Pacman in unity, controlled with a myo armband

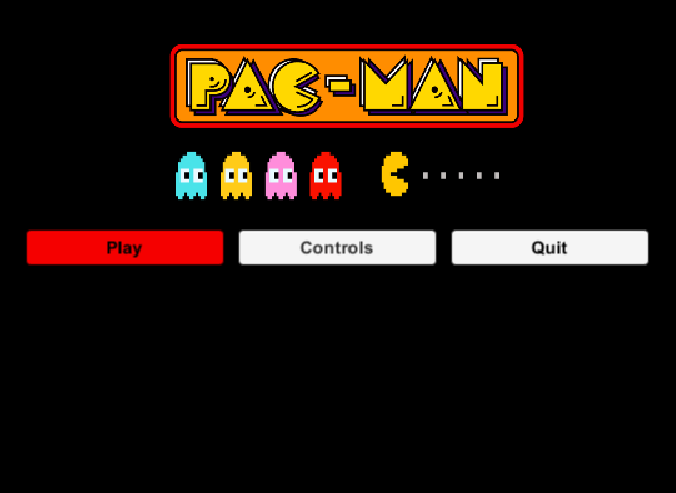
By James Kinsella & Niall Devery

**Project planning:** To begin we settled on what kind of application we would like to build. We decided on Pacman because it is an iconic game, it has simple functionality from the users perspective, the gameplay is intuitive, there are plenty of resources openly available, the game architecture is well know and that the simple functionality would correlate well with most hardware available to us.  
 We then decided on what hardware we would use as an appropriate control mechanism for this application. We researched different forms of gesture-based controls available. We decided on the MYO armband.  
 We then move onto a division of the workload Niall would set up the game environment while James would work on the player movement and animations initially. Once we had basic functionality implemented, we used a MYO armband to test if the player could control the Pacman using it. James would continue with the gameplay while Niall would development the menus and navigation.

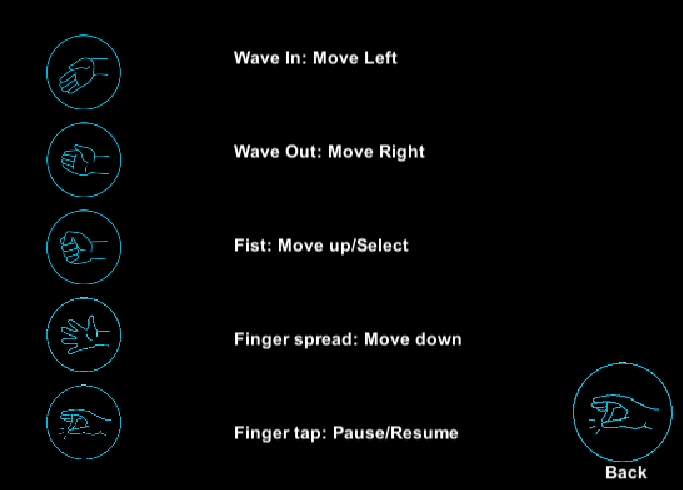
**Purpose of the application:**   
 The purpose of this application is to faithfully recreate Pacman in unity controlled by a MYO armband and to also have the menu navigation controlled using gestures.  
To start the player will begin in a the splash screen where the MYO hub is instantiated as well as the music for the gameplay.



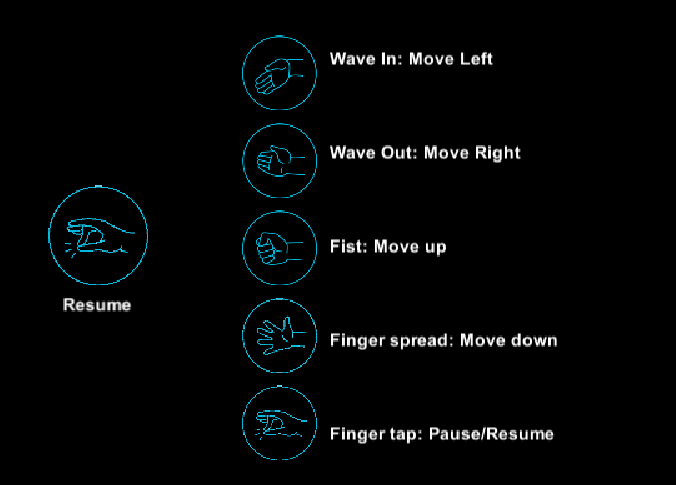
From there the user will be taken to the main menu after three seconds. The MYO object in the Hub is tagged so it is accessible across scenes. The main menu consists of three buttons that the user can manoeuvre between by performing the wave in and wave out gestures. We achieved this by using a input simulator, in unity the event system uses the arrow keys for navigation and space key for selection by default provided that one of the buttons or objects is set to be the initially select object in the event system itself, so what we did was essentially mapped certain gestures to the keys on the keyboard in order to navigate through the menus. Each one of these buttons can be selected by performing the fist gesture which triggers a simulated space key press.



