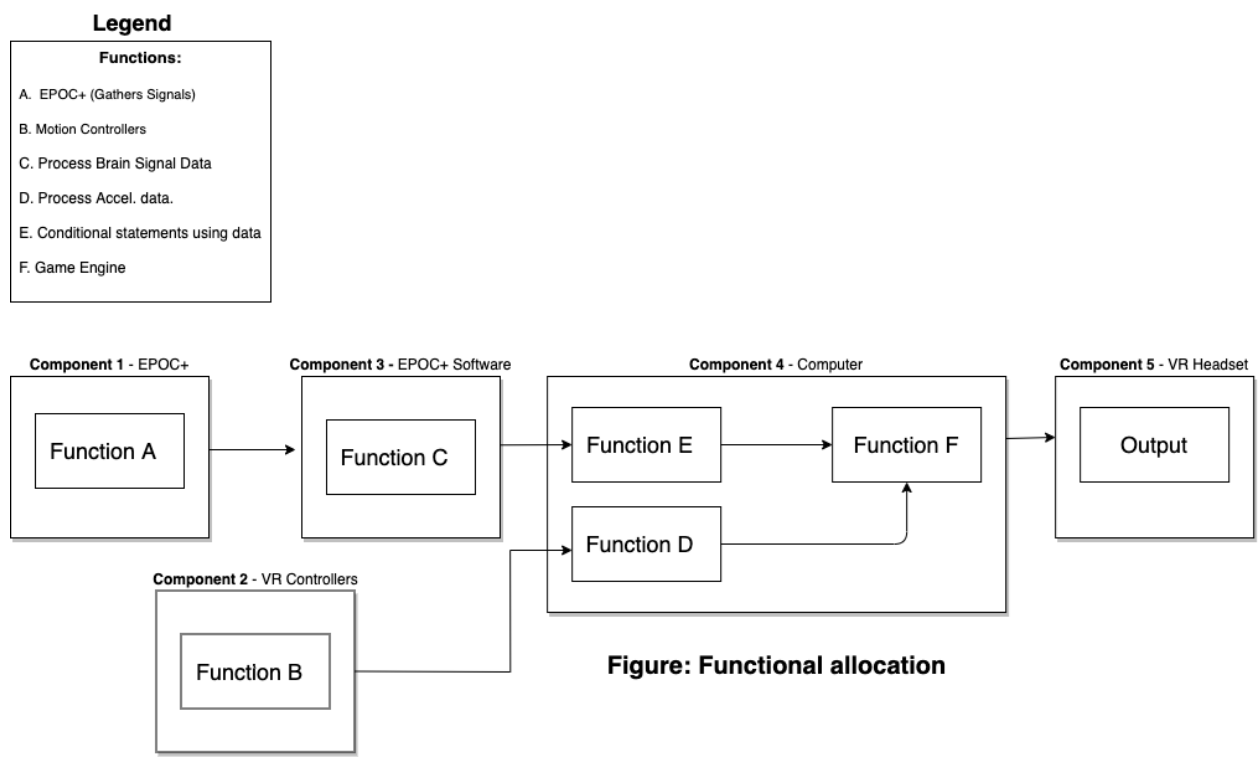
Physical Architecture

- ❖ This Figure shows a breakdown of Architecture the physical architecture of our three main components and their associated components.



System Architecture diagram

❖ This Figure shows a combined process from both our decomposition and physical architecture to show the process of how each of these components work.



Background Knowledge/Phenomenology

❖ Accelerometer as a 3D positioning mechanism

Accelerometers are sensitive to both linear acceleration and the local gravitational field. Changes in orientation are described by rotations in roll ϕ , pitch θ and yaw ψ about the x, y and z axes respectively

Equations to calculate pitch (y-axis), roll (x-axis), and yaw (z-axis) angles:

$$\tan \phi_{yxz} = \frac{G_{py}}{\sqrt{G_{px}^2 + G_{pz}^2}}$$

Figure: Eq. for Pitch
Source: Adapted from [9]

$$\tan \theta_{xyz} = \frac{-G_{px}}{\sqrt{G_{py}^2 + G_{pz}^2}}$$

Figure: Eq. for Roll
Source: Adapted from [9]

Where G_{px} , G_{py} , and G_{pz} are raw data outputs are read from the accelerometer that must be scaled to an interpretable value. The arctangent is taken on the right side of the equation to help get angle values.

Angle calculations were used in our code to help with angle positioning. Show below is a snippet of our code used with these equations:

```
float yAngle = atan( ay / (sqrt((ax*ax) + (az*az))))*180/M_PI;
float zAngle = atan( sqrt((ax*ax) + (ay*ay)) / az)*180/M_PI;
float xAngle = atan( ax / (sqrt((ay*ay) + (az*az))))*180/M_PI;
```

❖ EEG for sampling of Brain Signals

An EEG picks up the electric potential differences, on the order of tens of μV . The potentials measured, therefore, reflect neuronal activity and can be used to study a wide array of brain processes [3]. To collect EEG data, electrodes are placed on the scalp and wet with conducting liquid to facilitate the measurement of the electrical activity using scalp electrodes [10].

Event-related potentials (ERPs) are electrical potentials in the brain in response to specific events. While the EEG records ongoing signals from the electrodes in the EPOC+, different types of events are presented for the brain to respond to; for example, written or spoken words, letters, pictures, or sounds. By measuring the brain's response to these different kinds of events, conclusions can be drawn about how the brain processes different types of information [7].

The table below shows the brain waves the EPOC+ measures. They are classified as gamma, beta, alpha, theta, and delta, each measured at different frequencies [3]. These different brain wave categories are responsible for different brain functions. Applications and monitoring of these allow unbiased measures of, for instance, an individual's level of fatigue, mental workload, mood, or emotions. Beta waves, typically ranging from 12 to 38 hertz [4]. These types of waves are ideal for studying brain function, making them an ideal method to determine brain state during our designed modules. Alpha waves generally signal thoughts, aiding in, “mental coordination, calmness, alertness, mind/body integration and learning” [4]. Alpha waves will also be very important when monitoring the physiological state of a student. Tracking these waves provides a good indication of the active thought process and potential attention span. Theta waves act as a gateway to cognition, demonstrating when the user is in the process of falling asleep or waking up [4].

TABLE 2.1 Characteristics of the Five Basic Brain Waves

Frequency band	Frequency	Brain states
Gamma (γ)	>35 Hz	Concentration
Beta (β)	12–35 Hz	Anxiety dominant, active, external attention, relaxed
Alpha (α)	8–12 Hz	Very relaxed, passive attention
Theta (θ)	4–8 Hz	Deeply relaxed, inward focused
Delta (δ)	0.5–4 Hz	Sleep

**Table showing characteristic of basic brain waves
Source: Adapted from [4]**

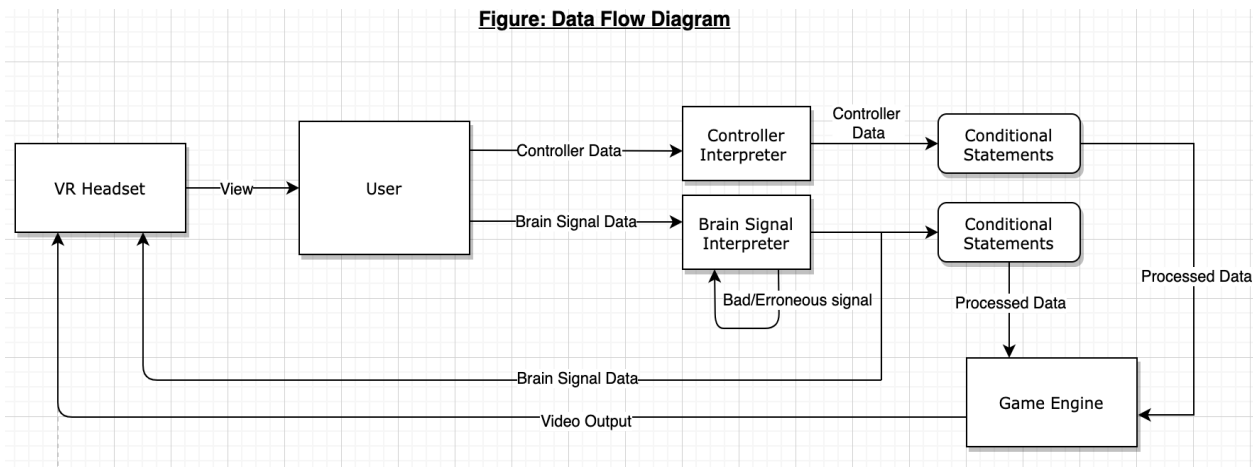
The EPOC+ headset includes 14 sensors and 2 ground reference points to which the voltage of all other sensors are compared. The 14 scalp sensors (channels) are high-pass filtered with a 0.16 Hz cut-off, pre-amplified, and low-pass filtered at an 83 Hz cut-off. The analog signals are then digitized at 2048 Hz. The digitized signal is filtered using a 5th-order sinc notch filter (50–60 Hz), low-pass filtered, and down-sampled to 128 Hz. The effective bandwidth measured is 0.16–43 Hz. [2]

Even though the EEG is a critical tool to this project, it is still important to acknowledge that it suffers from a few limitations that hinder its effective analysis and processing. An EEG has a low signal-to-noise ratio (SNR). The brain activity measured can be often buried under multiple sources of environmental, physiological, and activity-specific noise of similar or greater amplitudes. The EPOC+ uses various filtering and noise reduction techniques to minimize the impact of these noise sources and extract true brain activity from the recorded signals [10].

