

Airplane Simulator

Gesture Based URI Development for Bsc (Hons) of Science in Computing in Software Development

March 2020

**Authors:**

Vyacheslav Lukyanov

G00339839@gmit.ie

Marian Ziacik

G00340481@gmit.ie

**GitHub URL** :

https://github.com/majo-z/Gesture-Based-UI-Project

**Table of Contents**

[*Requirements* 2](#_Toc37184725)

[*Purpose of the application* 3](#_Toc37184726)

[*Hardware used in creating the application* 8](#_Toc37184727)

[*Gestures identified* 10](#_Toc37184728)

[*Architecture for the solution* 12](#_Toc37184729)

[*Conclusions & Recommendation* 16](#_Toc37184730)

[*References* 17](#_Toc37184731)

[*Appendix A* 18](#_Toc37184732)

[*Appendix B* 18](#_Toc37184733)

*Requirements*

* Development of a Natural User Interface application
* Local implementation of the application using gestures to interact with it (for example, an application using voice control, or implementation of a solution taking advantage of hardware like the Raspberry Pi, Myo Armbands, Leap Motion Controllers, Kinect, HoloLens, Durovis Dive, Arduino, Lego Mindstorms etc.)
* Reproduction of a classic game or system using a gesture-based interface
* The programming language of choice

The project should include the following headings (including all references as evidence of the research):

* Purpose of the application - design of the application including the screens of the user interface and how it works.
* Gestures identified as appropriate for this application - providing a justification for the gestures that can be incorporated into the application
* Hardware used in creating the application - the purpose of each piece of hardware should be given with a comparison to other options available
* Architecture for the solution - the full architecture for the solution, including the class diagrams, any data models, communications and distributed elements that are being creating
* Conclusions & Recommendations – learning outcome from the project and the associated research, different approaches, critical evaluation of the project

*Purpose of the application*

The main aim of this project is to showcase skills we have learned over our 4-year course in GMIT.

The project is a reproduction of a Custom Airplane Physics designed by Indie Pixel (Technical Artist / Programmer / 3D Artist), that can be initially controlled by keyboard and Xbox controller. Our goal is to implement an additional layer of control by using a Myo Armband hardware, so that the airplane can be controlled by the Myo Armband gestures.

The application has been created using Unity cross-platform game engine (version 2019.2.19f1) and C# language; and can run locally on a device with Windows 10 installed. The application hasn’t been tested on Apple device.