The controls panel is accessed by selecting the controls button on the main menu it gives a rundown of the controls both in the gameplay and menu navigation. It is somewhat up to the user to interpret the these and use them appropriately although they are somewhat intuitive, so less hand holding is needed provided the user has some knowledge as to how to use the MYO armband. The main menu is then loaded by using the double tap gesture and the user is notified of this as shown.



The pause menu is accessed by performing the double tap gesture in the game scene and is resumed in the same manner. It contains the controls for the game in order to keep the user informed about the gestures used for movement and menu navigation.



Once the player has collided with a ghost the game over screen is activated as shown and they can perform the double tap gesture to go back to the main menu.



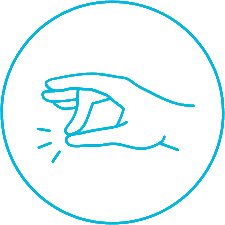
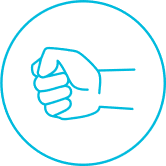
**Gestures identified as appropriate for this application:**The gestures initially selected for this project were wave in and out for menu navigation, fist for selection and using the gyroscope to move manoeuvre Pacman through the maze. We chose to have the wave in and out gestures move through the buttons on the menu because they are definitive gestures, another reason was because we chose to have the buttons laid out horizontally rather than the traditional vertical alignment to better suit this application and gestures.

The gestures selected for the menu navigation were chosen due to their simplicity and overall convenience to the user and to avoid overlap in use, the wave in and out gestures allowed the user to move left and right respectively depending on which arm the user was wearing the armband. The fist gesture was used for to select a button because it is like grabbing/selecting an object.

We identified these gestures as appropriate as they were intuitive and simple in theory. The issue we had using the gyroscope was that the readings were inconsistent along all the axes and had little to no bearing in the space used, when moving along the X axis the Y would dramatically change. We researched and experimented different techniques in order to use the information from the gyroscope like taking Euler angles from the Y axis to only move the player up and down while using a dead zone and using the accelerometer to the application to read the gyroscope and use the wave in and out to move left and right but due to the inconsistent readings this was unsuccessful. We then tried using the rotation value along the Z axis to move the player up and down, meaning the player would rotate their hand or arm in order to move upwards or downwards. This proved unsuccessful so we decided to use the fist and finger spread gestures to move the player up and down because these gestures are defined in the MYO by default and so they are more responsive than any other means of control we have experimented with using the MYO armband.

In the game itself the player can perform the double tap gesture to pause and resume the game we decided to use this gesture for the pause because it is difficult for the user to do it by mistake so the player will not have to stop and start and have fluid gameplay and in the game over screen the player can perform the double tap gesture to go back to the main menu, we decided to use the double tap for mainly navigation within the game itself in order to avoid more overlap in gestures.

We considered using speech recognition but decided against this because of the latency and possible misunderstanding of input that we would encounter as a result and we deemed it an inappropriate form of control for this classic arcade game.

**Hardware used in creating the application:** For this project we used a MYO armband. We chose to use the MYO because of its availability and that it used a variety of inputs with predefined gestures and a gyroscope, our initial plan was to use the gyroscope for movement along all axes.

We chose the MYO over the leap motion controller because of the MYO’s ability to navigate the menus using gestures as apposed the leap motion just tracking the joints on the users hands, we felt that the leap motion would be somewhat overkill in terms of information taken from it meaning that the leap motion would be overly sensitive and less definitive in its gestures to control the player.

The MYO also had a package made for unity meaning that there were resources freely available and easy to access and use. The MYO armband allows the wearer to create a custom profile, meaning the user can record themselves doing the predefined gestures in order to make the armband more responsive as it has a frame of reference as to what to expect for an individual user.