A problem that may be encountered is that the EEG is a non-stationary signal, and its statistics vary across time. As a result, a characteristic trained on a temporally-limited amount of user data might generalize poorly to data recorded at a different time on the same individual. This is important for instances where the EEG is working with limited amounts of data [10].

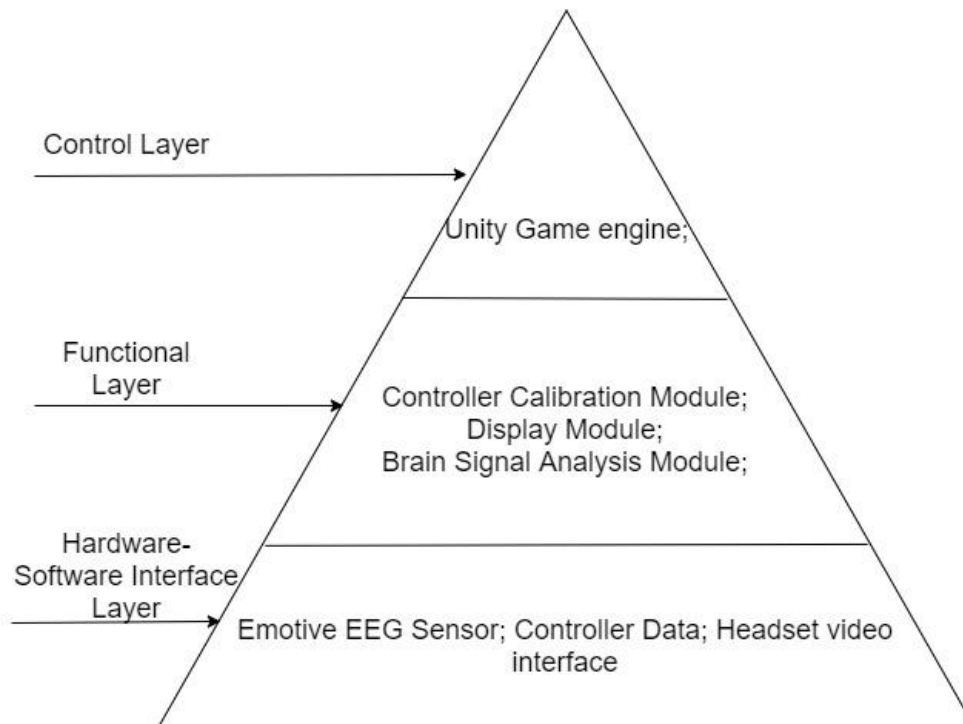
Detailed Design

Data Flow Diagram



The data flow diagram above depicts how the data will move and be analyzed throughout our project. The EPOC+ will gather brain signal data and input it into a program that will analyze the data and allow us to use the data to modify how the module progresses. While this happens, the user will also be using the controller. The data extracted from the controller will then be interpreted by a different program that will allow the user to move within the virtual environment. These two things together will then be analyzed by the game engine to provide an interface with the pi zero using GPIO pins 17 and 27.

Software Structure



As Hardware, we have the Epop+ sensors, accelerometer controllers, and the virtual reality headset. To bridge them to software, we have the controller calibration module, display module, and brain signal analysis module. All of these will lead into our control layer, the unity game engine.

Prototyping progress report

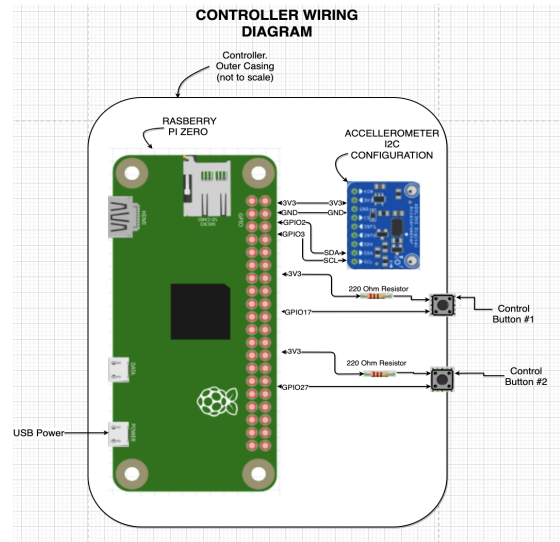
Task #1) Controller Modeling (Option #2)

- Objective: Create a CAD model for a 3D controller

3D Model:



Wiring Schematic:

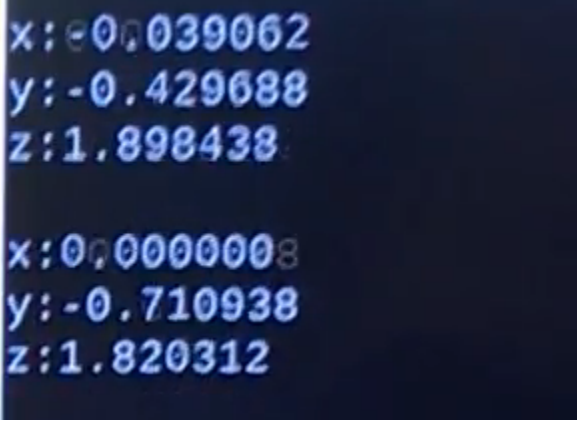


- Controller Information:
 - Two Buttons
 - Action
 - Settings Menu
 - Raspberry Pi Zero
 - Sized to fit within the casing
 - Ability to communicate with the Unity environment via Bluetooth
 - ADXL345
 - Maps information on x, y, and z planes
 - Dimensions: 105x45x22 mm
- Conclusion:
 - A model for a controller can be designed to fit within a single-handed controller given the components are small enough
 - A controller design can be created simple enough for use by anyone
 - This still requires calibration and mapping of the ADXL
 - A single cord can connect the controller to the computer due to the capabilities of the Raspberry Pi Zero
 - Mounting the accelerometer in a centralized location of the controller will allow for the most accurate tilt measurement

Task #2) Mapping the ADXL345

- Objective: Create a program that can interpret accelerometer data

- Getting uniform values from the accelerometer (defined range)
- Calibration of the sensor to understand tilt behavioral curve
- Terminal Output:
 - Stores values for x, y, and z plane
 - X and Y display values from -2 to 2
 - Z displays values from 0 to 4
 - Communication with the accelerometer follows I2C protocol



```
x: -0.039062
y: -0.429688
z: 1.898438

x: 0.000000
y: -0.710938
z: 1.820312
```

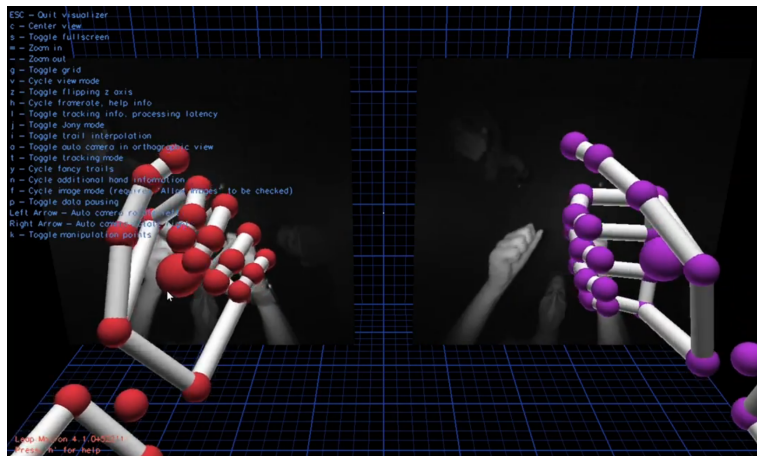
Output Readings from Created Program

- Conclusion:
 - Understanding the values from the program is important, but it will be necessary to calibrate the sensor to better understand the angle vs program output
 - After calibration of the sensor, this information should be able to map accelerometer movement in our environment
 - It may only be necessary to calculate the values of X and Y, as determining depth into the environment may not be necessary for implementation

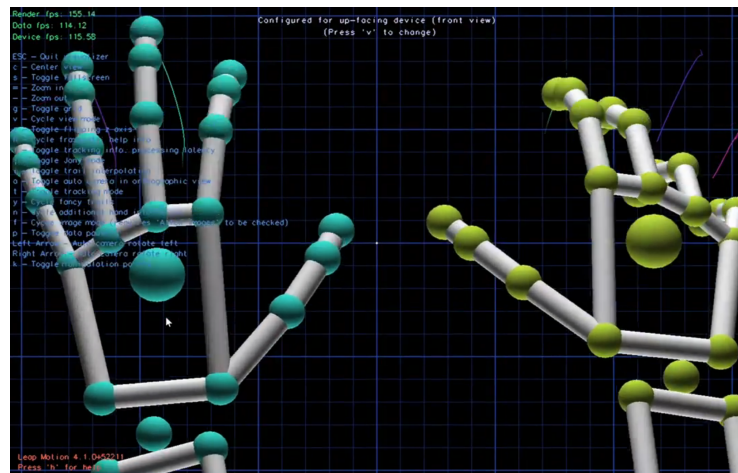
Task #3) Leap Motion Experimentation (Option #1)

