A3: Architectural Enhancement Report for the FlightGear Flight Simulator

CISC/CMPE 322/326

April 8th, 2024

Group 5: GoneFishing
Arshan Abdi (20298549)
Kenneth Laird (20222078)
Anson Liu (20232192)
Derek Ma (20109427)
Curtis Pike (20174323)

Abstract

When tasked with adding new features or enhancements to the current FlightGear application, we decided to implement AR/VR compatibility. This report outlines the implementation process with consideration to planning, testing and implementation. The report also includes a detailed analysis on affected stakeholders as well as teams that could be affected by the implementation of the new feature. This new feature should enhance user experience while also upgrading the application to further integrate new technology.

Introduction and Overview

In our previous assignments, we examined the conceptual and concrete architecture of the FlightGear software and developed a deeper understanding of the interactions between its subsystems. With that knowledge, we were tasked with implementing a new enhancement while updating our current architectural designs. Based on the requirements and the nature of the FlightGear software, we thought implementing AR/VR compatibility would make sense. This would enhance the user experience while also staying in line with the original purpose of the application. The AR/VR enhancement should be compatible with hardware already on the market rather than designing the new hardware piece ourselves. In order to successfully implement these changes, we had to look at various factors listed in the report including, but not limited to; subsystem interaction changes due to the new feature, risks and new testing plans, and effect on stakeholders and related parties.

This report will start with a detailed description of the proposed enhancement along with implementation plans as well as any possible effects associated with upgrading. The report will also show a SAAM Analysis outlining affected parties and comparing the pros and cons of implementing AR vs VR. Finally, we will cover the potential risks of the new enhancements along with potential tests to mitigate those risks.

Proposed Enhancement

The proposed enhancement for the FlightGear Flight Simulator is support for virtual reality and augmented reality to enhance the user immersion further. The reason this enhancement was chosen is due to the nature of the software as a training tool for new pilots. When using a controller and keyboard there is a limit of how productive it can be for training compared to the actual aircraft. The software is designed to be as realistic as possible, but ultimately is limited by its technology compared to the actual articles. The software can serve these purposes well as is, but with the addition of VR or AR the immersion that is already prioritized by the system can be further enhanced, which in turn increases the effectiveness of the training. The enhancement is also a very common feature in similar programs and would likely increase the casual user base as well as a natural result. As such, the VR implementation would increase immersion greatly compared to the AR, but would require much more equipment to set up comparatively.

To accomplish this, control compatibility will have to be implemented that allows the audio and visuals to be transmitted to the VR or AR headset and allow the controls to be

engaged with their respective controllers. This would require a decent amount of overhaul for the system's data displaying and control systems to accomplish but many of the existing systems could be repurposed, just requiring altercation to enable compatibility.

As such the first approach would be to implement VR compatibility to allow the software to work with commercial VR headsets. This would allow for the software to provide a realistic immersive experience of being inside the cockpit of an aircraft and allow the users to better train their flight control and awareness as pilots. Implementing this would require significant development effort, but much of it would be focused on the I/O interface subsystem that handles communicating data to and from the user. Also, the user requirements would be much higher, as the software is already quite demanding of the computer, the VR would require even greater processing power along with the purchase of a compatible headset.

The second approach would be to overlay the flight instruments onto the real world through an AR headset. The FlightGear software would then be tasked with rendering the controls and cockpit on top of the real world as captured by the camera. This would deliver a greater level of immersion than previously, but not to the same degree as the VR implementation. This would however cost less to develop and put less strain on the computer running the software comparatively. For both approaches a new subsystem would be added to handle the new VR or AR feature which would be called the "OpenXR API" for the first option. The AR option would be named according to the tools used to develop it but would interact with the other subsystems the same way. This subsystem would interact with the Main Build the I/O Interface to properly convey the information and cues to the user wearing the respective headset.