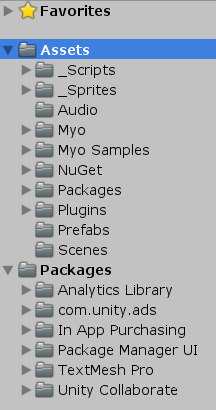
The Kinect would use full body skeletal tracking to control the player and we considered this but decided against it due the number of Kinects available would mean it would be less time for testing and debugging.

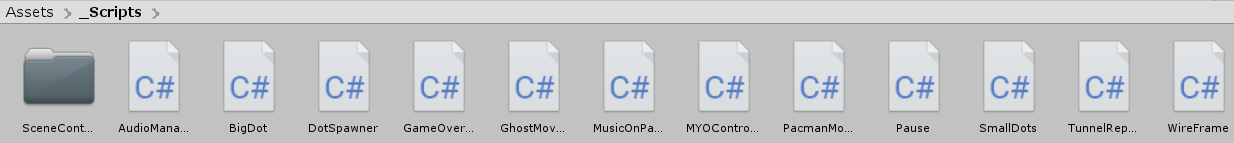


**Architecture for the Solution:**

Our solution was constructed in a systematic process. Firstly, a blank unity environment was created. Next, we created the 2D maze in which Pacman resides, as well as implementing player movement and animations. Next, we incorporated a basic input model which allowed the user to control Pacman using the keyboard, and later mapped these inputs to a digital keyboard. From this digital keyboard we mapped the MYO armband, using the intuitive gestures we described in the section previously.

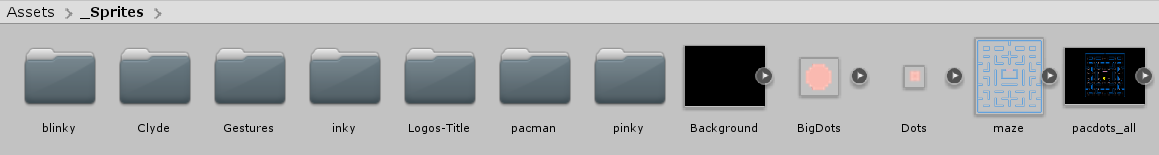
**The architecture of our solution (in Unity) is as follows:**



**Scripts:** 

A folder containing the various scripts used throughout the application. Every facet of the game logic is present here, from audio and animations to scene logic, geometry and objects present in-game.

**Sprites:**



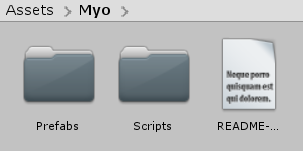
A folder containing the various sprites used in the game world. From player/ghost sprites to the geometry for the maze itself. Box colliders were applied to this particular sprite in order to facilitate the physical boundaries of the game world.

**Audio:**



A folder containing sound bites used in game. These clips are set to loop, and a trigger to fire them is incorporated in the movement script.

**MYO:**



This folder contains the core elements of MYO armband technology, with both the Prefab and Script folders being key.

The Prefab folder contains the both MYO Hubs 1 and 2. The first hub is to be used with one MYO Armband only. Conversely the second hub is for 2 MYO Armbands, although utilizing this secondary hub is less common.

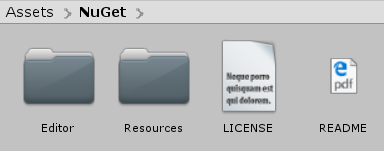
The script folder contains the ThalmicMyo script, which is essentially the connector for all MYO functionality.

**MYO Samples:**



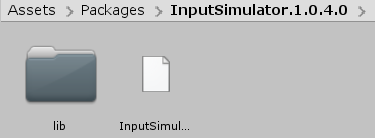
This folder contains sample materials, scripts and a sample scene to show the capabilities of the MYO Armband. They also serve as handy references for debugging, provided the code within is unchanged from the original MYO SDK.

**NuGet:**



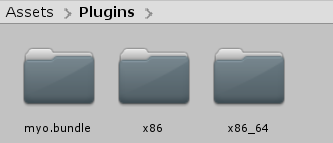
This folder contains all the necessisary assets for the NuGet package manager. It was from this manager that we downloaded and installed our digital keyboard (Input Simulator).

**Input Simulator:**



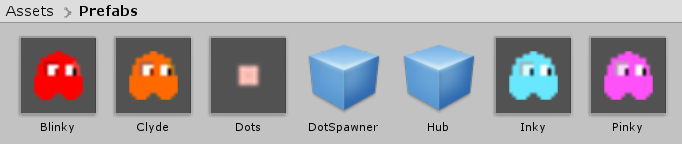
This is the digital keyboard we utilized when mapping MYO Armband gestures to specific movement inputs. As such, this made manual keyboard input essentially redundant, as we would be using the gestures from this point onwards. As this first method of input was not conflicting with the MYO Armband gestures we decided to keep it in, as it wasn’t doing any harm and helped with debugging.

