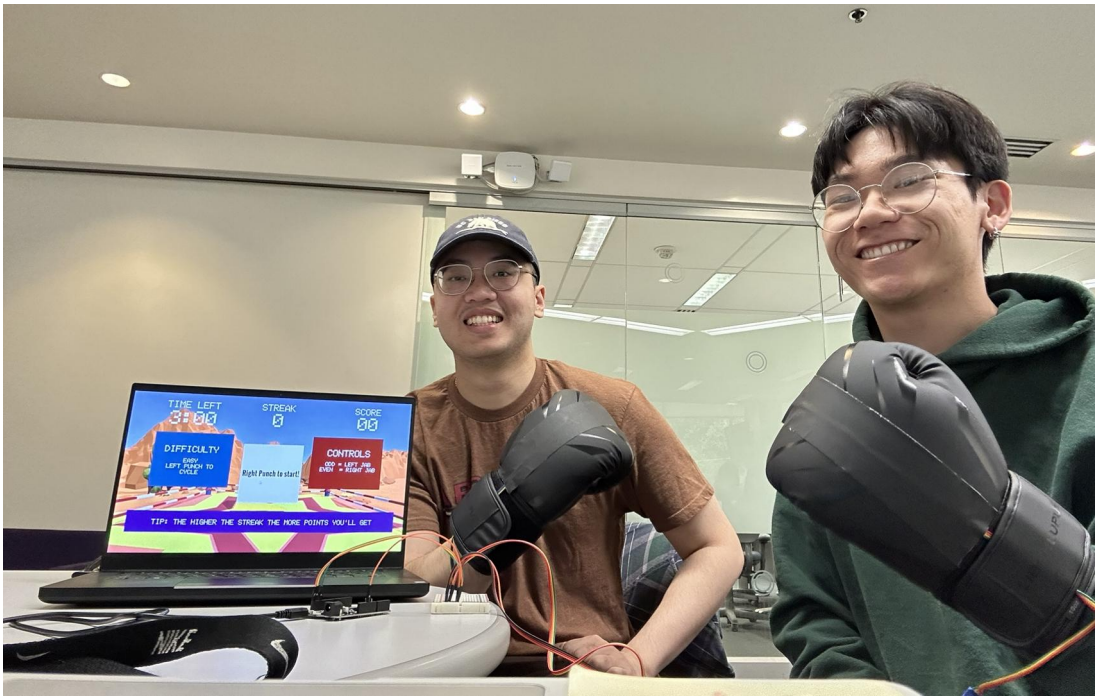

CSE176A/276D: Odd-Even Boxing

A boxing video game that is designed for patients with Parkinson's Disease to enable a safe way to get exercise while stimulating their minds. Odd-Even Boxing has players not only exercise but have them use different forms of cognition by having users think about math problems as well as what type of punch they need to throw.

Team Members | Cory Huynh - Led software development (Arduino IDE & Unity)

Jonathan Yip - Led hardware development & design

Stakeholder | Marty Acevedo

Code Links | Unity Code - Arduino IDE Code

1 | Executive Summary

Our stakeholder, Marty Acevedo, is a former dietitian who was diagnosed with Parkinson's Disease eight years ago. To manage her symptoms, Marty participates in exercise and cognitive stimulation through *New York Times* games like *Wordle*. After our first meeting with Marty, we wanted to design and implement a fun and easy way for people like Marty (who are also diagnosed with Parkinson's Disease) to get daily cognitive stimulation and exercise at the same time. After a few iterations of sketching, storyboarding, and paper prototyping, the project we committed to is Odd-Even Boxing: a game that combines math with the sport of boxing - a popular form of exercise amongst people with Parkinson's Disease. The project consists of two main components: creating the boxing gloves and developing the game. The boxing gloves use MPU6050 accelerometers to detect user punches and mimic keyboard inputs to communicate those punches. The game takes these keyboard inputs as controls to challenge the player with math problems to stimulate their mind. After completing our prototype, we tested the game with our stakeholder and a few others. While we did receive some feedback for potential improvements, the overall reception of our prototype was very positive. Our stakeholder Marty especially found that the game was fun, challenging, and very applicable to people with Parkinson's Disease like her. If given more time, we would have liked to implement Bluetooth and vibration motors within our controllers to provide for smoother gameplay. Otherwise, our project was a success and hopefully future students can expand on the base that we created.

2 | Introduction & Statement of Needs

Patients with Parkinson's Disease have symptoms such as tremors, slowed movements, rigid muscles, and loss of automatic movements (Mayo Clinic). After speaking with our stakeholder Marty Acevedo, a patient diagnosed with Parkinson's Disease for over eight years, we found that aerobic exercise was a factor that helps to improve these symptoms and slows down the deterioration process of Parkinson's Disease.

