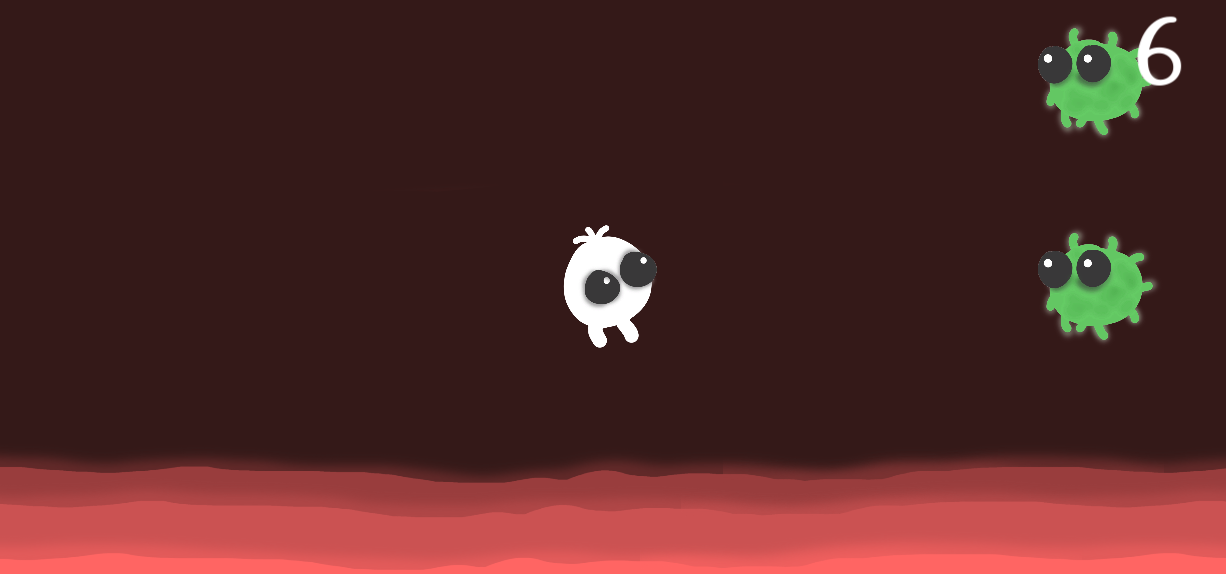
# Gesture Based Documentation

Ethan Horrigan & Dylan Loftus

<https://github.com/ethanhorrigan/Gesture-Based-Project>



Contents

[Gesture Based Documentation 1](#_Toc37168175)

[Contents 2](#_Toc37168176)

[Architecture 3](#_Toc37168177)

[3](#_Toc37168178)

[4](#_Toc37168179)

[Background Movement 5](#_Toc37168180)

[Player Movement 6](#_Toc37168181)

[Score Handler 7](#_Toc37168182)

[Gesture Phase 8](#_Toc37168183)

[Artwork 9](#_Toc37168184)

[Sound Design 10](#_Toc37168185)

[Speech Recognition 11](#_Toc37168186)

[Conclusions 12](#_Toc37168187)

[Libraries 13](#_Toc37168188)

[References 13](#_Toc37168189)

### Architecture

Ethan

### 

The Myo armband is responsible for controlling the player (i.e. the movement) and Speech Recognition is used to control UI elements, such as pausing the game.

Myo Controls (Player Movement):

WAVE IN: Moves the player down.

WAVE OUT: Moves the player up.

For the Gesture Phase, the pose used must equal the gesture given in-game.

### 

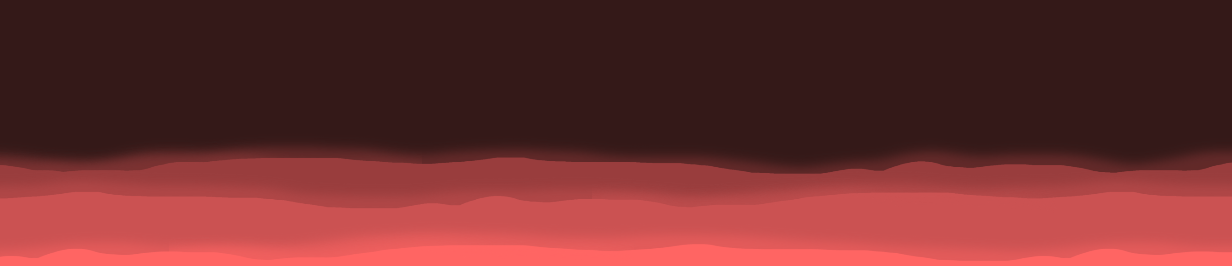
### Background Movement

Ethan

Since the genre of this game is an Endless Side Scroller, the background moves instead of the player moving, giving the illusion that the player is moving.