- Objective: Understand capabilities of Leap Motion

- Determine possible benefits to using over the controller
- Determine the capabilities for hand recognition from different angles
- Leap Motion Viewpoint on Table:



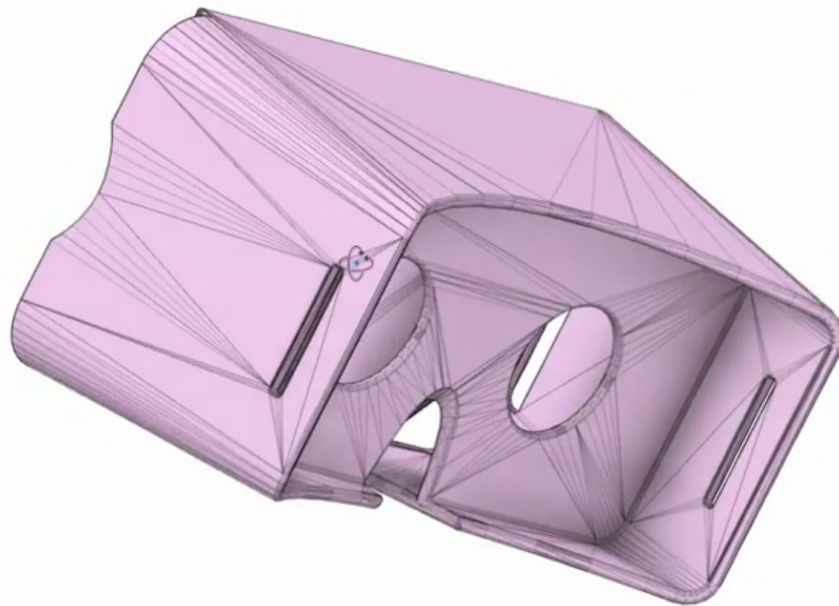
- Leap Motion Head Mounted



- Conclusions
 - Module in unity that allows for Leap Motion interaction with created environment
 - Still need to do testing with this to see capabilities inside the created environment
 - Leap Motions adaptability allows for diverse possibilities for implementation in our design
 - The software maps each hand by joint. This allows for hands shapes to be determined through mapping and multiple contract points when being used for interacting with objects.

Task #4) Creation of Headset 3D Model

- Objective: Create a 3D model for a headset that will allow for use without interfering with EMOTIV EPOC+ sensor placement
 - Look into the possibility of mounting Leap Motion on front of the headset
- Current Design:
 - Design Components:
 - Main Housing
 - Front Panels
 - Lens Holders
 - LCD Clip



Prototype for 3D Model of Headset

- Conclusions:
 - Although the group is limited with experience regarding 3D headset modelling, this seems well within the range of our capabilities
 - The casing was never the issue with sensor placement, but the custom design allows for implementation with leap motion if that route is chosen
 - The biggest issues with the VR headset interfering with EPOC+ sensor placement stem from the parts of the headset that aren't 3D modeled.
 - This includes the straps that will need to sit on the head. These straps cannot take up a lot of space, but they still need to be strong enough to keep the headset stable.
 - Designing/Finding lenses that fit with a custom model may be difficult.

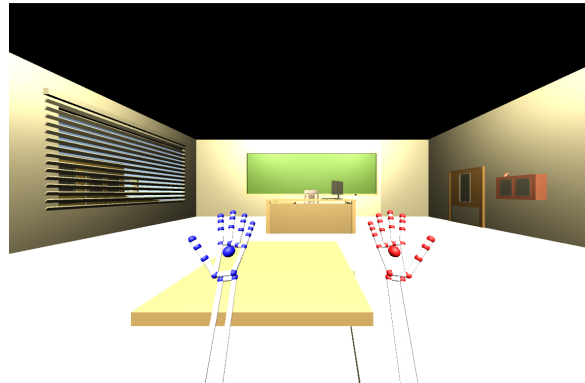
Task #5) Creation of Unity Virtual Environment

- Objective: Familiarize ourselves with the capabilities of Unity Development Software
- Test Design:
 - Implemented the basics of unity design for modelled of a classroom environment
 - Used lighting sources to provide ample experience in scene and game view
 - Implemented prefabs for classroom items
 - Several possibilities for use of prefabs
 - Custom objects can also be created as desired

Scene View:



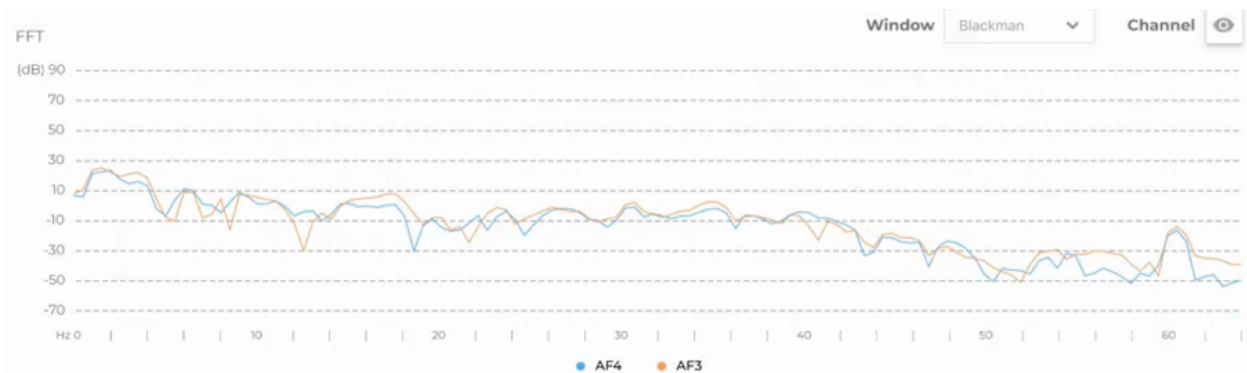
Game View:



- Conclusions:
 - Using different scenes allows us to create multiple modules for interactive learning
 - This design for the classroom has capabilities as a main menu
 - Using a text mesh can create a dialogue on the chalkboard that can be used to select modules
 - Environment design is relatively straight forward. The creation of custom objects and displaying proper lighting conditions provide the biggest issues with development
 - Leap Motion POV allows for easy implementation with Leap Motion software. The camera angle shown in the game view above will be loaded with a virtual set of hands mapped to the Leap Motion.

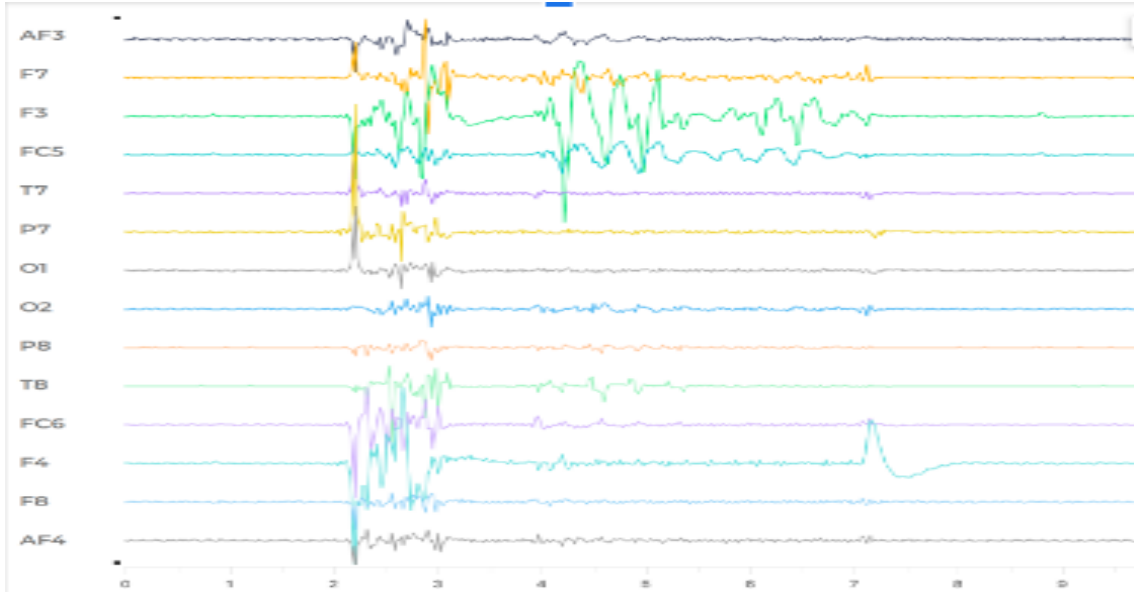
Task #6) Epoc+ Testing and Data Gathering with Emotivepro

- Objective: Obtain brain signal data while conducting various tasks
 - Understand how to position EMOTIV EPOC+ sensors
 - Familiarize ourselves with the capabilities of EMOTIV software
 - Experiment with extracting raw data
- Testing:
 - Playing a game
 - Provides example wave patterns for a relaxed and concentrated state of mind
 - Reading/Solving Math Problems
 - Provides example wave patterns for concentration and potential stress
 - Reading with Distractions
 - Provides example wave patterns for losing concentration when focussing on a task



AF3 and AF4 Sensors vs Time. Taken while playing a game.