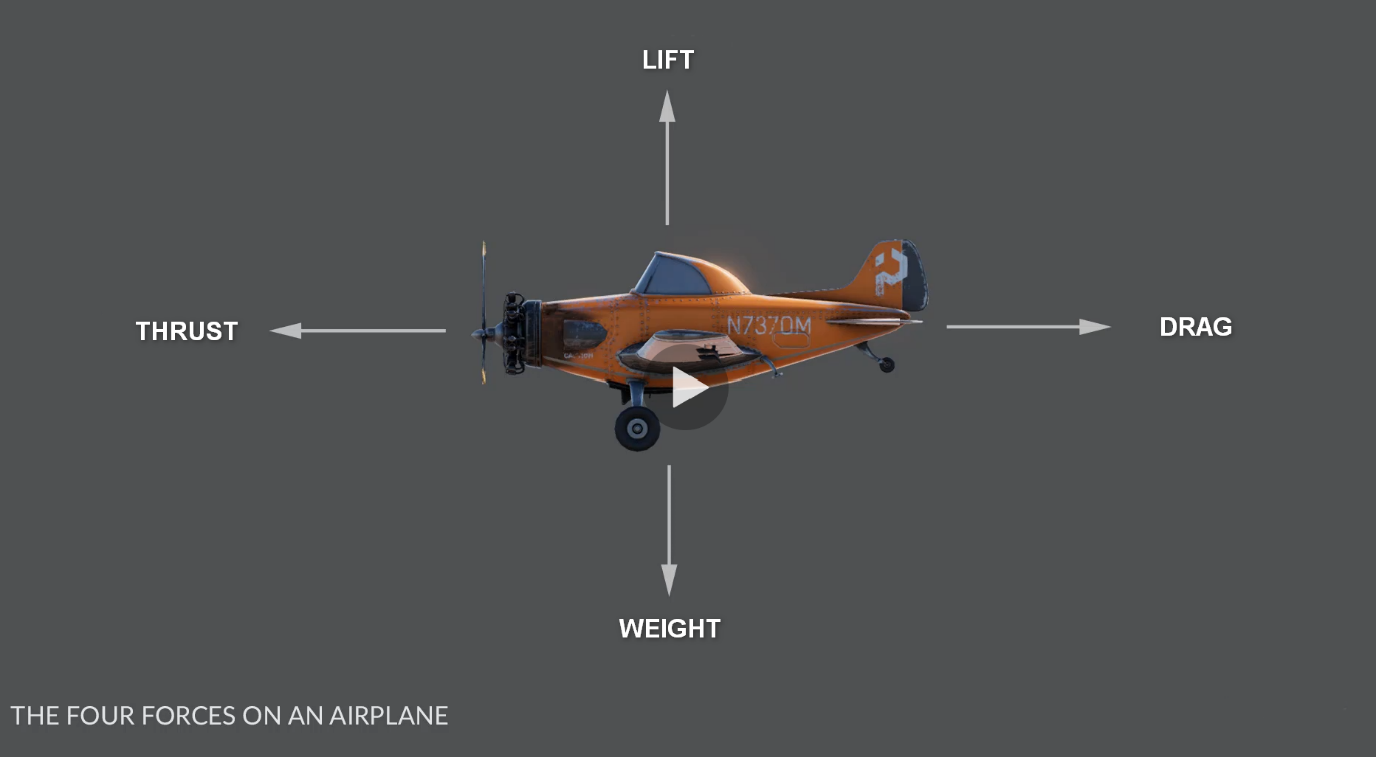
**About flying**

Flying a plane can be a difficult task as the pilot has to, besides controlling the airplane, learn to understand the weather, master crosswind and gusty wind landings, learn to tune in the correct radio frequencies, understand regulations, aircraft specifications and limitations etc. The pilot must always be attentive and notice what is going on around at all times, as being situational will help to avoid many aircraft accidents.

**Four acting forces on an airplane**

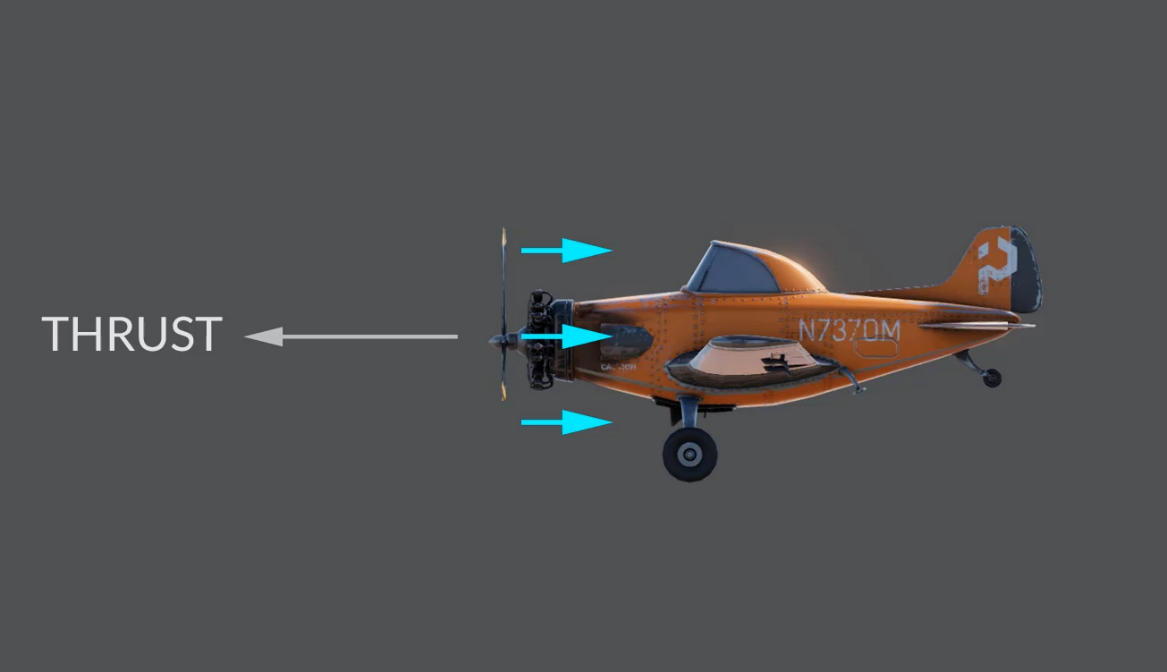
There are four acting forces on an airplane - lift, drag, weight and thrust. Maintaining a steady flight requires a balance (equilibrium) of all the forces acting upon an airplane. In a straight level flight, lift equals weight and thrust equals drag when the plane flies at constant velocity. It means that the opposite forces of flight are equal.