Figure 1: Parkinson's Disease Exercise Class

From our initial conversations with Marty, we found that boxing was a very popular form of exercise in the Parkinson's Disease community. We took it upon ourselves to research more about how boxing can be a beneficial exercise for people with Parkinson's Disease, and we came across an article from the American Parkinson's Disease Association (APDA) mentioning that Rock Steady Boxing is a fun and efficient form of exercise for patients. It pushes patients to physical heights that they did not think they could achieve (Franchina). Marty also mentioned that she plays cognitive video games, such as Wordle, daily to ensure that she works out her mind as well as her body to better her symptoms (*See Figure 2*).

Figure 2: Parkinson's Disease Patient Performing Boxing and New York Times's Wordle

In our project, we aimed to develop an innovative solution that combined both boxing and cognitive video games to provide a more engaging exercise experience for patients. The objective is to create a game that effectively stimulates both the body and the mind, encouraging patients to exercise in a fun and interactive way. To achieve this, we plan to incorporate boxing into the gameplay mechanics. By attaching accelerometers to boxing gloves, users will be able to use punches as inputs for the cognitive game. The intensity and type of punches thrown will directly impact the gameplay experience.

For the cognitive games themselves, we draw inspiration from Brain Academy (*See Figure 3*). Our aim is to strike a balance between providing sufficient brain stimulation without overwhelming the players. The games will be designed to be challenging enough to encourage critical thinking and mental engagement while avoiding excessive difficulty that might discourage patients, or simplicity that might bore them. By combining physical activity with cognitive challenges, we hope to create a compelling product that motivates patients to exercise both their bodies and minds, fostering their overall well-being.

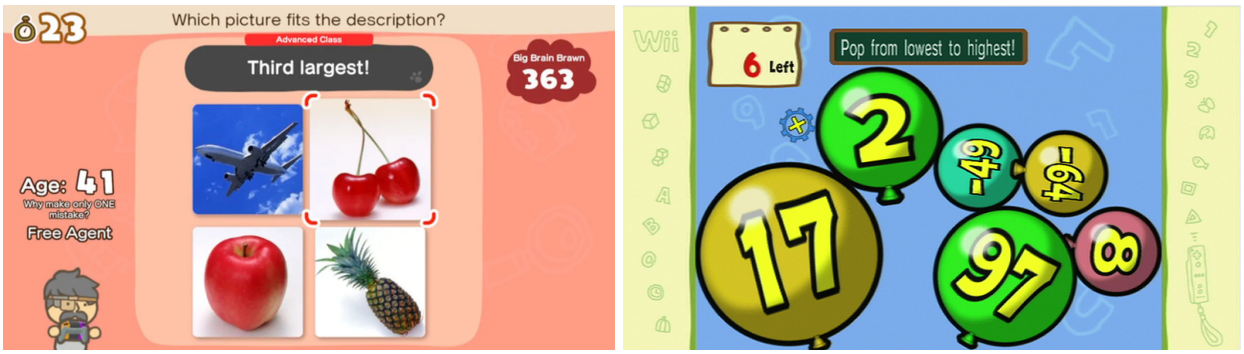

Figure 3: Brain Academy Game Examples

3 | Methodology

After solidifying our idea and getting feedback on it from Marty, we began to flesh out our concept and have it take shape. These are the steps we took to create our project.

3.1 | Design Process

To start, we began by making sketches of game concepts that we felt were suited for our cognitive boxing video game. Throughout our sketches, we made sure that we created each of them to follow a couple of criteria that both us and Marty came up with:

1. The game was difficult enough that it would be challenging but not too difficult to the point it was frustrating for players.
2. Players were able to choose what type of difficulty they wanted to play.
3. The game ensured that users would be able to get good exercise through multiple rounds of punches.
4. Players could safely play the game (ideally seated or standing still).
5. We must ensure that the game complies with our hardware constraints.

From the criteria above, we designed three different game concepts. Our first concept was a math boxing game. This would present players with a math equation, and the user would have to solve the math equation. Players would be given options like a multiple-choice test and punch in the direction of the number they wanted to choose. This would be timed-based and accumulate points to get players moving more as they earned more points (*See Figure 4)*.