- Emotive Pro Software:
 - Gives access to numerical data instead of only graphical data
 - Can give data from every sensor at one time
 - Allows for calculations of average and standard deviation of signals over the duration of the scan.



Brain Signal Waveforms from Every Sensor

time	EEG.Count	EEG.interp	EEG.AF3	EEG.F7	EEG.F3	EEG.FC5	EEG.T7	EEG.P7	EEG.O1	EEG.O2	EEG.P8	EEG.TB	EEG.FC6	EEG.F4	EEG.FB	EEG.AF4
2.6E+09	87	0	4155	4150.256	4184.615	4348.077	4171.539	4170.256	4175.256	4184.487	4189.359	4189.103	4170.256	4188.205	4202.051	4183.59
2.6E+09	88	0	4150.385	4142.308	4179.231	4349.231	4168.461	4170	4167.436	4180.641	4179.872	4182.308	4166.539	4180.385	4193.59	4182.564
2.6E+09	89	0	4150.128	4143.59	4183.333	4350.128	4167.692	4169.231	4171.154	4180.513	4179.615	4182.051	4165.385	4181.539	4196.026	4183.846
2.6E+09	90	0	4157.436	4154.615	4189.231	4354.359	4177.82	4172.949	4184.744	4186.539	4196.41	4197.949	4170.897	4197.308	4208.333	4190.385
2.6E+09	91	0	4158.718	4154.487	4183.333	4351.923	4178.59	4175.513	4178.974	4187.564	4193.461	4198.461	4171.795	4195.256	4204.359	4188.846
2.6E+09	92	0	4151.923	4147.436	4179.615	4347.564	4167.18	4173.333	4169.872	4180.897	4177.564	4183.205	4165.897	4177.82	4194.487	4177.308
2.6E+09	93	0	4150.513	4151.923	4182.308	4350	4171.923	4169.487	4180.769	4180.128	4182.692	4182.564	4167.18	4177.436	4197.436	4177.82
2.6E+09	94	0	4153.077	4155.256	4180.641	4347.051	4180.385	4172.18	4191.795	4183.077	4186.667	4186.154	4167.18	4183.461	4193.59	4183.461
2.6E+09	95	0	4153.205	4151.41	4182.564	4339.487	4175.256	4174.103	4177.949	4181.539	4177.564	4180.513	4164.103	4182.18	4184.487	4178.59
2.6E+09	96	0	4159.359	4154.103	4190.385	4342.051	4176.233	4166.667	4159.231	4173.333	4181.539	4182.564	4168.974	4186.026	4194.231	4180.128
2.6E+09	97	0	4163.59	4160.513	4185.769	4341.41	4182.692	4162.18	4160.128	4168.205	4187.308	4188.461	4173.846	4190.256	4205.128	4190.128
2.6E+09	98	0	4156.41	4157.692	4177.949	4327.308	4179.872	4169.487	4169.487	4177.051	4180.385	4187.18	4170.513	4185.897	4201.923	4190.641
2.6E+09	99	0	4151.026	4152.051	4179.487	4320.256	4177.949	4174.872	4168.846	4182.692	4176.41	4185.256	4167.564	4184.359	4196.154	4181.154
2.6E+09	100	0	4151.795	4154.103	4176.923	4314.231	4172.436	4165.385	4157.18	4173.077	4174.487	4182.949	4169.744	4182.051	4192.051	4176.923
2.6E+09	101	0	4150.641	4154.231	4173.077	4297.82	4164.744	4161.026	4152.18	4166.026	4167.692	4181.026	4167.051	4177.436	4188.718	4180
2.6E+09	102	0	4151.41	4153.333	4177.308	4291.795	4173.974	4171.154	4162.949	4170.256	4170.513	4184.487	4165.256	4179.744	4192.564	4184.744
2.6E+09	103	0	4162.692	4161.923	4181.795	4296.282	4182.564	4172.564	4173.718	4171.539	4183.205	4188.718	4170.385	4190.385	4201.154	4188.974
2.6E+09	104	0	4171.539	4167.949	4181.795	4289.359	4176.667	4168.59	4175.641	4173.461	4187.436	4189.359	4174.231	4194.744	4201.923	4191.667
2.6E+09	105	0	4170.513	4166.154	4183.846	4278.974	4175.897	4171.154	4175.513	4181.795	4186.795	4192.949	4175.897	4196.026	4199.615	4195.841
2.6E+09	106	0	4167.692	4161.923	4183.461	4275.769	4176.154	4167.564	4173.59	4184.103	4187.82	4198.846	4175.769	4197.692	4201.41	4192.692
2.6E+09	107	0	4167.82	4159.487	4177.82	4267.18	4171.026	4168.59	4169.744	4177.82	4187.692	4198.333	4172.436	4190.385	4201.154	4187.564
2.6E+09	108	0	4169.615	4160.385	4180.256	4263.539	4175.385	4171.667	4174.359	4174.872	4186.923	4196.282	4170.769	4182.436	4195.313	4190.769
2.6E+09	109	0	4169.872	4164.744	4187.564	4263.205	4181.923	4176.795	4176.154	4180.256	4184.615	4194.359	4173.461	4186.026	4193.846	4194.872
2.6E+09	110	0	4168.103	4162.564	4185.641	4255.641	4174.615	4173.077	4169.231	4180.256	4181.539	4191.026	4173.59	4191.154	4198.718	4190.128
2.6E+09	111	0	4168.231	4160.256	4180	4247.692	4172.308	4172.564	4176.282	4179.231	4183.846	4190.513	4170.897	4187.82	4200.769	4189.231
2.6E+09	112	0	4165.897	4160.641	4179.487	4247.308	4172.564	4172.436	4182.436	4180.128	4187.564	4188.59	4171.795	4185.128	4194.487	4190.641
2.6E+09	113	0	4161.539	4156.667	4176.667	4243.59	4167.051	4169.103	4170.769	4177.308	4183.846	4182.949	4171.539	4186.154	4190.256	4184.359
2.6E+09	114	0	4165.769	4158.59	4178.846	4243.205	4174.615	4169.103	4164.231	4175.385	4183.461	4183.461	4170.513	4186.923	4192.82	4184.231
2.6E+09	115	0	4173.333	4166.282	4187.949	4248.718	4187.18	4170.513	4170.513	4176.795	4187.18	4187.436	4169.103	4190.385	4191.539	4191.41
2.6E+09	116	0	4171.667	4163.333	4188.461	4248.59	4184.103	4169.359	4173.333	4172.436	4180.641	4181.282	4168.205	4191.795	4186.333	4191.41
2.6E+09	117	0	4168.718	4155.769	4185.769	4246.154	4178.846	4168.974	4169.744	4168.077	4173.205	4177.436	4168.974	4180	4184.231	4187.18
2.6E+09	118	0	4169.359	4157.308	4186.923	4245.256	4178.205	4174.103	4171.667	4178.718	4178.846	4183.461	4169.487	4189.103	4184.487	4189.231
2.6E+09	119	0	4168.59	4159.744	4182.051	4244.103	4175	4176.795	4172.18	4186.026	4190	4187.308	4169.103	4189.359	4184.487	4188.846
Working Data	EEG.AF3	EEG.F7	EEG.F3	EEG.FC5	EEG.T7	EEG.P7	EEG.O1	EEG.O2	EEG.P8	EEG.TB	EEG.FC6	EEG.F4	EEG.FB	EEG.AF4		
Average of Data	4186.1939	4173.60065	4174.29136	4188.92033	4173.91447	4167.03363	4171.45904	4169.63611	4171.97037	4171.47898	4163.84031	4173.68632	4161.90743	4172.44125		
Std. Dev	32.7370644	46.3543029	10.6259435	107.271927	12.7559304	13.0468561	15.6983844	21.4624505	28.26517	41.7259209	44.472743	31.0268126	98.9666371	28.3951943		

Corresponding Waveform Numerical Data from Test Above

- **Conclusions:**
 - The sensor is very specific and requires a good amount of saline solution for proper connections
 - Software allows viewing of all channels at once. It also allows viewing of individual channels for specific sensor monitoring
 - Having pictures of the brainwave graphs can be useful for monitoring from an outside perspective. However, the numerical data will be the most useful for implementation in learning module software.

Testing plan

- **Test #1 (Get Brainwave Information from EMOTIV EPOC+)**
 - Goal: Understand sensor placement and brain wave patterns from different stimuli
 - System Components: EMOTIV EPOC+, EMOTIV PRO software
 - Testing Process:
 - Configure the EPOC+ with sensor placements in the correct place. The proper placement is demonstrated through green indication in EMOTIV software.
 - Performance of tasks related to situations students will be subjected to when using our software
 - Data collection from EPOC+ in real-time alongside screen recording of the desired task
 - Data Processing and Visualization: Raw data will be displayed in graphical and numerical forms. This allows for the calculation of wave amplitudes at specific points in the video. Also, graphical representation allows for demonstration of patterns.
 - Evaluation: Focus on graph characteristics that show differences from steady state wave patterns, as well as differences in patterns in unique tasks.