* If the lift becomes greater than weight, then the plane will accelerate upward
* If the weight is greater than the lift, then the plane will accelerate downward
* If the thrust becomes greater than the drag, the plane will accelerate forward
* If the drag becomes greater than the thrust, the plane will decelerate



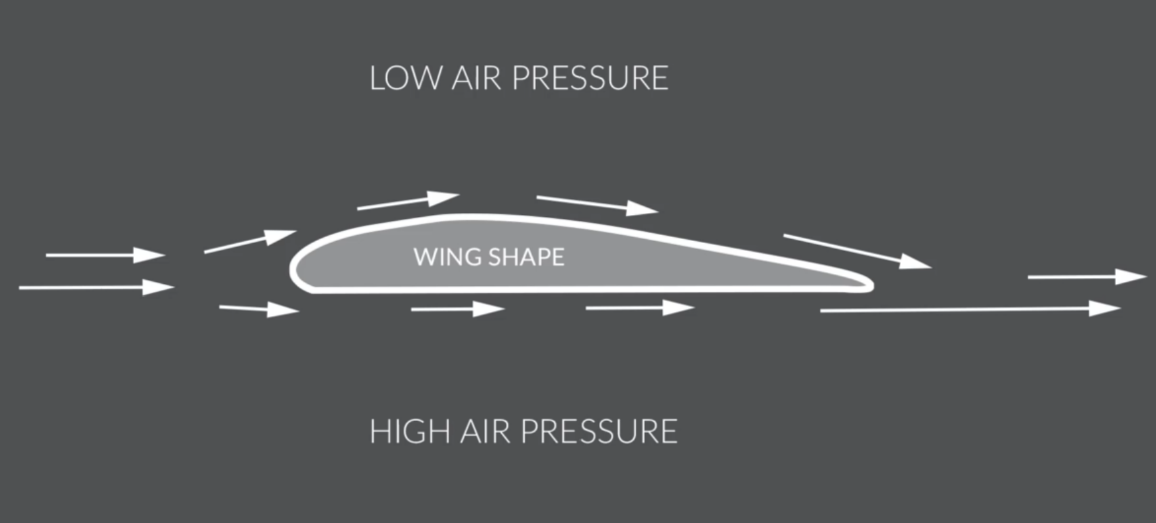
To put it simply, **thrust** force acting opposite to the **drag** force (and vice versa) and **lift** force acting opposite to the **weight** force (and vice versa).

The engine creates thrust and moves the plane forward. When thrust is greater than drag, the plane accelerates according to Newton's Law of motion, “The acceleration of an object is directly proportional to the net force acting upon it. The constant of proportionality is the mass.”. By the design an airplane must be able to accelerate vertically upwards, meaning the thrust must be greater than the weight and drag combined. When the thrust is lowered or engines of an airplane quit, drag slows the plane down.