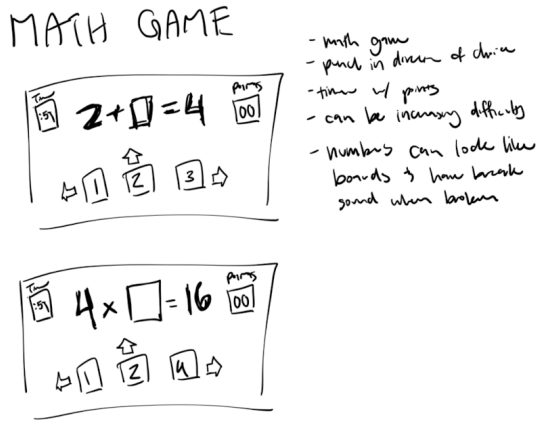

Figure 4: Math Game Concept Sketch

Another concept that we fleshed out would be a rhythm game. This would present players with colored circles which corresponded to a type of punch. A computer would perform the rhythm that the user would need to do and the user would repeat the same rhythm back to the computer by performing the punches in the correct order and timing. This would have been like a combination of the games *Taiko no Tatsujin: Drum 'n' Fun!* (a rhythm-based drumming video game) and *Friday Night Funkin'* (a rhythm-based battle game) (*See Figures 5 & 6)*.

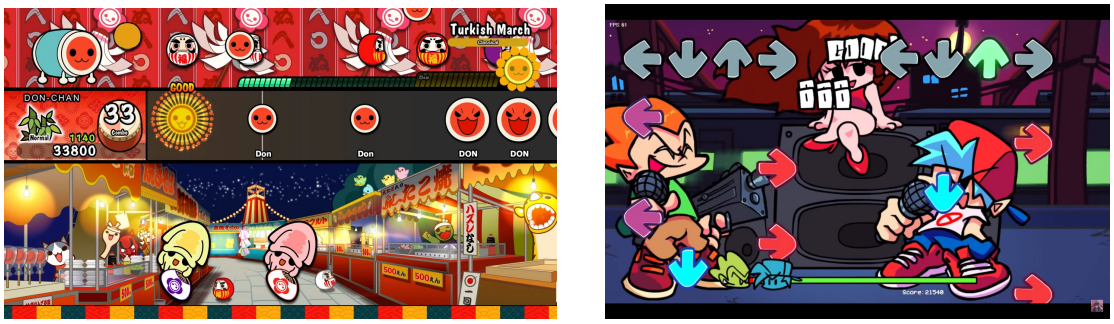

Figure 5: Gameplay of Taiko no Tatsujin: Drum 'n' Fun! (Left) & Friday Night Funkin' (Right)

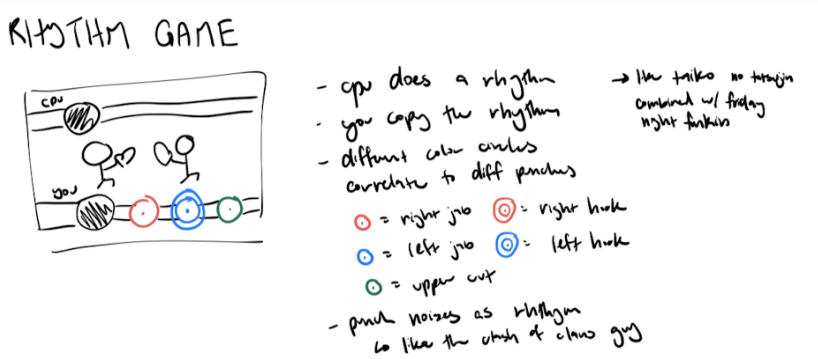

Figure 6: Rhythm Game Concept Sketch

In the end, we decided to take a different approach and go with a different type of math game. This final concept, and the one we ended up choosing, was a more complex math game that presented users with a tile with a math equation. Users would have to figure out if the math equation was even or odd and punch accordingly. There were also enemies on the screen that users would have to hook in order to get bonus points. This game was also time and point-based to get players to exercise (*See Figure 7)*.

Figure 7: Math Tile Game Concept Sketch

To test this out, we created some paper prototypes of our game concept (*See Figure 8*). We put numbers, equations, and pictures of items on post-it notes to resemble the tiles of the game while someone used the boxing gloves to punch left and right to resemble an even or odd number. Based on our initial tests, we found the game to be relatively fun and was a good workout overall. One issue that we encountered was that counting a picture of items brought up confusion amongst users and could have translated better as it involved counting rather than computing a math equation. Therefore, moving forward, we removed those sets of tiles from the game.

Figure 8: Paper Prototype Testing

3.2 | Technical Approach

Our technical approach began with figuring out what components we wanted to implement in our game and how we wanted to configure them. We designed an architectural diagram that helped to visualize the hardware (*See Figure 9)*. Through this, we show that we

are going to connect the two accelerometers to a breadboard which would then be connected to the Arduino. The two accelerometers would provide inputs for the Unity game on the host PC and computer speakers would be used for volume outputs for the user.