- **Test #2 (Evaluation of Controller Capabilities):**
 - Goal: Understand how our created controller can be used to interact with a virtual environment.
 - System Components: Raspberry Pi, ADXL345, Push Buttons
 - Testing Process:
 - Push-button activated at different times. Observe response from Raspberry Pi.
 - Create a program that takes accelerometer data and stores it into Raspberry Pi registers
 - Map obtained data into a designated range
 - Data Processing and Visualization: Data will be observed within the terminal. A button press should show the programmed response on screen, while the ADXL should output mapped positional data at a The software designated frequency.
 - Evaluation: The main focus for this testing is understanding the ADXL data. Calibrating values to degrees of tilt will allow for a cursor to be moved across the screen.
 - Note: With this information, a cursor can be moved at one set speed when the threshold value of the accelerometer passes a specific mapped value. Cursor acceleration can also be implemented using different angular ranges

- **Test #3 (Understand Capabilities of Leap Motion Controller)**
 - Goal: Understand capabilities of Leap Motion
 - System Components: Leap Motion, PC, Leap Motion compatible software
 - Testing Process:
 - Play premade games to understand the ability to distinguish premade hand signals
 - Enter premade environments to understand virtual hand collision box
 - Data Collection: No numerical data is obtained from this test. This experiment allows us to understand which controller may be best suited for interaction with the virtual environment
 - Evaluation:
 - Focus on mapping of joints that Leap Motion provides
 - Understand which hand signals can be used. This can allow for item selection and options menu to be mapped for specific hand shapes
 - Interaction with the virtual environment allows the user to understand the capability of Leap Motion. When creating our program, we can model human interactions based on this information,

- **Test #4 (Understand Capabilities of Unity Software)**
 - Goal: Have group members learn the basics of Unity and understand possibilities for environment creation with this software
 - System Components: PC, Unity Software
 - Testing Process:
 - Create a closed classroom setting. Requires implementation of shape creation and custom lighting
 - Model preset shapes into items. Save shapes as prefabs for use across the entire project
 - Experimentation with texture creation to add color and features to created shapes
 - Implementation of prefabs created by other users. Learn to implement premade objects into the environment
 - Implement object motion and Unity physics
 - Evaluation: Upon completion of this experiment, the user should understand the basics of scene creation in Unity
 - Note: Understanding the capabilities of Unity will allow for realistic expectations when creating classroom models. Understanding and learning software at this point in the project will allow for efficient environment creation when implementing our learning modules.

- **Test 5 (Implementation of Controller and Environment Together)**
 - Goal: Allow for our chosen controller option to interact with a created unity environment
 - System Components: Leap Motion, Custom Controller, Unity, PC
 - Testing Process:
 - Connect the controller into the virtual environment
 - Use controller to interact with different items and trigger in-game events
 - Open settings menu at any point during environmental interaction
 - Evaluation: The controller should be able to control every aspect within the virtual environment. This should allow for the user to complete the module with no inputs besides the controller.

Appendix C: Software printout

I. Matlab

For quaternions:

```

%%ECE-493-001
%VR learning environment with real time brain signal monitoring
clear
opengl software
quat = importdata('quaterniondata2.xlsx'); %insert filename of data
q0 = quat.data(:,1); % grabbing column vectors for quaternion data
q1 = quat.data(:,2); %column 2
q2 = quat.data(:,3); %column 3
q3 = quat.data(:,4); %column 3
q = [q0 q1 q2 q3];
%creating time vector
T = 1/64;
t = (0:810-1)*T; %creates time vector based off of EPOC+ sampling rate
% yaw, pitch, roll angle Vector
[yaw, pitch, roll] = quat2angle([q0 q1 q2 q3]);%converts quaternion data to yaw pitch and roll
yawangle = (180/pi)*yaw;
pitchangle = (180/pi)*pitch;
rollangle = (180/pi)*roll;
% Rotation angle Vector
eul = quat2eul([q0 q1 q2 q3]);
XAngle = eul(:,1)*(180/pi);
YAngle = eul(:,2)*(180/pi);
ZAngle = eul(:,3)*(180/pi);
plotting data
figure;
plot(t,rollangle, t, pitchangle, t, yawangle);
title('Quaternion Angles Overtime');
xlabel('time(seconds)');
ylabel('Angle(degrees)');
legend('Roll Angle', 'Pitch Angle', 'Yaw Angle');
angles = [XAngle'; YAngle'; ZAngle'];%data
for ii=1:1261
quiver3(0,0,0,angles(1,ii),angles(2,ii),angles(3,ii)) %plot arrow
axis([-180 180 -90 90 -180 180]) %just for convenience
pause(.1)
End

```

For performance metrics:

```

clear
opengl software
pm = importdata('performace metric.csv'); %insert filename of data
engage = pm.data(:,1)*100; % grabbing column vectors for pm data
excite= pm.data(:,2)*100; %column 2
stress = pm.data(:,3)*100; %column 3
relax = pm.data(:,4)*100; %column 4
interest = pm.data(:,5)*100; %column 5
focus= pm.data(:,6)*100; %column 6
T = 1/0.1;
t = (0:17-1)*T;
figure;
plot(t, engage, t, excite, t, stress, t, relax,t, interest, t, focus);
title('Performance Metrics overtime');
xlabel('time(seconds)');
ylabel('Metric');
legend('Engagement', 'Excitement', 'Stress', 'Relaxation','Interest','Focus');

```

II. Unity

Jumping Between Unity Scenes:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using VivoxUnity;
using MLAPI;
using MLAPI.Transports.UNET;

public class jumpScene : MonoBehaviour
{
    public void LoadScene(string sceneName)
    {
        NetworkManager.Singleton.StartClient();
        SceneManager.LoadScene(sceneName);
    }

    public void LoadSceneNoNet(string sceneName)
    {
        SceneManager.LoadScene(sceneName);
    }
}

```

GameManager.cs:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement; //Added for jumping between scenes to
test static variables
using UnityEngine.UI; //Used for invoke

public class gameManager : MonoBehaviour
{
    // Start is called before the first frame update
    private string equation;
    public int num1, num2, operation; //Public to allow switching from
teacher hub

```

```

public int startOp, endOp, teacherControl;
private int curCount, box, answer; //No modification
private char[] op = new char[2] { '+', '-' };
static public int completed = 0;
static public int correct = 0;

public enum Scene
{
    Equation_Saber,
}

void Start()
{
    operation = Random.Range(0, 2);

    if (operation == 0) //Addition
    {
        num1 = Random.Range(1, 10);
        num2 = Random.Range(1, 10);
        answer = num1 + num2;
    }
    else if (operation == 1)
    {
        num1 = Random.Range(11, 20);
        num2 = Random.Range(1, 11);
        answer = num1 - num2;
    }

    curCount = 0;

    equation = num1.ToString() + " " + op[operation] + " " +
num2.ToString() + " = __.";
}

// Update is called once per frame
void Update()
{
    if (curCount < answer)
    {

```

```

        equation = num1.ToString() + " " + op[operation] + " " +
num2.ToString() + " = __.";
    }
    else if (curCount == answer)
    {
        equation = "Correct: " + num1.ToString() + " " +
op[operation] + " " + num2.ToString() + " = " + curCount.ToString() +
".";

        StartCoroutine(Repeat(Scene.Equation_Saber));

        //Before Coroutine add something saying if (completed ==
desiredNoOfRounds){App.Quit}; or jump to new scene with performance
metrics
    }
    else
    {
        equation = "Incorrect. Correct Answer is: " + num1.ToString()
+ " " + op[operation] + " " + num2.ToString() + " = " + answer.ToString()
+ ".";

        StartCoroutine(Repeat(Scene.Equation_Saber));
        //Application.Quit();
    }

}

void OnGUI()
{
    GUIStyle fontS = new GUIStyle();
    fontS.fontSize = 50;
    fontS.normal.textColor = Color.blue;
    //GUI.contentColor = Color.blue;
    GUI.Label(new Rect(0, 0, 200, 200), equation + "\n" + "Current
Value: " + curCount + "\n" + "Score: " + correct + "/" + completed,
fontS);
}

public void updateCurrent(int rayVal, int op)
{
    if (op == 0)
    {
        if (curCount > answer)
        {

```

```

        curCount = rayVal; //Resets on going over value...
Temporary until scene manager is created
    }
    else if (curCount == answer)
    {
        curCount = answer;
    }
    else
    {
        curCount += rayVal;
    }
}
else
{
    curCount -= rayVal;
}
Debug.Log("count = " + curCount + " val = " + rayVal + " op = " +
op);
}

public IEnumerator Repeat(Scene scene)
{
    yield return new WaitForSeconds(2f);

    completed++;

    if (completed > 2)
    {
        Application.Quit(); //This works on Build and Run
    }

    if (curCount == answer)
    {
        correct++;
    }

    SceneManager.LoadScene(scene.ToString()); //Loads scene again
}
}

```

secondCameraMovement.cs:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

```