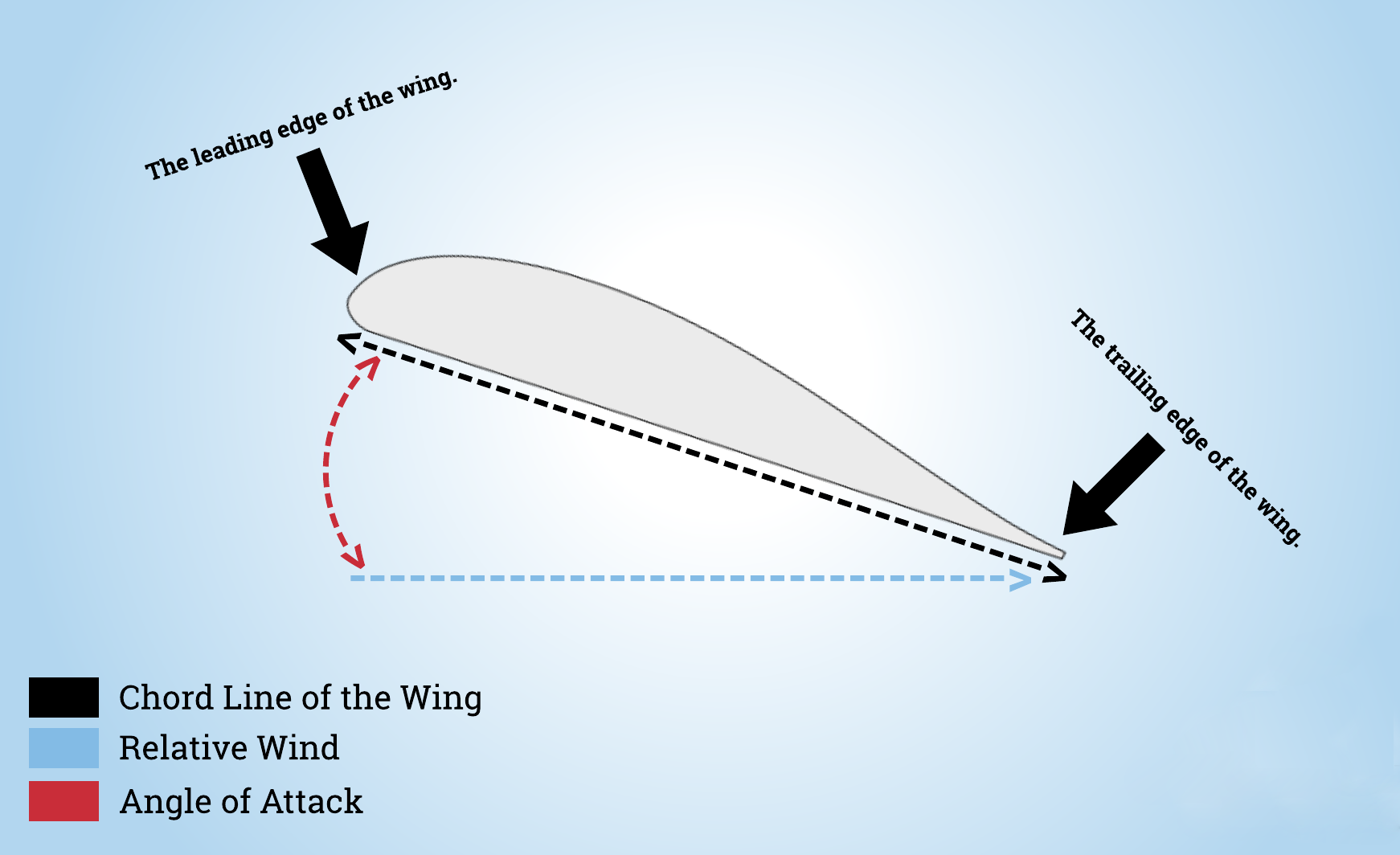


Newton's Laws and Bernoulli's Principle generate lift. When lift equals weight, the airplane can fly. Excess weight requires more lift, so heavy airplanes are more difficult to get off the ground. Lighter airplanes with less weight require less thrust. According to Newton's Laws of weight, mass and gravity, “The weight of an object is defined as the force of gravity on the object and is calculated as mass times the acceleration of gravity, w = mg, its SI unit is the newton.”

Bernoulli's Principle helps explain that an aircraft can achieve lift because of the shape of its wings. They are shaped so that the air flows faster over the top of the wing and slower underneath. Fast moving air equals low air pressure while slow moving air equals high air pressure. The high air pressure underneath the wings will therefore push the aircraft up through the lower air pressure.

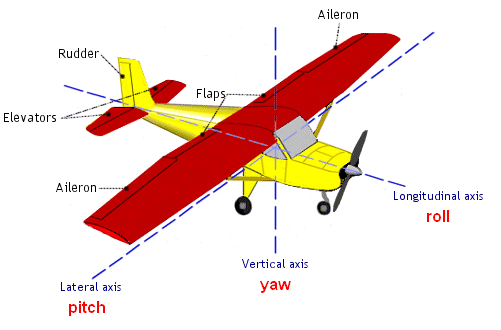


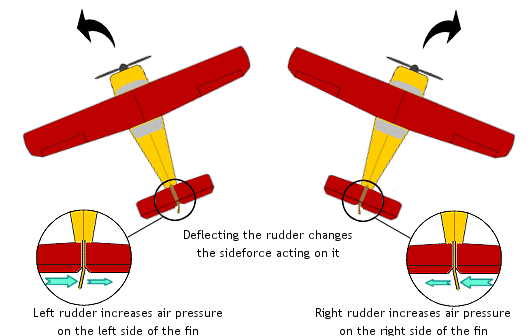
Another important property in aerodynamics is angle of attack. It specifies the angle between the chord line of the wing of a fixed-wing aircraft and the vector representing the relative motion between the aircraft and the atmosphere. When the angle of attack is too great, in relation to the velocity of the aircraft, lift on the wing is reduced.