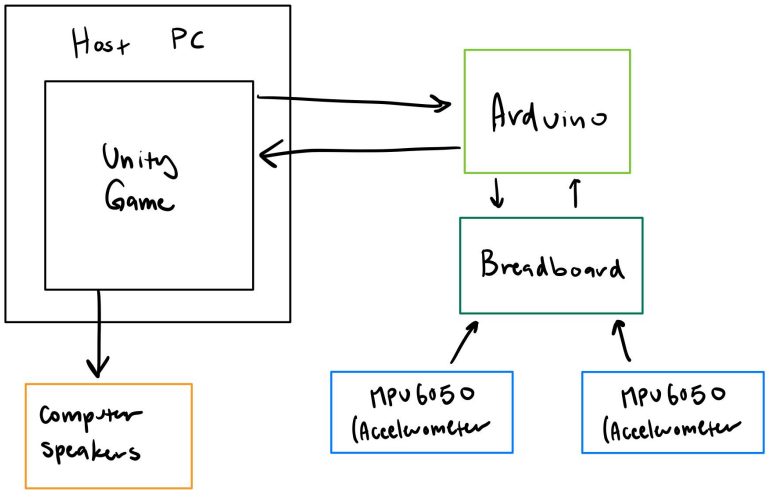

Figure 9: Architectural Diagram of Game

Once we figured out the outline of our hardware, we began to configure our hardware by checking if both of our accelerometers worked by using basic test files and wiring found online (Sanjeev). After that was finished, using the breadboard, we would test how to integrate two accelerometers into one Arduino. Wiring was the most difficult part of the project. However, we managed to figure out how to do so and began to alter the basic code provided to us. Once we found a proper threshold that would indicate a punch, we coded those as keyboard inputs for our game.

We developed our game on Unity and created a math video game that would require users to punch left if it is an odd number and right if it is an even number. To make this functional with our controllers, we made it so that the only two inputs of the game are ‘a’ and ‘b.’ This made it easier for us to route the punch to a keyboard input (left punch being ‘a’ and right punch being ‘b’) and program the game.

3.2.1 | Technical Subtask 1: Hardware & Assembly (Lead: Jonathan)

We first had to check the functionality of our accelerometers to ensure that both of them were working. We initially had to solder the accelerometers ourselves, but they ended up being unsatisfactory for our needs. Therefore, we ordered new, pre-soldered ones to aid the rest of

our project. Using our new accelerometers, we found some wiring and basic test files online (Sanjeev). Luckily, unlike our initial ones, these accelerometers were working (*See Figures 10 & 11*).