**Plugins:**



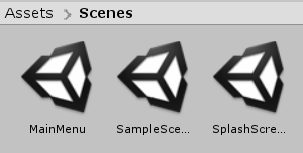
Plugins for the MYO Armband and associated scripts, for MacOSX (in the bundle folder), or for differing processor architectures.

**Prefabs:**



A folder containing the prefabs created for our application. Prefabs for ghosts, dots and the MYO Hub are all contained within.

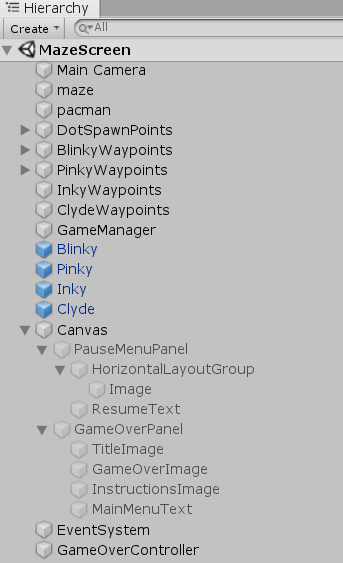
**Scenes:**



A folder containing the three scenes we used, including the Main Menu, Sample Scene (The maze itself) and the Splash Screen (which is shown on start-up).

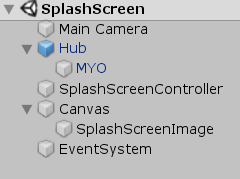
We will take a closer look at each scene hierarchy below:

**Main Menu: Game:**

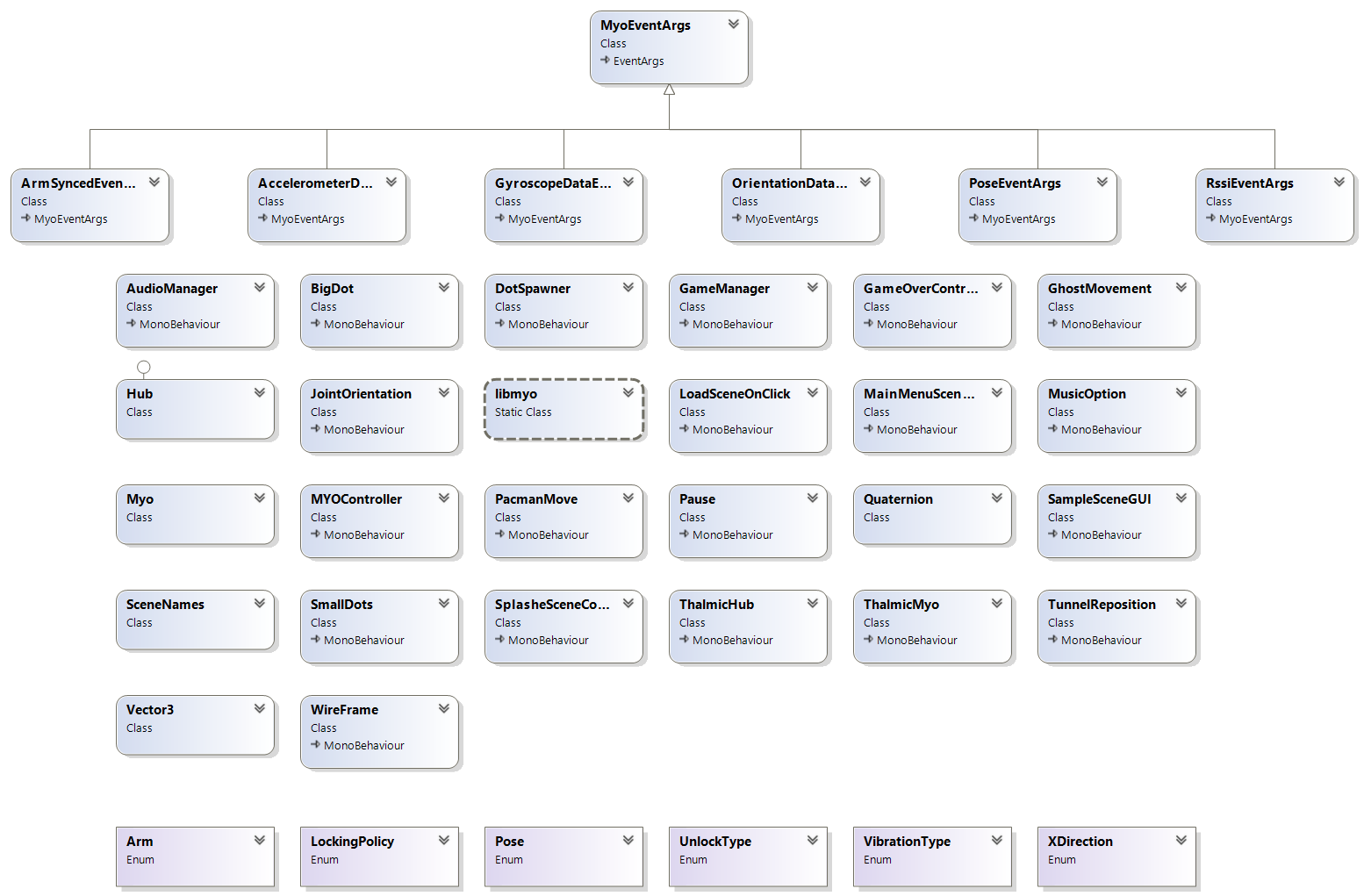
The main menu scene is used as a means of navigation at the start of the game, and consists of logos, sprites and three buttons. Each of these buttons are labelled “Play”, “Controls” and “Quit”, respectively. “Quit” quits the application, “Controls” brings up the control panel and “Play” brings the user to the next scene seen on the right.