The background sprites have a Starting X position and an End X position, the background is then translated on the X-axis. When the End of the sprite reaches the End X position, it is reset back to its Starting X position. To ensure this worked correctly, the starting X and ending X must meet at the same point or else the movement would not be smooth.

I created 3 “Wave” sprites using this movement at different opacity and speed to give the game a sense of depth.



Adapted from: <https://answers.unity.com/questions/19848/making-textures-scroll-animate-textures.html>

### Player Movement

Dylan

The player movement in this game is simple. The movement consists of three lanes. Top lane, Middle lane and Bottom lane. These three lanes are three transforms in Unity and the player’s transform.position is set to the transform.position of Middle lane at the beginning of the game.

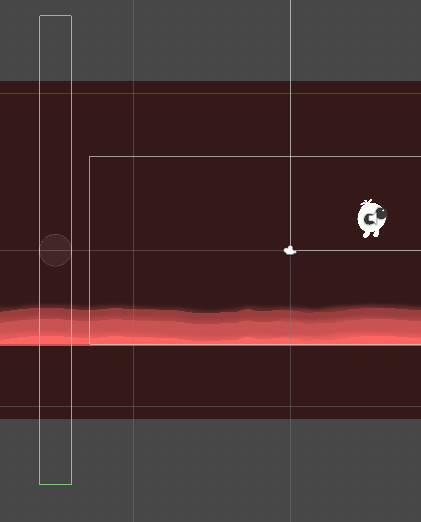
If the player is in Middle lane for example, and the user waves in the GoDownLane(); method is triggered in the PlayerMovementMyo script. This method checks to see what lane the user is in and sets the player’s transform to the lane below the current one that the player is in. If the player is in the Bottom lane though the player’s lane is set to the Bottom lane.

### Score Handler

Ethan and Dylan

Originally, Dylan had implemented the score handler which incremented based on how long the player was alive, we both agreed that refactoring it to an enemies-defeated based score handler would be better suited.

This was achieved by adding an invisible object off-camera with a collider, each time an enemy collided with this object, the score would increment.



### Gesture Phase

Dylan

If the user sores more than 4 points while traversing through enemies. We enter what we called the “Gesture phase” In this phase a “password” is generated. The password consists of an array of myo armband poses. This array’s length is random every time and each individual pose in the password is randomly generated every time also.

To create a random array of myo armband poses every time I first created and initialized an int array with random length between 2 and 5 with random values between 1 and 5. Each myo armband pose had its corresponding integer value. For example, 1 represented WAVE\_LEFT and 5 represented FINGERS\_SPREAD. This array is then thrown into a switch statement. This switch’s job is to create the myo armband Pose[]. If the first index in the int array is 3 for example, the first index in the Pose[] is set to DOUBLE\_TAP.

The aim for the user during this gesture phase is to replicate the gesture displayed on screen with the myo armband BEFORE the gesture meets the left-hand side of the screen. The score handler object is used to detect if the gesture has fallen off the screen.

A screenshot of a computer

Description automatically generated

### Artwork

Ethan

The artwork was made using Adobe Photoshop and a Huion Drawing Tablet.

The elements of the player were drawn on different layers so I could animate each part individually. This was also done for the enemies.

I’m glad that I opted to create unique assets instead of using other assets, because it gave me a lot more control of how the game should look.