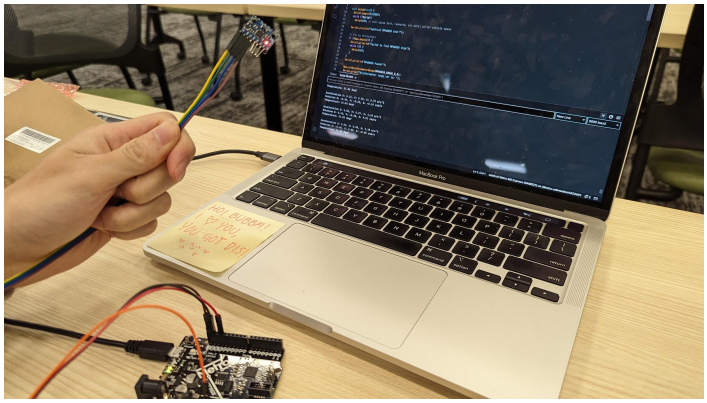

Figure 10: Initial Tests of the Accelerometers

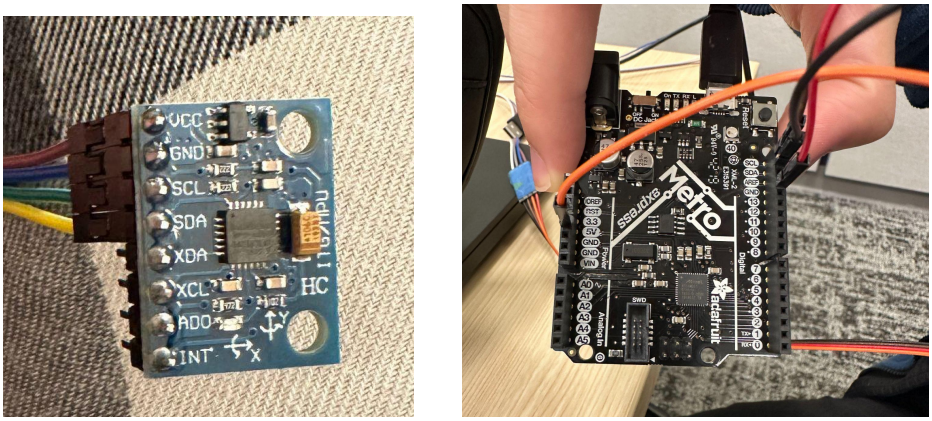

Figure 11: Wiring of Our Accelerometers

Once we were able to test the two accelerometers, we had the challenge of finding a way to get two accelerometers to work on a single Arduino. Because the Arduino does not have enough ports, we had to use a breadboard to help us connect both at the same time. We struggled a lot at first but found that we just had to swap a couple of wires to get the accelerometers to work. Initially, we wired it so that each accelerometer would be connected to the Arduino through its VCC, Ground, Serial Clock (SCL), and Serial Data (SDA) pins. However, for one accelerometer, we had to swap the wire connected to its VCC to its AD0 port. The port is meant to discern the addresses of the accelerometers in case multiple accelerometers are

connected to the Arduino (Components 101). This allowed us to successfully have two accelerometers on one Arduino (*See Figure 12)*.

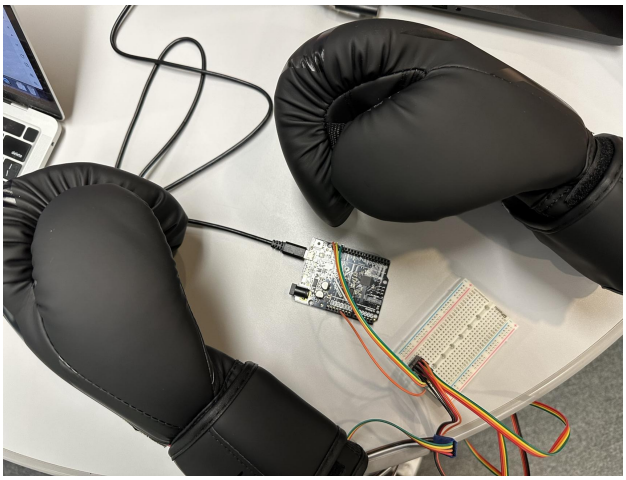

Figure 12: Successfully Connected Two MPU 6050 Accelerometers to One Arduino

We would then move on and alter the basic reading files we found to accommodate the two accelerometers. From here, we observed the readings from the accelerometers and found that when thrown a jab, there was a massive spike in the x-acceleration of both accelerometers. Therefore, we played around a little more with the code and made a threshold that if the x-acceleration of an accelerometer was surpassed, it was registered as a punch. Because we had two different accelerometers with two different x-accelerations, we were able to route the different addresses of each accelerometer to a punch of its respective hand (*See Figure 13)*.

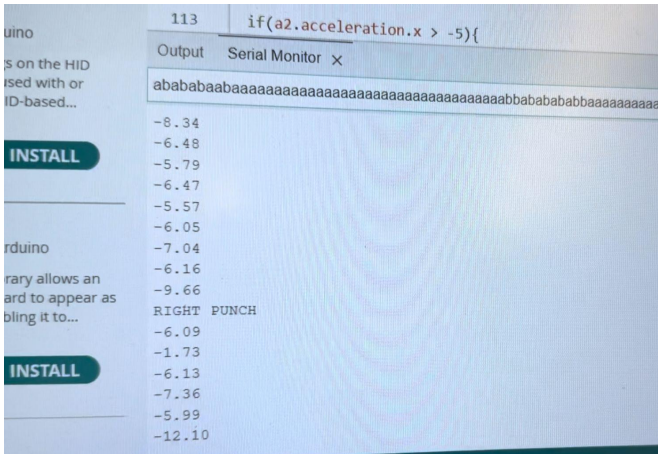

Figure 13: Initial Outputs of Our Testing

Once done, we assembled our gloves by cutting open our gloves and inserting the accelerometers into them. We connected multiple jumper wires so that players had an ample amount of room to swing their arms for punches. We ran the wires down the gloves and into the breadboard. We then duct-taped the gloves together to close off the opening that we made (*See Figure 14)*.