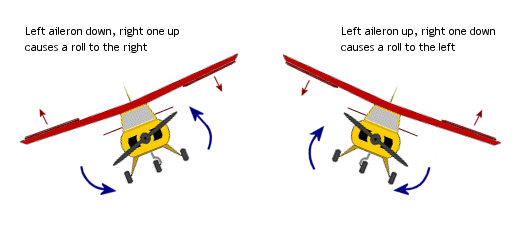
**Controlling the flight of an airplane**

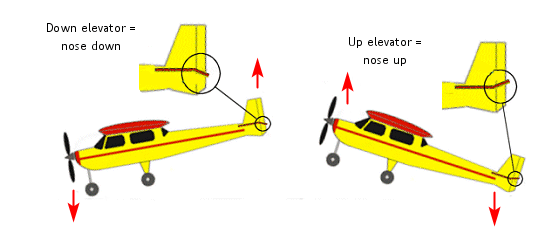
Rudder, elevators, flaps and ailerons are 4 parts of an airplane that are very important. They allow an airplane to freely rotate in three dimensions, called pitch, roll and yaw, i.e. make the plane to change the direction, descend or lift.



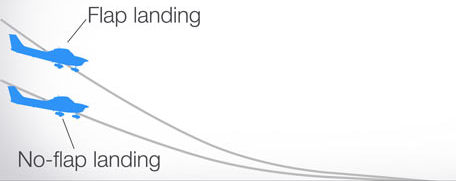
Rudder rotates left or right to control the yaw of the plane. When the rudder is turned to left side, the plane moves left and if it is turned to right side, the plane moves right. The airplane's nose is pointed in the same direction as the direction of the rudder.

Ailerons raise and lower the wings to control the roll of the plane. When the aileron on one side of the wing goes up, the other wing's aileron goes down, which causes the plane to roll left or right.



Elevator raise and lower to control the pitch of the plane, i.e. make the plane rise or descend. When the elevators go down, the plane goes down and when they go up, the plane goes up.

Flaps are important for landing. They lower to help the plane slow down. They also go up about 10 degrees when taking off for extra lift, so they have time to recover if something goes wrong.

****

*Hardware used in creating the application*

**What is** **Myo Airband**

Myo Airband was created by Canadian wearables company Thalmic Labs and unfortunately discontinued in 2018. Although discontinued, it is still applicable in today’s applications, devices and operating systems. It is compatible with Mac, Windows, iOS, and Android devices through Bluetooth Smart.



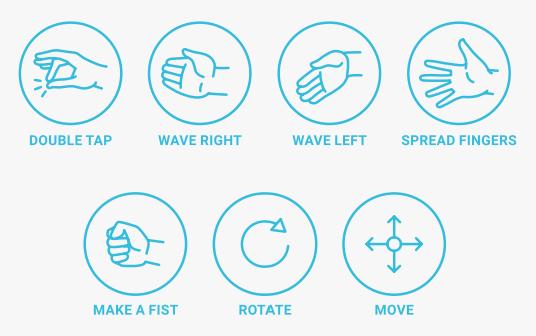
It combines two powerful technologies.

One is EMG sensor, also known as electromyography sensor. There are 8 EMG sensors on the armband and the sensors measure the small electrical signal that are generated on the surface of the skin when the muscle is activated. The technology has been around for many years and is commonly used to diagnose muscle and nerve disorders, as a control signal for prosthetic devices, in operating room setting, sports health etc.

The other technology is IMU, or Inertial measurement unit, that measures a visibility of the motion and orientation of the device. IMU is commonly used to manoeuvre aircraft by calculating attitude and heading reference system, to measures and reports a body's specific force, angular rate, linear velocity and position relative to a global reference frame etc., using a combination of accelerometers, gyroscopes and magnetometers.

The two technologies are packed in to the armband that is worn on the forearm and understand the subtleties between various muscle movements and the individual fingers. This enables gestures, such as make a fist and rotate, which is used by most media applications in the Myo market to control the volume. The gesture control armband can also be used with any existing Windows and Mac applications. For instance, a piece of software can be mapped out the gestures to keyboard commands for to have a full control of that software.

There are various poses there are available in the SDK and they are the following: double tap, wave right and left, spread fingers, make a fist, rotate and move.



