Title: Development of Control and Interface Systems for a Flight Simulation Experience

# Abstract

This report covers the design and development process of a control and interface system for a 'Flight Simulation' ride experience. The project is being carried out in collaboration with other final-year students responsible for the ride’s accessibility features, as well as the Royal Aeronautical Society, which plans to use the completed ride at outreach events such as school visits and expositions.

The specific programme this project is a part of is the “Falcon 2 programme” [1], a STEM outreach programme for schools, colleges and youth groups aged (6-19) to contribute their design and engineering skills to develop and build a real-life mobile flight simulator which will travel to Special Educational Needs and Disability (SEND) schools and public events around the UK to introduce people from all backgrounds to the wonder of flight.

The final ride is designed to be a highly immersive experience, allowing the user to feel as though they are truly piloting an airplane. The chair moves in sync with the plane’s motions, and the user can observe their surroundings through a wide monitor (or VR headset ideally), interacting with them as if they were inside the airplane.

# Introduction

First and foremost, some research was done into existing solutions . A high-level outline of the ride control system is shown in **Fig 0.0**.

A diagram of a computer system

Description automatically generated

**Fig 0.0: High level flowchart showing how different components within the ride communicate with each other.**

The core goal of this project is to develop a comprehensive set of ready-to-use software tailored for the RaeS Falcon 2 programme that enables the virtual flight simulator (or potentially a variety of simulators) to interface seamlessly with the physical system.

Additionally, the project aims to provide direct human operator manual chair control through a peripheral device. Furthermore, the software will prioritize operational safety by incorporating extensive failsafe mechanisms throughout the system such as stopping the simulation if the wheelchair dock sensor is released or if the chair muscle attachment is released. As far as this project is concerned, the sensors are just incoming signals. Details on actual hardware wont be delved on.

A significant focus will also be placed on quality-of-life features and aiming to create a fully complete system, ensuring it is user-friendly and accessible to individuals who may not be familiar with the system's inner workings.

# Chapter 1-Preliminary research

According to some pre-liminary research, the main simulator of choice for the project is “X-Plane”.

The client in charge of getting the telemetry of the plane from X-Plane (or Unity) is written in python and the communication protocols between software/hardware are done via UDP protocol. This includes the communication from X-Plane to the python client to send the plane telemetry as well as the communication from the python program to the PLC to control the mechanical chair PLC.

There’s some pre-existing work by MDX on a similar motion platform which can be adapted into the mechanical chair. The project is open source and can be found on Git Hub [2]. Its worth noting this project inherits several parts from the MDX rollercoaster ride [3]

So far the library of choice to interface with X-Plane is XPPython [4] which enables access to airplane telemetry and other data from the game. The project will later look into controlling the game through python to start the simulation but for now the main focus is getting data from X-Plane through the XPPython API.

The python client is in charge of several jobs, the most important of which is to translate the xyzrpy (xyz translation, roll, pitch and yaw) into actuator distances for the platform (inverse kinematics). The actuators in the chair are fluidic muscles, therefore for a given weight, the client sends out a given pressure value via UDP to actuate each one by the desired distance.

**Fig 1.0** shows a more specific software layout of the system.

A diagram of a flowchart

Description automatically generated

**Fig 1.0: High level flowchart showing a more specific software layout of the entire ride system**

The Unity User Interface is designed to serve as an interface for the operator, enabling easy control of the ride through a user-friendly system. This is achieved by establishing a connection between the Unity UI and the Python controller client, where the Unity UI primarily displays system information and relays control commands to the Python controller, which acts as the core control software for the chair.

The main reason for selecting Unity is its robust set of UI tools, which allow developers to create interfaces efficiently without the need to code every individual element. Another advantage of using Unity is its inherent capability to handle 3D simulations. This not only facilitates the creation of a well-rounded and intuitive UI but also enables the real-time display of a 3D digital twin of the chair. This feature allows the operator to debug and manually control the chair as needed, further enhancing the system's usability and functionality.

## Requirements

The following list shows what the final project software must be capable of achieving in conjunction with the actual chair system. This is shown in **Table 1.0**. Requirements with \* are “wish” requirements which are prioritized below the ones without an \*.

**Table 1.0: Table showing final project requirements**

|  |  |  |
| --- | --- | --- |
| Requirement number | Requirement Description | Is wish requirement? (\*) |
| 1 | Software is able to move mechanical chair in real time with the flight simulation. The ride should not cause motion sickness after at least 2 repeated uses. |  |
| 2 | Software comes with a basic user interface. |  |
| 3 | Software provides fail START/PAUSE/STOP functionality to control the ride simulation as required. |  |
| 4 | User interface is friendly and can be easily used by an inexperience operator. | \* |
| 5 | User interfaces allows operator to manually control the chair’s pose and see a side by side simulation of it. | \* |
| 6 | Software is allows support to connect chair to other flight sims aside from X-Plane. | \* |
| 7 | Final ride has VR support | \* |

It is important to note that the latency of the mechanical chair must be kept below 200 ms, as motion lag becomes noticeable to humans beyond this threshold. Latency exceeding 200 ms can lead to motion sickness, which is highly undesirable for a platform intended to attract users and immerse them in the experience.

These are the initial requirements. The project is likely to grow and become more detailed as the dev weeks go by and further requirements will likely be introduced as more and more of the project is cleared.

# Chapter 2-High Level Design

Once enough pre-liminary research was completed, the next stage was to design the high level outline of how the software is to function. **Flowchart 2.0** outlines the high level layer architecture design of how the python client is to interact with the simulator and PLC controlling the chair. The core aim of this chapter will be to finalize a basic skeletal system for the software to later evolve from.

A diagram of a software company

Description automatically generated

**Flowchart 2.0: High level outline of how python client is to interact with the simulator and chair PLC**

The scripts in the “TELEMETRY LAYER” will solely be in charge of obtaining the desired airplane telemetry data from their respective simulator. This modular approach leaves room in the future to work with other simulators by simply creating an appropriate “\_Getter.py” script. The controller layer code does not need to be changed in that case.

In this design, “Chair\_Controller.py” will handle most of the workload. This node (running script) will be in charge of receiving the airplane telemetry data in the aforementioned format (Chapter 1) and translate it into distance and subsequently pressure values to send to the motion platform. **Flowchart 2.1** further outlines how the Chair\_Controller.py script will operate.

A diagram of a control system

Description automatically generated

**Flowchart 2.0: State machine diagram of how Chair\_Controller.py will operate**

A state machine design approach was chosen to clearly define the actual operation of the script. Moreover a state machine description will allow to easily create a basic python GUI later on.

# References

[1]Falcon 2 programme homepage: <https://www.aerosociety.com/careers-education/schools-outreach/the-falcon-2-programme/>

[2] Motion Platform GitHub Page: <https://github.com/michaelmargolis/MdxMotionPlatformV3>

[3] MDX rollercoaster news page: <https://www.mdx.ac.uk/news/2024/12/new-scientist/>

[4] XPPython Homepage: <https://xppython3.readthedocs.io/en/3.1.5/index.html>