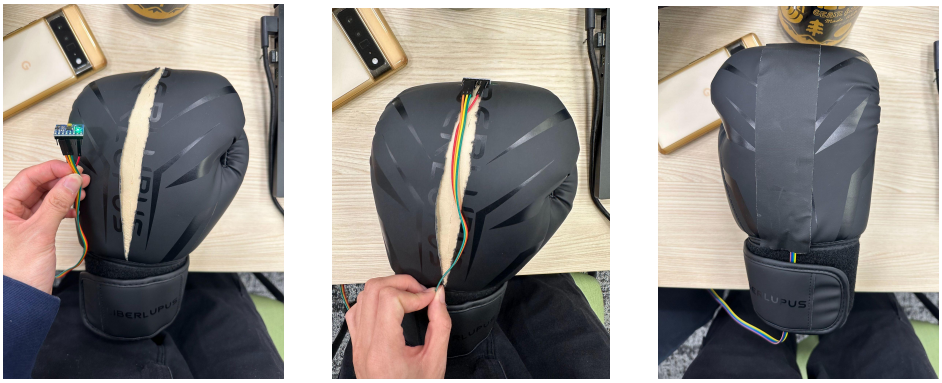

Figure 14: Assembly of the Boxing Glove Controllers

3.2.2 | Technical Subtask 2: Software (Lead: Cory)

Once we committed to a specific game design (Odd-Even Boxing) we needed to decide how the game would be developed. Ultimately, we settled on using Unity Game Engine as it hosts tools that make developing our simple game quick and easy. We also had prior experience with the engine which made this decision fairly obvious to us.

The first aspect of the game we worked on was the main game loop which can be summarized as follows: 1) Start a timer. 2) Present the player with an equation containing basic arithmetic and whole numbers. 3) The player answers even or odd for the solution to that equation. 4) Points are assigned if the player answers correctly, no points are assigned if they answer incorrectly.. 5) Loop back to (2) until the timer runs out. The next aspect was the controls. We knew that the controls had to be simple since they would ultimately be controlled using left and right punches and this had to be reflected in all aspects of the game including the start menu (*See Figure 15*).

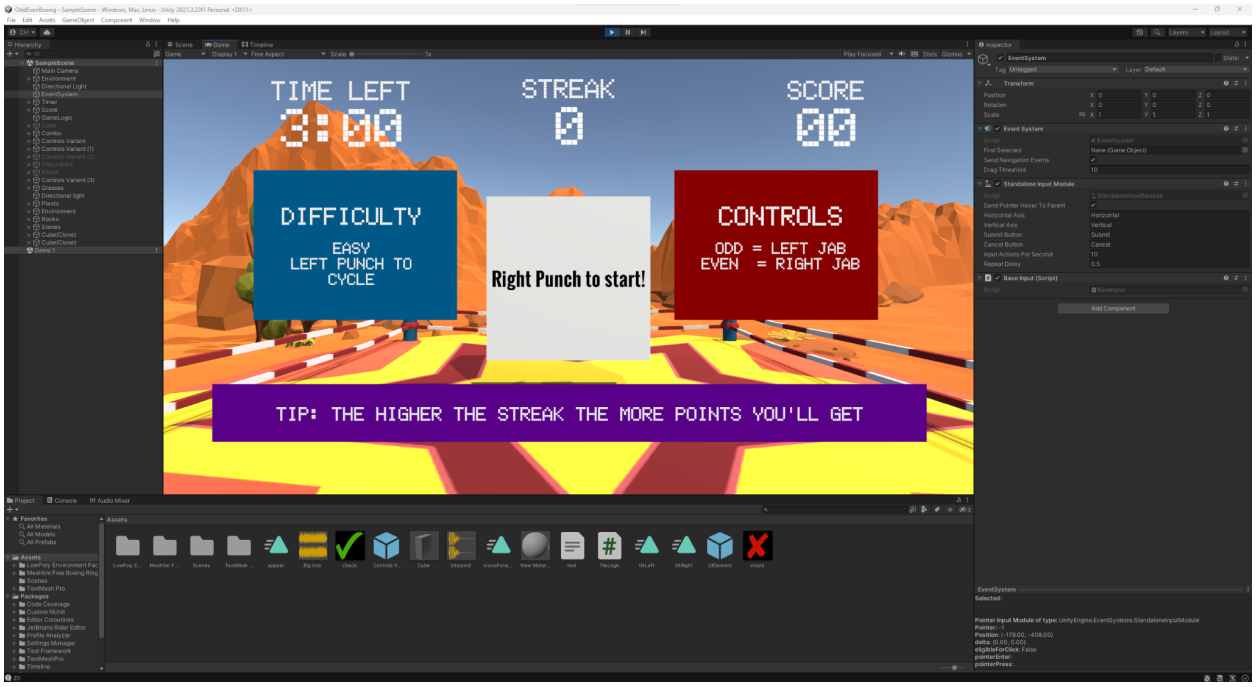

Figure 15: Screenshot of the start menu with basic controls displayed

Once these aspects of the game were fleshed out, implementation was fairly quick and simple. The equations are programmed using two arrays, an array of randomly generated numbers and an array of randomly generated signs - add, subtract, or multiply (*See Figure 16).* To find the solution to these randomly generated equations, we iterate through the array of signs and add, subtract, or multiply the numbers accordingly (*See Figure 17).*

```
for (int i = 0; i < difficulty; i++)
{
    nextNumbers[i] = UnityEngine.Random.Range(1, 55 / difficulty + 1); //Randomly generate numbers
}
if(difficulty > 1)
{
    for (int i = 0; i < difficulty - 1; i++)
    {
        nextSigns[i] = UnityEngine.Random.Range(1, 4); //Randomly generate signs (add, sub, mult)
    }
}
```