```

using System;
using EmotivUnityPlugin;

public class secondCameraMovement : MonoBehaviour
{
    public float moveSpeed = 5000f;

    float rotationX = 0F;
    float epocX;
    float epocY;

    private Channel_t xDirection = Channel_t.CHAN_Q0;
    private Channel_t yDirection = Channel_t.CHAN_Q1;
    private Channel_t zDirection = Channel_t.CHAN_Q2;
    private Channel_t wDirection = Channel_t.CHAN_Q3;
    double[] intx, inty, intz, intw;

    DataStreamManager ds = DataStreamManager.Instance;

    void Start()
    {
        intx = ds.GetMotionData(xDirection);
        inty = ds.GetMotionData(yDirection);
        intz = ds.GetMotionData(zDirection);
        intw = ds.GetMotionData(wDirection);
    }

    double[] inX, inY, inZ, inW;
    float inputX, inputY, inputZ, inputW;
    float smoot = 5.0f;
    // Update is called once per frame
    void Update()
    {
        inX = ds.GetMotionData(xDirection);
        inY = ds.GetMotionData(yDirection);
        inZ = ds.GetMotionData(zDirection);
        inW = ds.GetMotionData(wDirection);

        if (inZ != null && inZ.Length > 0 && inW != null && inW.Length >
0 && inX != null && inX.Length > 0 && inY != null && inY.Length > 0)
        {
            Quaternion cameraRotation = new Quaternion((float) (inX[0]) ,
(float) (inY[0]), (float) (inZ[0]), (float) (inW[0]));
            var qEul = cameraRotation.eulerAngles;

```

```

        Quaternion tempEul = new Quaternion((float)intx[0],
(float)inty[0], (float)intz[0], (float)intw[0]);
        var temp = tempEul.eulerAngles;
        Quaternion corrected = Quaternion.Euler(qEul.z - temp.z, 0,
0);

        Quaternion yaw = Quaternion.Euler(0, qEul.x - temp.x, 0);

        transform.rotation = yaw;
        transform.Rotate(transform.rotation.x, transform.rotation.y +
270, transform.rotation.x);
    }
}
}

```

CSVReadWrite.cs:

```

using System.Collections;
using System.Collections.Generic;
using System.Text;
using System.IO;
using UnityEngine;
using EmotivUnityPlugin;

public class CsvReadWrite : MonoBehaviour
{

    private List<string[]> rowData = new List<string[]>();
    public Vector3 lastposition;

    DataStreamManager ds = DataStreamManager.Instance;

    void Start()
    {
        lastposition = transform.position;

        string[] rowDataTemp = new string[10];

        rowDataTemp[0] = "Stress Level";
        rowDataTemp[1] = "Engagement Level";
        rowDataTemp[2] = "Interest Level";
        rowDataTemp[3] = "Excitement Level";
        rowDataTemp[4] = "Focus Level";
        rowDataTemp[5] = "Relaxation Level";

        rowDataTemp[6] = "Q0";
    }
}

```

```

        rowDataTemp[7] = "Q1";
        rowDataTemp[8] = "Q2";
        rowDataTemp[9] = "Q3";

        rowData.Add(rowDataTemp);
    }

    void Update()
    {
        Save();
    }

    void Save()
    {
        string[] rowDataTemp = new string[10];

        rowDataTemp[0] = "0";
        rowDataTemp[1] = "0";
        rowDataTemp[2] = "0";
        rowDataTemp[3] = "0";
        rowDataTemp[4] = "0";
        rowDataTemp[5] = "0";

        rowDataTemp[6] =
ds.GetMotionData(Channel_t.CHAN_Q0)[0].ToString();
        rowDataTemp[7] =
ds.GetMotionData(Channel_t.CHAN_Q1)[0].ToString();
        rowDataTemp[8] =
ds.GetMotionData(Channel_t.CHAN_Q2)[0].ToString();
        rowDataTemp[9] =
ds.GetMotionData(Channel_t.CHAN_Q3)[0].ToString();

        //rowDataTemp[2] = transform.position.z.ToString();

        rowData.Add(rowDataTemp);

        string[][] output = new string[rowData.Count][];

        for (int i = 0; i < output.Length; i++)
        {
            output[i] = rowData[i];
        }

        int length = output.GetLength(0);
        string delimiter = ",";

```

```

        StringBuilder sb = new StringBuilder();

        for (int index = 0; index < length; index++)
            sb.AppendLine(string.Join(delimiter, output[index]));

        string filePath = getPath();

        StreamWriter outputStream = System.IO.File.CreateText(filePath);
        outputStream.WriteLine(sb);
        outputStream.Close();
    }

    private string getPath()
    {
        return Application.dataPath + "/CSV/" + "Saved_data.csv";
    }
}

```

rayCastTest.cs:

```

using Leap;
using Leap.Unity;
using System.Collections.Generic;
using UnityEngine;

public class rayCastTest : MonoBehaviour
{
    Hand leapHand;
    FingerModel finger;
    HandModel handModel;
    public LayerMask layer;

    // Use this for initialization
    void Start()
    {
        /*Controller controller = new Controller();
        Frame frame = controller.Frame(); // controller is a Controller
object
        if (frame.Hands.Count > 0)
        {
            List<Hand> hands = frame.Hands;
            leapHand = hands[0];
        }

```

```

        finger = leapHand.Fingers;
        Finger.FingerType fingerType = finger.Type;*/
        handModel = GetComponent<HandModel>();
        leapHand = handModel.GetLeapHand();
        if (leapHand == null) Debug.LogError("No leap_hand founded");
    }

    // Update is called once per frame
    void Update()
    {
        Vector3 fwd = transform.TransformDirection(Vector3.right);
        RaycastHit hit;

        for (int i = 0; i < HandModel.NUM_FINGERS; i++)
        {
            finger = handModel.fingers[i];
            if (Physics.Raycast(finger.GetTipPosition(), fwd, out hit,
10, layer))
            {
                //Send the value of the gameObject to GameManger for
                incrementing
                gameManager gmTemp;
                int temp =
                hit.transform.gameObject.GetComponent<boxMovement>().getValue();
                int tempOp =
                hit.transform.gameObject.GetComponent<boxMovement>().getOp();
                Debug.Log(" op in raycast = " + tempOp);
                gmTemp = GameObject.FindObjectOfType<gameManager>();
                Debug.Log(temp.ToString());
                gmTemp.updateCurrent(temp, tempOp);
                Destroy(hit.transform.gameObject);
                break;
            }
            Debug.DrawRay(finger.GetTipPosition(),
            finger.GetRay().direction, Color.red, Time.deltaTime, true);
        }
    }
}

```

Spawner_Script.cs:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

```

```

public class Spawner_Script : MonoBehaviour
{
    public GameObject[] cubes;
    public Transform[] points;
    public float beat = (60 / 130);
    private float timer;
    public double speed;
    public int pause;

    // Start is called before the first frame update
    void Start()
    {
        speed = 2;
        pause = 0;
    }

    // Update is called once per frame
    void Update()
    {
        if (timer > beat)
        {
            if (pause == 0)
            {
                int cubeOp = Random.Range(0, 2);
                GameObject cube = Instantiate(cubes[cubeOp],
points[Random.Range(0, 4)]);
                cube.GetComponent<boxMovement>().setOp(cubeOp);
                //cube.GetComponent<boxMovement>().setSpeed(speed);
                cube.transform.localPosition = Vector3.zero;
                //cube.transform.Rotate(transform.right, 90 *
Random.Range(0, 4));

                }
            timer -= beat;
        }
        timer += Time.deltaTime;
    }

    public void setPause()
    {
        pause = 1;
    }
    public void setStart()
    {
        pause = 0;
    }
}

```

```

    }
    public void togglePause()
    {
        if (pause == 0)
            pause = 1;
        else
            pause = 0;
    }
    public void setSpeed(double x)
    {
        if (x > 0)
            speed = x;
        else
            speed = 2;
    }
}

```

SettingsMenu.cs:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using EmotivUnityPlugin;

public class SettingsMenu : MonoBehaviour
{
    // values to move to teacher hub
    // motion data and performance metric data
    private DataStreamManager ds = DataStreamManager.Instance;
    private static double[] emptyDoub = { };
    public double[] pmData;
    public GameObject cube;
    public double oldSpeed, newSpeed;
    // Start is called before the first frame update
    void Start()
    {
        pmData = new double[7];
        oldSpeed = 2;
        newSpeed = 2;
    }

    // Update is called once per frame
    void Update()
    {
        if (oldSpeed != newSpeed)
        {

```

```

        Debug.Log(pmData[0] + pmData[1] + pmData[2] + pmData[3] +
pmData[4] + "PM data");
        cube.GetComponent<boxMovement>().setSpeed((float)newSpeed);
        oldSpeed = newSpeed;
        getPMDataList();
        Debug.Log(pmData);
    }

}

public double[] getPMDataList()
{
    double[] dat = new double[7];
    int counter = 0;
    dat[0] = ds.GetPMDData("eng");
    dat[1] = ds.GetPMDData("exc");
    dat[2] = ds.GetPMDData("lex");
    dat[3] = ds.GetPMDData("str");
    dat[4] = ds.GetPMDData("rel");
    dat[5] = ds.GetPMDData("int");
    dat[6] = ds.GetPMDData("foc");
    return dat; // check if empty
}

public void setSpeed(int x)
{
    newSpeed = x;
}
}

```

setText.cs:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class textSet : MonoBehaviour
{
    // Start is called before the first frame update

    private TextMeshProUGUI test;
    int score;
    //Calls boxMovement script and uses getter to obtain value to display
as text
    private boxMovement bmTemp;