The maze scene hierarchy consists of all the elements present in the scene at run time. The four main components of the maze scene are the maze, the dot spawn points, waypoints and the Pacman sprite that the player controls. The scene also contains a canvas for the pause menus and game over panel where the user can go back to the main menu.



The Splash screen contains the MYO game object and the audio controller allowing them to be carried across scenes. The MYO object is tagged so that it can be found by other object in other scenes at run time.

**Exploded class diagram:**



**Conclusions:**

We have derived many conclusions from this project, and as such we found it helpful to divide them into two categories, Pros and Cons.

**Pros:**

* An enjoyable experience overall. From game design, implementation and testing to the MYO integration. The process itself was well established and laid out, and the development became intuitive.
* Using the MYO itself was a fun experience, as we haven’t had much hardware experience thus far.
* The visualization of EMG readings and vibration feedback were particularly interesting. (See <http://diagnostics.myo.com/> for more!)

**Cons:**

* Many readings were erratic and contrary to expected results. This was particularly evident in the erratic readings we received from the MYO concerning the Y-Axis, as it was making readings for movement awkward. Furthermore, Z-Axis readings for rotation were also erratic, leading to us taking an alternative approach to control.
* Syncing and resyncing issues with the MYO continued throughout development, though creating a custom user profile helped alleviate some of these problems.
* The Raycast method we implemented was somewhat troublesome, as it used the line cast from Pacman to determine if he could proceed that direction. We found that the sprite was moving too far from a single input, meaning they bypassed avenues of travel.
* Gesture recognition between open hand and fist was erratic, depending on the user.

An enjoyable experience overall, as it challenged our thinking of how to incorporate the 3D technologies of the MYO Armband into a 2D game environment. This project also gave us an excellent insight into how we used the hardware, and how we could use the hardware again in a future project of differing specifications.

We found the MYO Armband to be reasonably suited to this application. While there were some teething problems throughout development, we found the MYO Armband to be quite receptive, not only to the gestures we ended up using but also throughout testing. Ideally, we would have liked to use the gyroscope functionality for control, rather than the gestures. Following this failure, we found the gestures to be a fine substitution for the gyroscopic controls.

**What could we have done differently?**

* We would have changed how the PacmanMove script was implemented.
* How the game environment was constructed could have been more streamlined. Rather than hardcoding or placing individual elements, we could have constructed it programmatically.
* Implementing more gameplay mechanics overall, as we were more focused on the MYO integration and Gesture aspects.
* The scale of the maze itself could have been reduced somewhat.
* In retrospect we could have looked at the Leap Motion controller, as it seemed to offer a viable means of control. From other student presentations we noticed good pattern and gesture recognition, but in quite a narrow field of detection.
* From other student presentations we also noticed the speed and accuracy of voice controls, but felt it still wasn’t appropriate for our needs.