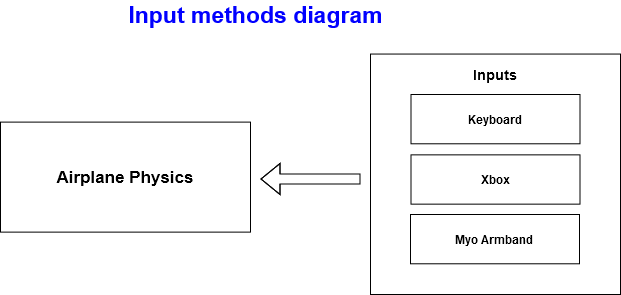
**Other options**

Other option available for the project would be to navigate the plane by using voice control. However, this option has been ruled out soon after exploring, because navigating the plane is complex task and requires use of different gestures at the same time. For example, lifting the plane from the ground requires throttle up and pitch down gestures to be used at the same time, or flying the plane requires throttle up/down, pitch, yaw and roll gestures to be used at the same time. Another option would be using Leap motion that interact with computers by using natural hand movements and gestures by using two monochromatic IR cameras and three infrared LEDs to a distance of about 1 meter. However, as the Myo Airband has been introduced early in the labs and we have had a chance to interact with it, we decided it was the best option for implementing it to our application. Another reason is that the developers at Thalmic Labs that released the Myo, presented a demo where they used the Myo armband to control a Quadcopter AR Drone. It shows capabilities of the Myo as how to design a mapping from gestures to actual commands for the drone, which is similar to our application, because it is a flying object that uses similar gestures. More information about the drone gesture implementation can be found in attached document (Appendix B) and demonstration video can be found on the following link: <https://www.youtube.com/watch?v=aXoDK0EHdzM>

*Gestures identified*

The project is originally controlled by keyboard and Xbox controller. We have chosen to use the Myo Armband as solution for our application, because we are not restricted by the gestures that can be implemented and fits perfectly to application like this. Controlling of the plane by using keyboard and Xbox doesn’t have any significance to us, because we have focused on implementing the armband gestures, it is only to be mentioned for the purpose of understanding how to control it if a person wants to download and play the game, but has no access / does not own the Myo Armband.

**How to control the airplane in our application**

**

Our application is a Cessna 152 airplane simulator by its design, physics, characteristics and performance. The user has 3 options to interact with the plane, by using a keyboard, Xbox controller and most importantly Myo Armband.

**Control the airplane using keyboard:**

The user interacts with this airplane by using the following keyboard keys:

* Up arrow - throttle up
* Down arrow - throttle down
* Left arrow - yaw left
* Right arrow - yaw right
* W - pitch down
* S - pitch up
* A - roll left
* D - roll right
* F - landing flaps down (2 steps)
* G - landing flaps up…from down position (2 steps)
* Spacebar – apply brakes
* C - camera flip (object, cabin)

**Control the airplane using Xbox controller:**

The user interacts with this airplane by using the following Xbox buttons:

* Right stick up - throttle up
* Right stick down - throttle down
* Right stick left - yaw left
* Right stick right - yaw right
* Left stick up - pitch down
* Left stick down - pitch up
* Left stick left - roll left
* Left stick right - roll right
* Right bumper - landing flaps down (2 steps)
* Left bumper - landing flaps up…from down position (2 steps)
* B button – apply brakes
* Y button - camera flip (object, cabin)

**Control the airplane using Myo Airband:**

The user interacts with this airplane by using Myo Armband hardware attached to the forearm, that sends a signal to the application. Based on the implemented gestures the user can than control the airplane by using following distinguishable movements - waving left and right, double tapping, fingers spreading, making a fist, raising the arm forward and backward, sideways and rotating the arm.

* Spread fingers - throttle up

A close up of a logo

Description automatically generated

* Make a fist - throttle down

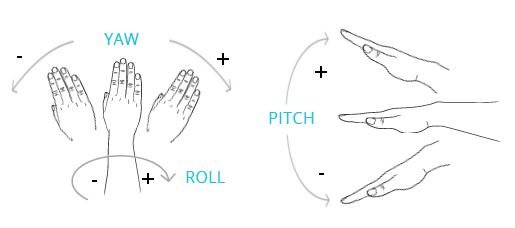
A close up of a logo

Description automatically generated

* Double tap – center the plane
* A close up of a map

  Description automatically generatedRaise arm and move it left horizontally - yaw left
* Raise arm and move it right horizontally - yaw right
* Lower arm down - pitch down
* Raise arm up - pitch up
* Raise arm and rotate it left - roll left
* Raise arm and rotate it right - roll right
* Wave left - landing flaps down (2 steps)
* Wave right - landing flaps up…from down position (2 steps)

The below picture can better explain the pitch, yaw and roll movements



*Architecture for the solution*

**Keyboard and Xbox input**

Keyboard input script is located in:

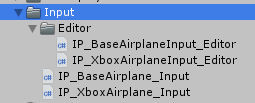
../AirplaneSimulator/Assets/AirplanePhysics/Code/Scripts/Input/ IP\_BaseAirplane\_Input