Figure 16: Screenshot of code representing the randomly generated equations

```
int finalVal = (int)numbers[0];
for (int i = 0; i < difficulty - 1; i++)
{
    if ((int)signs[i] == 1) //addition
    {
        finalVal += (int)numbers[i +1];
    }
    else if ((int)signs[i] == 2) //subtraction
    {
        finalVal -= (int)numbers[i + 1];
    }
    else if ((int)signs[i] == 3) //mult
    {
        finalVal *= (int)numbers[i + 1];
    }
}
return finalVal % 2 == 0;
```

Figure 17: Screenshot of code representing finding the solution to the equations

One technical challenge that we needed to solve was dealing with equations involving 3 or more numbers. This meant that order of operations needed to be considered when finding the solution to the equations and looping through the arrays in order could be incorrect. This is addressed in our code by first looping through the array of signs and resolving any multiplications (*See Figure 18).* For example, the equation A + B * C + D * E would become A + 0 + BC + 0 + DE.

```
if(difficulty >= 3)
{
    for(int i = 0; i < difficulty - 1; i++)
    {
        if ((int)signs[i] > 2) // if its multiplication
        {
            int num = (int)numbers[i] * (int)numbers[i + 1];
            numbers[i] = 0;
            numbers[i + 1] = num;
            signs[i] = 0;
        }
    }
}
```

Figure 18: Screenshot of code representing solution to PEMDAS

Once the development of the game was completed, combining the software and hardware implementations was easy. The simple controls of the game were configured using only two keyboard inputs ‘a’ and ‘b’. We simply needed to connect the two boxing gloves to an Arduino that would detect the left and right punch and mimic the keyboard inputs ‘a’ and ‘b’ respectively.

3.3 Evaluative Approach

To evaluate our product, we needed to know if our game worked as intended and also if players found the game usable, fun, and challenging. To make sure that our game worked on a technical level, we tested the hardware and software side individually. For the boxing gloves, we opened a notepad text document and both used the gloves to test a variety of punch speeds, punch angles, and starting positions to ensure accuracy and responsiveness. Since the boxing gloves mimic keyboard inputs we would be able to see when the gloves were not working as intended. To test the software side, we simply played the game many times and made sure that the game logic worked as intended, focusing on making sure that the solutions to the randomly generated equations were correct.

For a more human-focused evaluation, we planned to first test with our stakeholders and a few roommates and friends. Our approach to getting feedback from these tests was to first showcase or let them play the game with only a brief introduction (trying not to explain the game too much). Once they got the chance to play the game, we would ask them a series of questions to start the conversation and get feedback (*See Figure 19).* Once we got this feedback we would iterate and improve on our game.

Questions
Does this game make sense?
What would you change about the game? What would you keep the same?
Does this align with your goals and the initial idea of our project?
What do you like about our game? What do you think can be improved?

Is this something you might actually play? If not, what would you change?

Figure 19: Table of questions for our game testers

4 | Results

4.1 Technical Results

On the software side, our video game was very successful with no bugs spotted and it ran flawlessly without crashing or any bugs showing up. On the hardware side, while not perfect, we think we did a pretty good job of getting the punch detection to a usable and playable state. We found that punch detection would work using a variety of punch speeds which is very important as people with Parkinson's Disease have different severity of symptoms. The punch accuracy and responsiveness were good, but would occasionally detect punches when none were thrown. It also seemed like the initial stance of the player was important as punches would be incorrectly detected whenever the player put on or took off the gloves which hindered the usability of the product.

4.2 Human-Centered Results

Our first person to see our finished prototype was our stakeholder Marty. Her reaction was extremely positive and she really liked the design and implementation of our game. She also thought it made a lot of sense for people with Parkinson's Disease to play as it cognitively stimulates, provides a safe exercise experience, and challenges the mind-muscle connection of the player. From the questions we asked (*See Figure 19)*, we got a lot of useful feedback. She thought one of the features of our game called the "Combo" system was confusing as she thought it meant punch combos rather than player streak. Marty also thought that the timer may be too rushed, especially on the harder difficulties. She also thought that it would be a good idea to provide even harder difficulties for a more challenging experience. After our meeting, we used this feedback and implemented a few changes: 1) We changed "Combo" to "Streak". 2) We doubled the timer. 3) We added two additional difficulty levels: "Very hard" and "Extreme".