### Sound Design

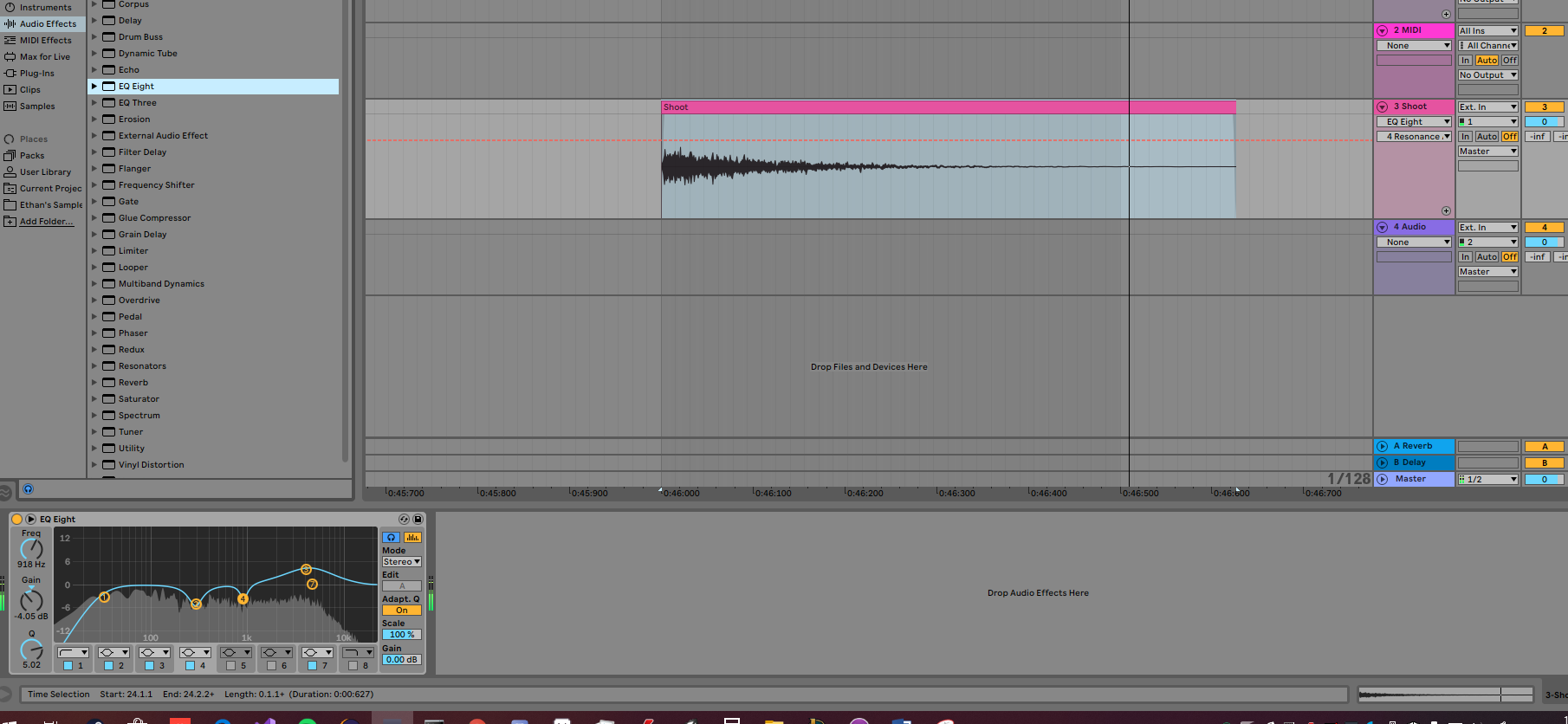
Ethan and Dylan

We used <https://freesound.org/> to obtain sound effects. These sound effects were then imported into Ableton for further processing.

I did this because some of the sound’s quality were not great and I could also process them in any way with the use of equalizers, compression, reverb etc.

I compressed each sound effect with similar settings so the range of volume between sound effects were relatively similar.

Program: Ableton



### Speech Recognition

Ethan

Speech recognition was added using Unity's speech library.

In the beginning, the state of the game was managed using the Myo Armband, but this caused conflict with the gesture phase and not being able to pause during this part of the game. I incorporated speech recognition to solve this. Using 4 states

Play

Pause

Continue

Quit

### Conclusions

Ethan:

Overall, the project was enjoyable to create, working with hardware that I haven't used before was an interesting experience and exposed me to more interesting technologies. My favorite part of this project was implementing Myo's Gesture to control the player and implementing Speech Recognition.

Initially, Myo controlled all elements of the game, but it caused bugs when trying to control the UI. Therefore, I added speech recognition to control the UI instead.

The most satisfying part of the project was when it was converted from a prototype game into a polished game with the use of Artwork, Sounds and Effects. If I was to do this project again, I would implement more mechanics to incorporate the Myo more.

Dylan:

During the development of this project I had great fun. Creating a dynamic password for the gesture phase in the game was a challenging but enjoyable experience. At the beginning of the project I had the myo armband for a while and exploring the sample scenes provided in the SDK was really fun.

The most challenging part of the project for me was when I was trying to implement a method that would return true if all of the locks in a given password were unlocked, meaning that the user had accomplished the gesture phase. Originally, I had an array of Booleans and was constantly checking to see if all Booleans in the array were true. After discussing my issue with Ethan we changed the Boolean array to a counter that was the same length as the Pose[].

If I were to take on this project again I would add more collectable items such as health items or score boosting items.

### Libraries

Myo SDK: <https://developerblog.myo.com/setting-myo-package-unity/>

Speech: <https://docs.microsoft.com/en-us/windows/mixed-reality/voice-input-in-unity>

### References

[SoundFX] <https://freesound.org/>

[Font] <https://www.fontsmarket.com/font-download/burbank-big-condensed-bold>

[Music Playing Throughout] <https://answers.unity.com/questions/1260393/make-music-continue-playing-through-scenes.html>