GESTURE BASED UI DEVELOPMENT PROJECT

ABSTRACT
This is a Word document for my Project with Gesture implementation with the Myo Armband and voice recognition in the module Gesture-Based UI Development for Mr. Damien Costello.

Jack Haugh
Software Development

Link to the GitHub Repo

https://github.com/JackH97/GestureBasedUI-MotorMountain.git

Link to Screencast

https://youtu.be/qOqlUuT2RKY

Introduction

For this project, the aim was to develop an application with a natural user interface. There were multiple types of technology that was available for us to use. For my project I created a 2D unity game which allows the user to play the game and control different menus using the Myo armband and voice recognition. The game I created is like Hill Climb Racing because it is a game I played over my youth and still enjoy playing to this day. To play the game, the user uses voice commands to go around the main menu and options menu and gestures with the myo armband to play the game and pause also.

Purpose of the application

The main purpose for this application we are creating is to show off how using different gestures such as voice commands and using hardware such as the Myo-Armband for myself can be used to control the application instead of the usual mouse and keyboard approach we have been used to for years. For my application, I used the voice commands to let user start a new game from main menu, go to options menu where you can mute audio and exit back to main menu and end game. While in-game, the user uses the Myo-Armband to go forward and go backwards and to pause and resume the game.

The Myo-Armband is a gesture control armband developed by Thalmic Labs. It uses a set of electromyographic (EMG) sensors that sense electrical activity in the forearm muscles, combined with a gyroscope, accelerometer, and magnetometer to recognize gestures. When worn on a user's forearm, the Myo armband detects the user's hand gestures and arm motion. The Myo can be used to control video games, presentations, music, and visual entertainment. To work with the user’s pc, we had to download drivers and the application manager. Then if the user takes it off,

they must resync the device to their pc again to make gesture work again to the best of its ability again.

Gestures identified as appropriate for this application

Regarding the Myo-Armband, it uses different variety of gestures to incorporate into your application which are:

The armband is linked to the arm and synced up with the Armband Manager application for the Myo then the user as long as they have setup the scripts and code to work with application can use the technology with their game depending on the gestures they use. For my application, waving left makes the car go backwards and waving right makes the car go forwards, also double tapping or pinching pauses the game and making a fist resumes the game.

Voice recognition was integrated into the application for use with the main menu and options menu to navigate between the menus from playing game and going to the options menu and muting the sound there also. I thought using voice feature for the menus was better to ease the user's way of interacting with the application and leaving the Myo-Armband for just in-game use and I feel this is better for sure.

Hardware used in creating the application

For my research in deciding what to use for the project for this module and hearing of its availability I decided to go with the **Myo-Armband** to use for the project because of the uniqueness with the device and technology I hadn't come across until this yr. The armband as of now has been discontinued by Thalmic Labs but I was interested in the challenge in succeeding in creating an application and incorporating the technology into my project. During the installation of the armband, the armband manager application gives the user a demo on how to use the device on their pc with different commands. After the walkthrough the device is ready for future use again by just syncing up the device again with a wave out gesture. Regarding the device, there is drawbacks such as limited amount of gestures recognized with the device and the documentation and tutorials of the technology is limited because of the technology being discontinued now by the developers. During the production stage I decided to incorporate **voice functionality** also to the project to make navigation through the menus much easier than the usual mouse and keyboard approach. I had past knowledge regarding voice with having done a project previously with this and found it good to incorporate into my projects and had no difference with it for this application. in the main menu and options menu as another wat to interact in the game and their ease of use is also a plus. To add them into the application, we create a .xml file where we have to set up rules with hashtag's and with these rules we create items in these different rules to have certain phrases for the application to react with and implement in game. From here we go to the c# file where we want the voice recognition to work and add different methods to call the .xml file we created and the rule to be called over from the file an example as to what I done was the following. We also use a microphone and in project settings under player we must check off for unity to allow the use of a microphone.

```
<rule id="startstate">
    <item>
      <tag>out.action = "new";</tag>
      <one-of>
        <item> Start a new game </item>
        <item> Begin a new game </item>
        <item> New game </item>
        <item> I want to play a game </item>
      </one-of>
    </item>
</rule>
```

```
<rule id="optionsstate">
    <item>
      <tag>out.action = "option";</tag>
      <one-of>
        <item> Go to Options </item>
        <item> Options Menu </item>
        <item> Options </item>
        <item> I want to go to the Options Menu </item>
      </one-of>
    </item>
</rule>
```

```
<rule id="quitstate">
    <item>
      <tag>out.action = "quit";</tag>
      <one-of>
        <item> finish the game </item>
        <item> exit game </item>
        <item> I give up </item>
      </one-of>
    </item>
</rule>
```

```
switch (KeyWord)
{
    case "new":
        StartGame();
        break;
    case "option":
        OptionsScreen();
        break;
    case "quit":
        QuitGame();
        break;
}
```

Architecture for the solution

My application was created using unity in a 2D scene, so it means the camera is fixed and doesn't move in the scene and stay in the same view for the game. Using voice commands will navigate the user between menus to get to the game or option menu and the Myo gestures are used in game then to play the game by moving the car forwards and backwards and for pausing and resuming the game. The aim for my game is for the user to reach the end of the track while balancing himself so he doesn't rotate over damaging himself and then having to start again but also reaching the fuel canisters before he runs out or then the car stops and the controls are also out of action. The Myo-Armband is the main piece of hardware here to control the car and go backwards and forwards and pause and resume the game. In my folder called "StreamingAssets" has MainMenu.xml and OptionsMenu.xml files for the speech recognition for my menus for the application and then they link to my MainMenu.cs and Options.cs files to call the phrases from their xml classes and bring into the application so when any of the phrases are said they interact with the application. I have a DeathTrigger.cs file so when the players head or body touches the ground it restarts the game again to play. I have a AddFuel.cs file so when the player collides with a fuel canister it adds 1 fuel to the car to continue playing and deletes the canister from scene. My main file CarController.cs imports the myo functionality from the MyoMovement.cs file to let the gestures work with my application and here also have set up to play the game with the left and right arrow keys also to have that option. To play the game after all the explaining you must have the Myo-Armband synced up to your device and microphone ready also for the phrases in the menus.

Conclusion

Overall, after finishing the project and submitting, I was happy with the implementation I did with different gestures with arm movement with the Myo-Armband and voice functionality. With the positives of the technology there were negatives I had also. Firstly, I found sometimes the Myo-Armband was unresponsive with some gestures such as wave-in can be harder for the Armband to recognize compared to wave-out which I found worked nearly all the time as

intended. The warmup process can be time consuming when trying to test the application with the band and sitting there for minutes waiting for the device to be ready for use. Guides and resources for the armband are as of now very limited with little to no support now for the device as its discontinued which made the learning and implementing process tough at stages. I agree though it has a place in regards to different games and applications but for something like I created, some people might find it annoying having your arm in the air and holding a gesture in place for so long can be annoying and tiring and I think if I came back to change things around I'd look into this by allowing user to just make the gesture once tells the game to stay in that mode until doing another gesture cancels that gesture out. As for the negatives I have pointed out with the technology, there was some positives for sure with learning a technology like this which I never did before and thoroughly enjoyed the learning aspect of this project and found working with it in unity made me enjoy it more compared to using any other language or program. I would for sure work again with the technology after learning what I know now and look at other avenues as to see what to use the technology with.