We also performed a similar evaluation approach with other game testers (consisting of roommates and friends) and received additional feedback. Some thought that there needed to be better indications for correct and incorrect answers, some didn't initially understand the streak system, and some thought the streak system was unbalanced. We addressed these concerns by 1) Implementing checkmark and x-mark indicators. 2) Adding a tooltip on the start menu. 3) Max the streak to 10. (*See Figure 20)*

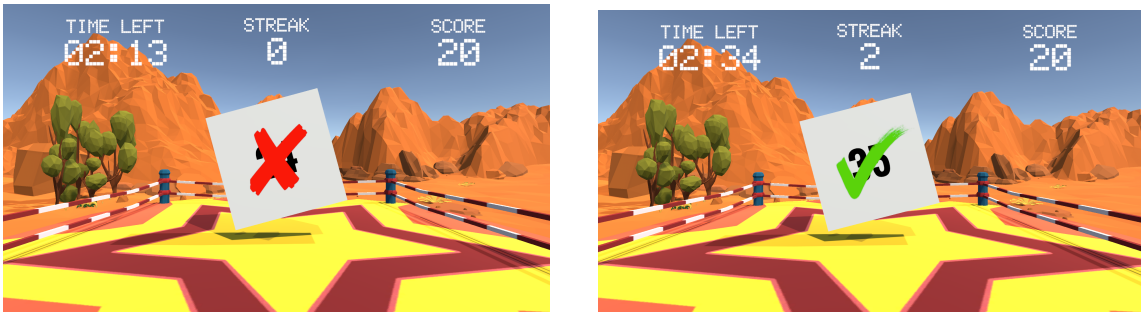

Figure 20: Changes based on feedback from user testing

Overall, we think our project was a success. While we didn't get the chance to implement some of the boxing features we would have liked to such as the vibration motor for feedback or the detection of more complex punches like uppercuts and hooks, we made a final prototype (*See Figure 21)* that our stakeholder could really see herself using and went beyond her expectations.

Figure 21: Final prototype (right) and prototype in use (left)

5 | Discussion & Future Works

Though we found our project to be a success, there were some shortcomings on our part that we wished we did differently when we initially began the project. Firstly, we hoped that we researched a little more into what our hardware could have done a lot earlier than we ended up doing. We were not sure about the capabilities of the accelerometer and what metrics it was able to output. Therefore, we did not use it to its full capabilities in our current project. We originally wanted to implement a variety of punches such as hooks and uppercuts. However, due to our lack of knowledge of our hardware, we were not able to efficiently implement them.

If we were to be given more time to do our project, we would have liked to implement two things, Bluetooth and vibration motors. Due to time constraints, we were unable to integrate either into our current project. However, having both of these would definitely improve our current design and make it more polished. Being able to have Bluetooth controllers rather than having super long jumper wires connected to the Arduino would make it so users would not get tangled within the controllers' wires. It would also provide a cleaner look to our product. Having vibration motors within the gloves would also allow for better gameplay as there was greater feedback between the user and the game. Currently, the only feedback players receive is the auditory feedback of the punch and the visual feedback of the tile turning left or right. Implementing physical feedback would definitely help to improve the game and make it more immersive.

In the end, our project was a success. We are happy to have worked with Marty and are glad that we were able to develop a product that both parties are happy with. If future students were to do the same project, we would encourage them to start early and make sure they know the extent of their hardware as it was something that we struggled with. We would also encourage them to keep game concepts practical because though we had ambitious ideas, we had to keep them all within the scope of our time frame. Hopefully, students would find it in themselves to continue our current project and expand upon the possibilities that it could have been if given more time.

6 | Work Cited

Franchina, Phil. “Boxing and Parkinson’s Disease.” *American Parkinson Disease Association*, 11 Nov. 2020, www.apdaparkinson.org/article/boxing-for-parkinsons-disease/#:~:text=What%20is%20Rock%20Steady%20Boxing,eye%20coordination%2C%20footwork%20and%20strength.

“MPU6050 Accelerometer and Gyroscope Module.” *Components101*, components101.com/sensors/mpu6050-module. Accessed 14 June 2023.

“Parkinson’s Disease.” *Mayo Clinic*, 26 May 2023, www.mayoclinic.org/diseases-conditions/parkinsons-disease/symptoms-causes/syc-20376055.

Sanjeev, Arvind. “How to Interface Arduino and the MPU 6050 Sensor: Arduino.” *Maker Pro*, 14 June 2023, maker.pro/arduino/tutorial/how-to-interface-arduino-and-the-mpu-6050-sensor.