```



```

void Start()
{
    test = FindObjectOfType<TextMeshProUGUI>();
    bmTemp = GameObject.FindObjectOfType<boxMovement>();
}

// Update is called once per frame
void Update()
{
    score = bmTemp.getValue();
    test.SetText(score.ToString());
}
}

```

boxMovement:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class boxMovement : MonoBehaviour
{
    static public float speed; //Speed of the box movement
    private int value; //Holds a random value for box
    private int op; // operation of the box -- 0 = add 1 = sub

    // Start is called before the first frame update
    void Start()
    {
        //speed = 2.0f;
        Object.Destroy(gameObject, 25f);
        value = Random.Range(1, 10);
        //op = 0;
    }

    // Update is called once per frame
    void Update()
    {
        transform.position += Time.deltaTime * transform.forward * speed;
    }
}

```

```

public float getSpeed()
{
    return speed;
}

public void setSpeed(float x)
{
    speed = x;
}

public int getValue()
{
    return value;
}
public void setOp(int x)
{
    op = x;
    //Debug.Log("operator = " + x);
}
public int getOp()
{
    return op;
}
}

```

EventPlayer.cs: (Moving variables across the Network)

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using MLAPI;
using VivoxUnity;
using MLAPI.Messaging;
using MLAPI.NetworkVariable;

public class EventPlayer : NetworkBehaviour
{
    public NetworkVariable<double> stress = new
NetworkVariable<double>(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone, ReadPermission =

```

```

NetworkVariablePermission.Everyone });
    public NetworkVariable<double> engagement = new
NetworkVariable<double>(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone, ReadPermission =
NetworkVariablePermission.Everyone });
    public NetworkVariable<double> focus = new
NetworkVariable<double>(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone, ReadPermission =
NetworkVariablePermission.Everyone });
    public NetworkVariable<double> interest = new
NetworkVariable<double>(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone, ReadPermission =
NetworkVariablePermission.Everyone });
    public NetworkVariable<double> relaxation = new
NetworkVariable<double>(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone, ReadPermission =
NetworkVariablePermission.Everyone });
    public NetworkVariable<double> excitement = new
NetworkVariable<double>(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone, ReadPermission =
NetworkVariablePermission.Everyone });
    public NetworkVariable<double> speed = new NetworkVariable<double>();
    private double[] pmData = new double[7];
    int randomVal;

void Start()
{
    randomVal = Random.Range(0, 100);
    if (NetworkManager.Singleton.IsClient)
    {
        pmData = FindObjectOfType<SettingsMenu>().pmData;

        for (int i = 0; i < pmData.Length; i++)
        {
            if (pmData[i] == -1)
            {
                continue;
            }
            if (i == 0)
            {
                engagement.Value = pmData[i];
            }
            if (i == 1)
            {
                excitement.Value = pmData[i];
            }
        }
    }
}

```

```

    }
    if (i == 2)
    {

    }
    if (i == 3)
    {
        stress.Value = pmData[i];
    }
    if (i == 4)
    {
        relaxation.Value = pmData[i];
    }
    if (i == 5)
    {
        interest.Value = pmData[i];
    }
    if (i == 6)
    {
        focus.Value = pmData[i];
    }
    }
    FindObjectOfType<SettingsMenu>().newSpeed = speed.Value;
}
}

// Update is called once per frame
void Update()
{
    pmData = FindObjectOfType<SettingsMenu>().getPMDataList();

    for (int i = 0; i < pmData.Length; i++)
    {
        if (pmData[i] == -1)
        {
            continue;
        }
        if (i == 0)
        {
            engagement.Value = pmData[i];
        }
        if (i == 1)
        {
            excitement.Value = pmData[i];
        }
    }
}

```

```

        if (i == 2)
        {

        }
        if (i == 3)
        {
            stress.Value = pmData[i];
        }
        if (i == 4)
        {
            relaxation.Value = pmData[i];
        }
        if (i == 5)
        {
            interest.Value = pmData[i];
        }
        if (i == 6)
        {
            focus.Value = pmData[i];
        }
    }
    FindObjectOfType<SettingsMenu>().newSpeed = speed.Value;
}
}

```

MenuScript.cs:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;
using MLAPI;
using MLAPI.Transports.UNET;
using VivoxUnity;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class MenuScript : MonoBehaviour
{
    public GameObject menuPanel;
    public GameObject mainPanel;
    public InputField inputField, HostField;
    VivoxUnity.Client client = new VivoxUnity.Client();
    string student1Channel, student2Channel,
    student3Channel, playerName;
}

```

```

VivoxVoiceManager vivox;
private void Start()
{
    mainPanel.SetActive(false);
    menuPanel.SetActive(true);
}

private void Update()
{
    List<MLAPI.Connection.NetworkClient> clients =
NetworkManager.Singleton.ConnectedClientsList;
    if (clients.Count == 1)
    {
        var player =
clients[0].PlayerObject.GetComponent<EventPlayer>();
    }
    else if (clients.Count == 2)
    {
        var player =
clients[0].PlayerObject.GetComponent<EventPlayer>();
        player =
clients[1].PlayerObject.GetComponent<EventPlayer>();
    }
}

public void Host()
{
    vivox = VivoxVoiceManager.Instance;
    vivox = VivoxVoiceManager.Instance;
    client.Uninitialize();
    client.Initialize();
    playerName = "teacher";
    vivox.Login(playerName);
    NetworkManager.Singleton.StartServer();
    menuPanel.SetActive(false);
    mainPanel.SetActive(true);
}

public void Join()
{
    //clicked join
    if (inputField.text.Length <= 0)
    {
NetworkManager.Singleton.GetComponent<UNetTransport>().ConnectAddress =

```

```

"52.147.204.115";
        }
        else
        {

NetworkManager.Singleton.GetComponent<UNetTransport>().ConnectAddress =
inputField.text;
        }
        NetworkManager.Singleton.StartClient();
        menuPanel.SetActive(false);
        mainPanel.SetActive(true);
    }

    public void SceneChange()
    {
        mainPanel.SetActive(false);
    }

    public void joinChannel1()
    {
        vivox.JoinChannel("student0_channel",
ChannelType.NonPositional, VivoxVoiceManager.ChatCapability.AudioOnly);
    }

    public void joinChannel2()
    {
        vivox.JoinChannel("student1_channel",
ChannelType.NonPositional, VivoxVoiceManager.ChatCapability.AudioOnly);
    }
}

```

EventManager.cs:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using MLAPI.NetworkVariable;
using MLAPI;

public class EventManager : MonoBehaviour
{
    public Slider stressBox, engagementBox, focusBox, interestBox,
relaxationBox, excitementBox, stressBox2, engagementBox2, focusBox2,

```

```

interestBox2, relaxationBox2, excitementBox2;
    // Start is called before the first frame update
    private Color green = Color.green;
    public NetworkVariableDouble stressnet = new
NetworkVariableDouble(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone });
    public NetworkVariable<double> engagementNet = new
NetworkVariable<double>(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone });
    public NetworkVariable<double> focusNet = new
NetworkVariable<double>(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone });
    public NetworkVariable<double> interestNet = new
NetworkVariable<double>(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone });
    public NetworkVariable<double> relaxationNet = new
NetworkVariable<double>(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone });
    public NetworkVariable<double> excitementNet = new
NetworkVariable<double>(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone });
    public NetworkVariable<double> stressnet2 = new
NetworkVariable<double>(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone });
    public NetworkVariable<double> engagementNet2 = new
NetworkVariable<double>(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone });
    public NetworkVariable<double> focusNet2 = new
NetworkVariable<double>(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone });
    public NetworkVariable<double> interestNet2 = new
NetworkVariable<double>(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone });
    public NetworkVariable<double> relaxationNet2 = new
NetworkVariable<double>(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone });
    public NetworkVariable<double> excitementNet2 = new
NetworkVariable<double>(new NetworkVariableSettings { WritePermission =
NetworkVariablePermission.Everyone });
    public NetworkVariable<double> speed = new NetworkVariable<double>();
    public NetworkVariable<double> speed2 = new
NetworkVariable<double>();

    private Color red = Color.red;

