Unity Game

Members:

- Patrick Griffin – G00314635
- Gareth Lynskey – G00312651

GitHub Link: https://github.com/lynskey08/Unity-Mobile-Application-Game

Project Overview:

The purpose of this project is to create a single player Space Shooter survival game. The game is an existing project we had done in Unity in third year. We are going to use the Myo Gesture Control Armband to control the movement and functionality of the spaceship/game.

We want to experiment with a variety or gestures in the game and incorporate the gestures that have the most relevance to the game. We also hope to incorporate a target goal for the game such as a final boss or levels.

How to download and run the application:

To operate this game, you must have the following:

- Unity 5.5.1 or a newer version. The game is not backwards compatible.
- Myo Gesture Control Armband.

You must have the Myo Connect installed which can be found here https://developer.thalmic.com/downloads.

Download and extract the project to your desktop.

Open unity and at the project menu, open the extracted project located on your desktop.

Put the Myo Armband on and make sure it is synced while playing the game as the gestures may not be recognized.

Have a look as the gestures we created to become familiar with the game.

Click the play button in Unity.

Game Functionality:

- The aim of the game is to destroy the enemy ships and asteroids, and to stay alive as long as possible.
- The game difficulty is set to increase as the player progresses.
- If the score is less than 200 the enemy ships will not shoot lasers.
- When the score is greater than 200 the enemy ships will begin to shoot at a low fire rate.
- When the score is greater than 400 the enemy ships fire rate increases.
- When the score reaches 600 the enemy ship will reach its maximum power and fire a continuous laser beam.
- We capped the progression of the game at these score for testing and marking purposes as it was very difficult to progress to the maximum difficulty progression of the game with 5 individual checkpoints to pass.

Gestures identified as appropriate for this application

Gestures that we are working on and hope to complete. We thought that these gestures made most sense in our opinion for the game. We wanted to use as many gestures as possible with significance to the game.

We used the fist gesture to fire single shots with a higher rate of fire. We were debating over this gesture and the spread finger gesture but we thought this made more sense for the single fire as a fist seems like one

We used the spread fingers gesture to fire double shots with a lower rate of fire. We decided to use the spread fingers gesture for this as it seems to be the most relevant to the game and because while we were testing out the gestures, the fist and the spread fingers gestures seems to mix up sometimes so we tried to avoid this problem as best as possible by making the two gesture relatively similar to when using one of the two, so that the user was never left without the option of firing a weapon. Therefore, the user wouldn't be at a great deal of a disadvantage. We found this to be a logical choice.

We used the wave in gesture to pause the game. We wanted a way to pause the game with a gesture and we found that the wave in gesture suited best because we were playing around with the program VLC media player this gesture was used to stop and rewind the video, so we thought this gesture would fit best for pausing the game.

We used the wave out gesture to resume the game. Similar to the reason above but for resuming the game, we found this gesture relevant to the game as it was incorporated in VLC media player's myo gestures.

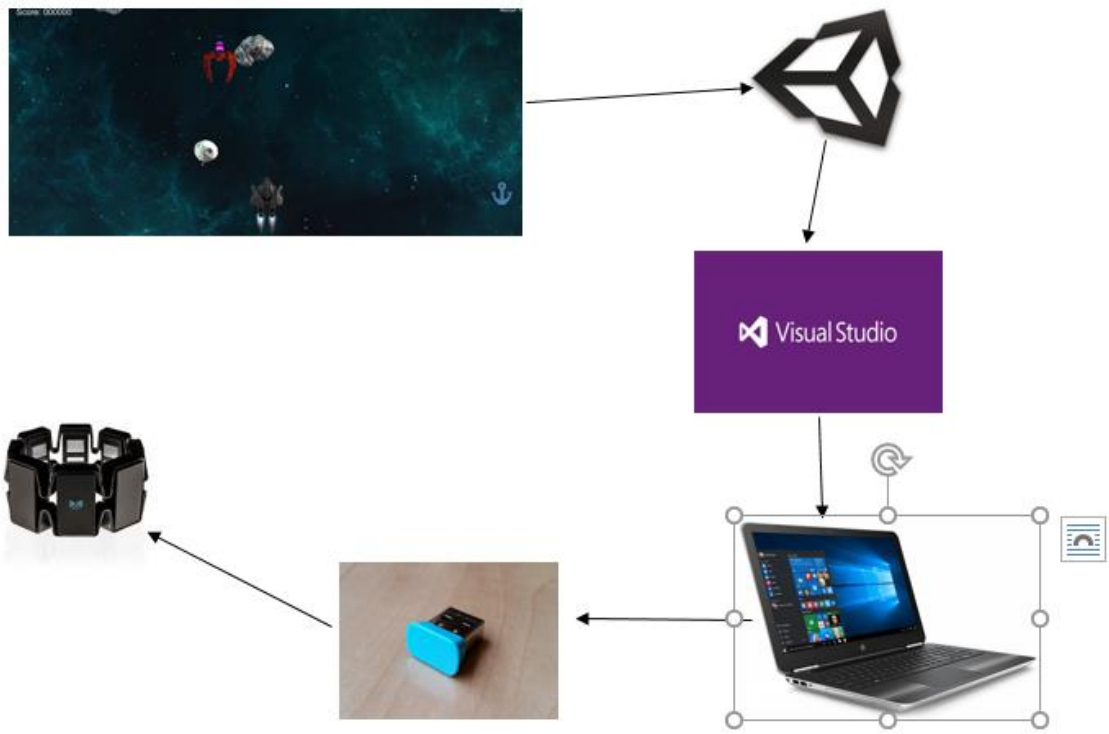
Hardware and Technologies:

- Myo armband
- Unity
- GitHub
- Visual studio/Mono Develop

Myo Gesture Controlled Armband

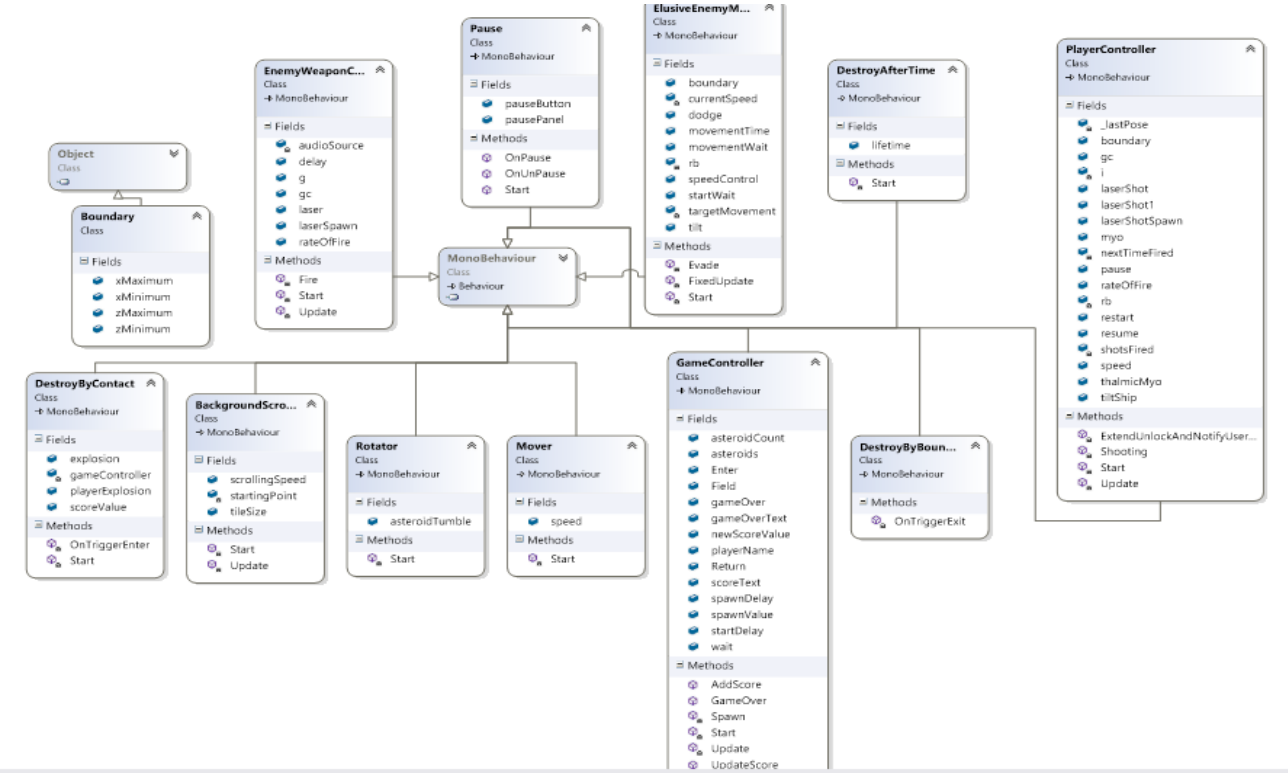
The Myo armband is a device that recognizes hand gestures and movement by the user with the device on their upper forearm. The Myo itself works by having a set of sensors called electromyographic (recording the electrical activity produced by skeletal muscles). It also combines itself with a gyroscope, accelerometer, and magnetometer to recognize gestures. That is the main difference between the Myo and the Leap Motion while the Myo uses sensors on the device connected to the users arm the Leap Motion uses two cameras and three infrared LEDs to track where the user's hands are. This device isn't wearable its used by the user hovering their arms over the controller to make their gestures. The Myo also lets users to add their own unique gestures.

Architecture for the solution



The above diagram shows that the Myo armband connects with the laptop using a Bluetooth adapter. Visual studio is used to code the gestures in C# and the sprites and the user interface is created in Unity.

We also used Mono Develop for some parts but predominately used visual studio as it was preferred for breakpoints and debugging.



Session Updates and Problems encountered

Session 1

Got Myo packages and dependencies installed on unity.