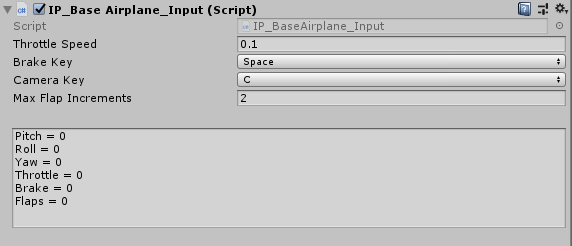
Xbox input script is located in:

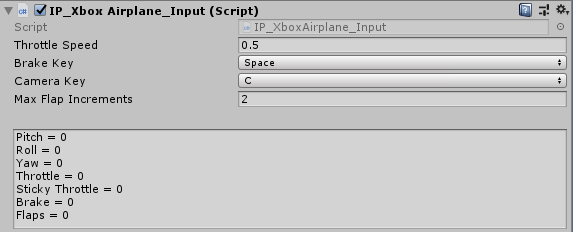
../AirplaneSimulator/Assets/AirplanePhysics/Code/Scripts/Input/ IP\_XboxAirplane\_Input

…and are attached to the airplane object.

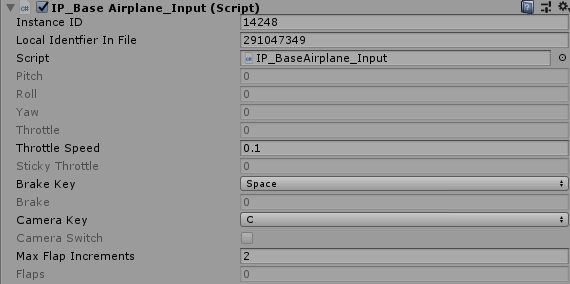
IP\_XboxAirplane\_Input inherits from IP\_BaseAirplane\_Input, meaning some controls, like pitch and roll remain unchanged

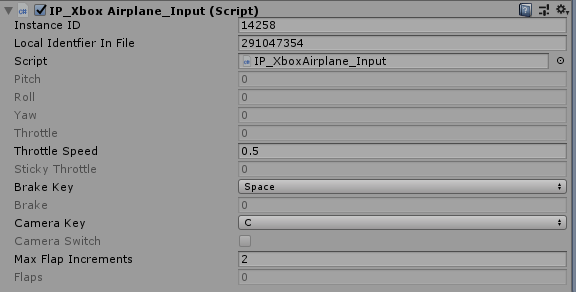


IP\_BaseAirplaneInput\_Editor and IP\_XboxAirplaneInput\_Editor both inherit from Editor class, that are overridden by custom editor of type IP\_BaseAirplane\_Input and IP\_XboxAirplane\_Input respectively. The editors serve as input visual aid.

****

The same action can be achieved by switching the Inspector to debug mode



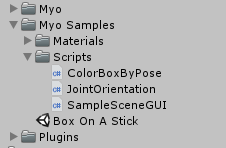


Key mapping for both input methods is done in Project settings/Input

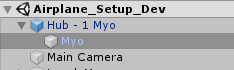
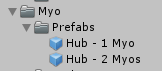
**Myo Armband input**

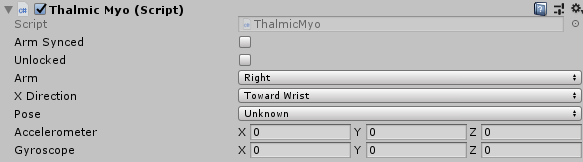
Initial set up for the project includes installing the Myo sdk software and pairing it with the dedicated usb dongle. Myo sdk allows the user to connect the armband to computer to access Myo guides, personalization options, gesture recognition, and more. The guide will show the user how to wear the armband properly and how it works, in addition to leading the user through connecting and syncing.

To use the Unity with Myo Armband, MyoUnity.unitypackage has to be added to the project. This is done in Unity by clicking at Assets/Import package/Custom package -> myo-sdk-win-0.9.0 that has been downloaded and saved to hard drive/ MyoUnity.unitypackage

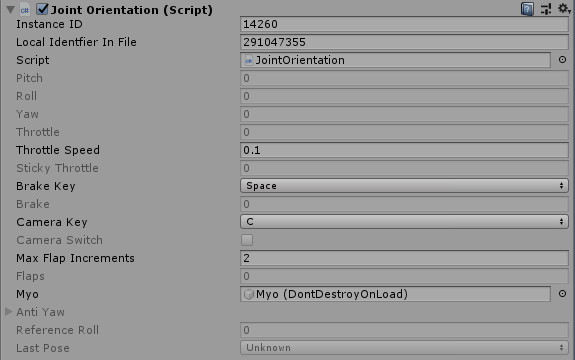
Screenshot of the myo sdk successfully installed

Screenshots of Myo hub from Myo prefab folder attached to the scene. Here the user can set up different parameters, such as arm sync, directions, poses etc.