```



```
public Image Fill1,Fill2, Fill3, Fill4, Fill5, Fill6, Fill7, Fill8,
Fill9, Fill10,Fill11,Fill12;

void Start()
{
    speed.Value = 2.0;
    speed2.Value = 2.0;
}

public void IncreaseSpeed()
{
    speed.Value += .2f;
    if (speed.Value > 2.5f)
    {
        speed.Value = 2.5f;
    }
}

public void DecreaseSpeed()
{
    speed.Value -= .2f;
    if (speed.Value < .5f)
    {
        speed.Value = .5f;
    }
}

public void IncreaseSpeedtwo()
{
    speed.Value += .2f;
    if (speed.Value > 2.5f)
    {
        speed.Value = 2.5f;
    }
}

public void DecreaseSpeedtwo()
{
    speed.Value -= .2f;
    if (speed.Value < .5f)
    {
        speed.Value = .5f;
    }
}

// Update is called once per frame
void Update()
{
    if (NetworkManager.Singleton.IsServer)
```

```

    {
        List<MLAPI.Connection.NetworkClient> clients =
NetworkManager.Singleton.ConnectedClientsList;
        for (int i = 0; i < clients.Count; i++)
        {
            Debug.Log(clients[i].PlayerObject.NetworkObjectId + "
client");
        }

        if (clients.Count == 1)
        {

            Debug.Log("Someone has joined");
            var player =
clients[0].PlayerObject.GetComponent<EventPlayer>();

            stressnet = player.stress;
            stressBox.value = (float)stressnet.Value;
            engagementNet = player.engagement;
            engagementBox.value = (float)engagementNet.Value;
            focusNet = player.focus;
            focusBox.value = (float)focusNet.Value;
            interestNet = player.interest;
            interestBox.value = (float)interestNet.Value;
            relaxationNet = player.relaxation;
            relaxationBox.value = (float)relaxationNet.Value;
            excitementNet = player.excitement;
            excitementBox.value = (float)excitementNet.Value;
            player.speed.Value = speed.Value;
            Debug.Log(player.speed.Value + " speed " +
player.stress.Value + " stress " + player.engagement.Value+ " engagement
");
        }
        if (clients.Count == 2)
        {
            var player =
clients[0].PlayerObject.GetComponent<EventPlayer>();
            stressnet = player.stress;
            stressBox.value = (float)stressnet.Value;
            engagementNet = player.engagement;
            engagementBox.value = (float)engagementNet.Value;
            focusNet = player.focus;
            focusBox.value = (float)focusNet.Value;
            interestNet = player.interest;
            interestBox.value = (float)interestNet.Value;

```

```

        relaxationNet = player.relaxation;
        relaxationBox.value = (float)relaxationNet.Value;
        excitementNet = player.excitement;
        excitementBox.value = (float)excitementNet.Value;
        player.speed.Value = speed.Value;
        player =
clients[1].PlayerObject.GetComponent<EventPlayer>();
        stressnet2 = player.stress;
        stressBox2.value = (float)stressnet2.Value;
        engagementNet2 = player.engagement;
        engagementBox2.value = (float)engagementNet2.Value;
        focusNet2 = player.focus;
        focusBox2.value = (float)focusNet2.Value;
        interestNet2 = player.interest;
        interestBox2.value = (float)interestNet2.Value;
        relaxationNet2 = player.relaxation;
        relaxationBox2.value = (float)relaxationNet2.Value;
        excitementNet2 = player.excitement;
        excitementBox2.value = (float)excitementNet2.Value;
        player.speed.Value = speed2.Value;
    }
    Fill11.color = Color.Lerp(green, red, stressBox.value);
    Fill12.color = Color.Lerp(red, green, engagementBox.value);
    Fill13.color = Color.Lerp(red, green, focusBox.value);
    Fill14.color = Color.Lerp(red, green, interestBox.value);
    Fill15.color = Color.Lerp(red, green, relaxationBox.value);
    Fill16.color = Color.Lerp(red, green, excitementBox.value);
    Fill17.color = Color.Lerp(green, red, stressBox2.value);
    Fill18.color = Color.Lerp(red, green, engagementBox2.value);
    Fill19.color = Color.Lerp(red, green, focusBox2.value);
    Fill10 .color = Color.Lerp(red, green, interestBox2.value);
    Fill11.color = Color.Lerp(red, green, relaxationBox2.value);
    Fill12.color = Color.Lerp(red, green, excitementBox2.value);
}

}

}

```

III.Cortex App API

```

Get cortex Info:
{"id":1,"jsonrpc":"2.0","method":"getCortexInfo"}

```

Get User login:

```
{
  "id": 1,
  "jsonrpc": "2.0",
  "method": "getUserLogin"
}
```

Approve Client:

```
{
  "id": 1,
  "jsonrpc": "2.0",
  "method": "requestAccess",
  "params": {
    "clientId": "xxx",
    "clientSecret": "xxx"
  }
}
```

Check If access granted:

```
{
  "id": 1,
  "jsonrpc": "2.0",
  "method": "hasAccessRight",
  "params": {
    "clientId": "xxx",
    "clientSecret": "xxx"
  }
}
```

Get new corext token:

```
{
  "id": 1,
  "jsonrpc": "2.0",
  "method": "authorize",
  "params": {
    "clientId": "UxGJKhyNwU8Sv18ETanYUz0fSfQDiuIrd2YxNfu5",
    "clientSecret":
      "WKuBJFmjr2QE7Dna49UBGGZWy0Nv1SJqXki4Z76sdI9PsaPhVrTbyvvnK5azog32PVvBHNcC
      65dslbIDsTDY5o92AhVql56KCFiZ5LQ039oQCfz9cjGYWGrbQekiPIRz"
  }
}
```

```
getUserInformation:
{
  "id": 1,
  "jsonrpc": "2.0",
  "method": "getUserInformation",
  "params": {
    "cortexToken": "xxx"
  }
}
```

```
getLicenseInfo:
{
  "id": 1,
  "jsonrpc": "2.0",
  "method": "getLicenseInfo",
  "params": {
    "cortexToken": "xxx"
  }
}
```

```
Query headsets
{
  "id": 1,
  "jsonrpc": "2.0",
  "method": "queryHeadsets"
}
```

```
Update headset positioning:
{
  "id": 1,
  "jsonrpc": "2.0",
  "method": "updateHeadsetCustomInfo",
  "params": {
    "cortexToken": "xxx",
    "headbandPosition": "top",
    "headsetId": "EPOCX-12345678"
  }
}
```

```
To change settings:
{
  "id": "EPOCPLUS-3B9AXXXX",
  "status": "connected",
```

```

"connectedBy": "dongle",
"customName": "",
"dongle": "6ff",
"firmware": "625",
"motionSensors": [
  "GYROX",
  "GYROY",
  "GYROZ",
  "ACCX",
  "ACCY",
  "ACCZ",
  "MAGX",
  "MAGY",
  "MAGZ"
],
"sensors": [
  "AF3",
  "F7",
  "F3",
  "FC5",
  "T7",
  "P7",
  "O1",
  "O2",
  "P8",
  "T8",
  "FC6",
  "F4",
  "F8",
  "AF4"
],
"settings": {
  "eegRate": 256,
  "eegRes": 16,
  "memsRate": 64,
  "memsRes": 16,
  "mode": "EPOCPLUS"
}
}

```

Create Session:

```

{
  "id": 1,
  "jsonrpc": "2.0",
  "method": "createSession",
  "params": {
    "cortexToken":
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBzI6IjEiLCJ1b290Ijpmcm9udC51cmVudDM0NS5kYXR

```

```

hc3RyZWfTiiwiYXBwVmVyc2lvbiI6IjEuMCIsImV4cCI6MTYxMjc5NzY1NywibmJmIjoxNjEy
NTM4NDU3LCJlc2VySWQiOiJmZmI1NDRhZi05NzhkLTQ0MDYtODY4YS0yZDUyZGIwOGIxNTIiL
CJlc2VybmFtZSI6ImJyZW5kMzQ1IiwidmVyc2lvbiI6IjIuMCJ9.ntIgN4Y5RMhAmcqnM++vs
hNln2IPqOEVTRPyIQEZbz8=",
    "headset": "EPOCPLUS-4A2C0128",
    "status": "open"
  }
}

```

To export a csv:

```

{
  "id": 1,
  "jsonrpc": "2.0",
  "method": "createRecord",
  "params": {
    "cortexToken":
      "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBzI6ImNvbS5icmVudm0NS5kYXR
      hc3RyZWfTiiwiYXBwVmVyc2lvbiI6IjEuMCIsImV4cCI6MTYxMjc5NzY1NywibmJmIjoxNjEy
      NTM4NDU3LCJlc2VySWQiOiJmZmI1NDRhZi05NzhkLTQ0MDYtODY4YS0yZDUyZGIwOGIxNTIiL
      CJlc2VybmFtZSI6ImJyZW5kMzQ1IiwidmVyc2lvbiI6IjIuMCJ9.ntIgN4Y5RMhAmcqnM++vs
      hNln2IPqOEVTRPyIQEZbz8=",
    "session": "b2250bd9-cbb7-49a7-99db-a8115bf0c147",

    "title": "brendan"
  }
}

```

Subscribe:

```

{
  "id": 1,
  "jsonrpc": "2.0",
  "method": "subscribe",
  "params": {
    "cortexToken":
      "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhcHBzI6ImNvbS5icmVudm0NS5kYXR
      hc3RyZWfTiiwiYXBwVmVyc2lvbiI6IjEuMCIsImV4cCI6MTYxMjc5NzY1NywibmJmIjoxNjEy
      NTM4NDU3LCJlc2VySWQiOiJmZmI1NDRhZi05NzhkLTQ0MDYtODY4YS0yZDUyZGIwOGIxNTIiL
      CJlc2VybmFtZSI6ImJyZW5kMzQ1IiwidmVyc2lvbiI6IjIuMCJ9.ntIgN4Y5RMhAmcqnM++vs
      hNln2IPqOEVTRPyIQEZbz8=",
    "session": "3668e75c-d908-4e1e-9beb-0089da5e129a",
    "streams": ["eeg", "mot"]
  }
}

```