Fixed the code in Myo samples as code was old and obsolete.

Reference- http://developerblog.myo.com/setting-myo-package-unity/

This tutorial shows how to get started and set up.

We found it difficult to incorporate the Myo into our project and didn't know where to start as the myo.dll caused problems. We followed the tutorial above in what to do with the myo.dll file but kept getting errors saying there was duplicates of the myo.dll even though there wasn't. So in the end we left the myo.dll file as it was and everything worked fine.

Session 2

Space Ship is moving with arm movement is completed. This was done with using the Myo samples provided to us. We attached the Hub - 1 Myo to a gameObject we created for the Player's ship that we got from the Myo samples given and added Myo Script to the PlayerController to control the ship. It was difficulty to get the Myo and the ship connected but after much trial and error we finally got it.

Reference - https://www.youtube.com/watch?v=5GLbbWq_d2w&list=PLhQ99zkWQzkPENhfnaaUyrOpSomBGPKrs&index=1

We managed to get the Player ship to map to the orientation of the myo armband, simulating the being moved by our arm joint movement along the x-axis.

Session 3

As we have the player moving we now want to work on a shooting gesture. We decided to use two gestures for two different shooting methods. We thought of just using the fist gesture for a single weapon but decided that adding the spread fingers would be good also as the myo fist and spread fingers gesture sometime glitched and got mixed up, so used this solution to resolve the conflicts so it wouldn't cause any major problems to the functionality of the game.

Session 4

We had difficulties adding the spread fingers gesture as we were assigning two different laser bolts to the one game object array, so we would have two different weapons using one object array, but it cause problems interchanging between the two gestures as the second object array wouldn't register with the gesture. To solve this problem we created two separate objects for shooting.

Session 5

Working on another two gestures to pause and resume the game. We thought of these gestures as we thought every game should have a pause and resume button. We thought that the wave in and wave out would be suitable gestures for pause and resume.

We created the pause and play buttons which are fully functional. Both the buttons are onClick events in their own separate script.

Session 6

Attempting to map the wave in and wave out gestures to the pause and play button. We found it difficult to invoke an onClick event within a function in the PlayerController script from the Pause script but eventually we figured it out.

Session 7

All gestures complete but we need an end goal for the game. We decided to make the game difficulty increase as the game progresses.

Conclusions & Recommendations

Throughout the project having the game registering a specific gesture turned out more difficult than we thought. For example, if we want to make a fist gesture it would register the spread finger gesture and vice versa, so we incorporated these two gestures for firing so the user wouldn't be at a disadvantage. We figured out that it wasn't the code but the myo armband itself. We needed to set the Myo armband setting for the gestures and record these gestures for the specific user. A simple error but caused a lot of problems and a lot of time.

We had difficulty with setting up the myo game object with the player object in unity. This is done by dragging the hub-1 Myo Game Object from the myo sample files onto the myo game object in unity which we created in visual studio. This took us a long time to figure out because we jumped straight into researching it on the internet when the solution was in the myo samples the entire time.

We also encountered a problem with mapping the double tap gesture to a restart/quit button. We kept being prompted errors saying "ThalmicHub failed to initialize. More than one ThalmicHub in the current scene". We found no solution to the problem and asked other student about it and they encountered also encountered the same problem. In the end we left the double tap gesture out of the project which I wasn't really happy about but it had to be done in order to avoid the problem.

There are some irritating aspects of the Myo armband for example sometimes when testing the application, the Myo would be "out of sync" causing us to re-sync and to restart the application. However, we were happy with the reaction time of the Myo armband throughout the game after the Myo was "warmed up" fully.

We found that once you get everything set up correctly it's a matter of trial and error.

We were asked to add a boss or level to our game as it was originally just the player shooting asteroids and enemies which added to a score, to simulate an end goal. We knew that we weren't able to create a boss due to the lack of time we had left because that would require giving the boss health, which would also require giving the player health due to the fact that the game was always a one shot kill. So we solved this problem by making the game progressively harder by speeding up the fire rate and the delay between shots of the enemy ship as you passed a certain score.

We would recommend on undertaking a Myo armband project using unity as it was actually a pretty fun project to do in the end. I recommend to anyone using the myo in a future project to analyse the myo samples files as they helped us out a lot and they are certainly not to be ignored like the way we did.

If we were to undertake the project again we would look more into using the accelerometer and gyroscope to possibly help add a couple of more cool gesture.

All References

- http://developerblog.myo.com/setting-myo-package-unity/ - Setting up the Myo
- http://developerblog.myo.com/myocraft-hack-with-us-for-thalmichackd/ - Getting set up
- https://developer.thalmic.com/forums/topic/10708/?page=1#post-19317 - Helped with import error
- https://github.com/thalmiclabs/myo-unity/ - Helped with Myo sample
- http://answers.unity3d.com/questions/964292/any-ideas-as-to-why-i-get-these-errors.html - Helped with error
- http://grimpixelstudios.com/getting-the-myo-armband-to-work-with-unity-on-android/ -getting Myo working with unity