Joint orientation script is attached to the game object (picture of script in Inspector’s debug mode). It inherits from IP\_XboxAirplane\_Input



**Technologies**

* Myo Armband
* Unity Engine
* Visual Studio 2017
* Adobe Photoshop
* C#

*Conclusions & Recommendation*

The developers at Thalmic Labs that were behind the Myo Armband development, showed capabilities of the Myo gestures to command a Quadcopter AR Drone. We chose airplane simulator based on these capabilities and decided it was a good game to try to implement with gestures. The rules are easy to understand from a logic perspective and we were able to change the flying track a bit to offer a bit more variety in our scoring system. We felt it would be a good demonstration of the skills we have picked up over the last 4 years of college. We also explained keyboard and Xbox controls for people who want to play the game but have no access to Myo armband hardware.

If we were to do this project again there are a few things we might look to change:

**Time management**

We feel that once we had a concrete idea of what our project would be, that the time management became easier. Before settling on the game, we were meeting up and going through ideas to find one that seemed right for this project.

Although we do feel that we handled the development cycle better in this project, we have progressed in our development as software developers and our time management skills.

This is still an area where we could improve further for future projects.

**Gameplay**

We would possibly look at implementing other flying object, such as helicopter to see the difference in physics and gesture control.

**Issues**

* Myo Accuracy
* Gesture Limitation
* Myo production has officially ended as of Oct 12,2018

**Credits:**

We thank to Indie Pixel for providing a beautiful base for our game, his projects can be found on this website: [https://www.indie-pixel.com](https://www.indie-pixel.com/)

*References*

<https://support.getmyo.com/hc/en-us/articles/203398347-Getting-started-with-your-Myo-armband>

<https://venturebeat.com/2018/10/12/amazon-backed-wearables-company-thalmic-labs-kills-its-myo-armband-teases-new-product/>

<https://sites.google.com/site/thebasicsofaviation/rudder-empennage-and-ailerons>

<https://www.grc.nasa.gov/www/k-12/UEET/StudentSite/dynamicsofflight.html>

<https://www.mansfieldct.org/Schools/MMS/staff/hand/flight4forcesoverview.htm>

<https://www.mansfieldct.org/Schools/MMS/staff/hand/Lawsnewton2law.htm>

<https://study.com/academy/lesson/newtons-laws-and-weight-mass-gravity.html>

<https://en.wikipedia.org/wiki/Bernoulli's_principle>

<https://en.wikipedia.org/wiki/Angle_of_attack>

<http://hyperphysics.phy-astr.gsu.edu/hbase/mass.html>

<https://howthingsfly.si.edu/forces-flight/four-forces>

<https://www.youtube.com/watch?v=Q5kcXojBRtY>

<https://en.wikipedia.org/wiki/Cessna_152>

<https://github.com/thalmiclabs>

<https://support.getmyo.com/hc/en-us>

<https://www.youtube.com/watch?v=aXoDK0EHdzM>

<https://www.youtube.com/watch?v=3VLoGSVORjY>

https://www.youtube.com/watch?v=UL\_pDatlLOg

<https://www.robotshop.com/eu/en/myo-gesture-control-armband-black.html>

<https://www.popularmechanics.com/flight/a12454/difficult-descent-4-trick-kinds-of-airplane-landings-15686244/>

<https://www.seeedstudio.com/blog/2019/12/27/what-is-emg-sensor-myoware-and-how-to-use-with-arduino/>

<https://en.wikipedia.org/wiki/Leap_Motion>

<https://developer-archive.leapmotion.com/articles/designing-intuitive-applications>

<https://en.wikipedia.org/wiki/Inertial_measurement_unit>

<https://www.quora.com/What-makes-flying-a-plane-so-difficult>

<https://books.google.ie/books?id=49uHAwAAQBAJ&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false>

*Appendix A*

**GitHub URL** : <https://github.com/majo-z/Gesture-Based-UI-Project>

**Unity version:** 2019.2.19f1

**Other software:** MS Visual Studio 2017 & 2019, MS Visual Studio Code, FastStone Image Viewer, MS Word 365, XODO PDF Viewer

**Hardware:** Myo Armband,Laptop (OS Windows 10)

*Appendix B*

Implementing Gesture Control for a Quadcopter.pdf