Approach to Implementation

The implementation will leverage the Microservices Architecture, which is popular for handling the variety of requirements needed for VR or AR enhancements. This architecture style facilitates scalability, allows individual service updates, and promotes robustness in the simulation environment. Each component, such as graphics rendering, environment simulation, and user interaction, will be developed and maintained as independent services.

The approach to implementation can be divided into these five key steps:

Step 1 is developing the VR or AR Interface that allows FlightGear to communicate with the respective hardware. In this respect the need to create VR or AR microservices that serve as the bridge between the FlightGear engine and specific hardware is required for features such as its tracking devices. These services will translate simulation data into VR/AR outputs and vice versa.

Step 2 is about enhancing the visual and interactive environment of FlightGear to support VR or AR capabilities. By Integrating advanced 3D rendering techniques and real-time environment updates to ensure that visual fidelity and frame rates are sufficient for immersive user experiences.

Step 3 is developing a system that allows users to interact with the simulation using VR/AR controllers and gestures. The method is to implement a control system microservice that processes user inputs from VR or AR hardware and translates them into simulation controls.

Step 4 is Testing and Integration to ensure that the enhancements work seamlessly with existing FlightGear systems and conduct extensive testing using both automated test suites and pilot user groups to validate the functionality and user experience.
Step 5 is about deployment and feedback loop, by deploying the updated FlightGear simulation and gathering user feedback for improvement for future updates. This would entail rolling out the enhanced version to a controlled group of users and gathering structured feedback through surveys and usage analytics.

SAAM Analysis

1. Identifying the major stakeholders of the proposed enhancement.

End Users: Pilots, aviation students, and anyone utilizing the FlightGear software for flight training or simulation purposes.
Project Team: Developers, engineers, designers, and other team members responsible for implementing the enhancement.
Project Managers: Those overseeing the development process, resource allocation, timelines, and overall project success.
Funders/Investors: Individuals or organizations providing financial support for the enhancement, such as sponsors, investors, or funding agencies.
Regulatory Bodies: Aviation regulatory agencies or organizations responsible for setting standards and regulations related to flight training software.
Suppliers/Partners: Companies or individuals providing hardware components, software tools, or other resources necessary for the enhancements
Training Institutions: Flight schools, educational institutions, or organizations utilizing the FlightGear software for training purposes.
Aviation Industry: Companies, organizations, and professionals involved in the aviation industry who may benefit from or be impacted by the enhanced training capabilities.
Software Users Community: The broader community of FlightGear users, including enthusiasts, developers, and contributors, whose feedback and engagement may influence the enhancement.
Accessibility Advocates: Individuals or organizations advocating for accessibility and inclusion in software design, ensuring that the enhancement is usable by all users, including those with disabilities.

2. Identifying, for each such stakeholder, the most important non-functional requirements (NFRs) regarding the enhancement.

End Users:
Usability: The software should be intuitive and easy to navigate, ensuring a smooth and enjoyable user experience.
Effectiveness: The enhancement should significantly improve training effectiveness, aiding in skill development and proficiency.
Reliability: Users rely on the software for accurate simulation and training, so it must operate consistently without unexpected failures.

Performance: The software should run smoothly without lags or delays, providing real-time feedback and responsiveness.

Project Team:

Maintainability: The codebase should be well-organized and modular, facilitating easier maintenance and future updates.

Scalability: The software should be able to accommodate potential increases in user demand and feature complexity without sacrificing performance.

Security: Protecting user data and ensuring the software is secure from potential threats or vulnerabilities is paramount.

Interoperability: The enhancement should seamlessly integrate with existing systems and technologies to maximize compatibility and ease of use.

Project Managers:

Manageability: Project management tools and processes should be in place to effectively track progress, allocate resources, and meet deadlines.

Cost-effectiveness: Ensuring that the enhancement is developed within budgetary constraints and provides value for the investment made.

Risk Management: Identifying and mitigating potential risks, such as technical challenges or delays, to ensure project success.

Funders/Investors:

Return on Investment (ROI): The enhancement should deliver tangible benefits and value to justify the financial investment made.

Marketability: Ensuring that the enhanced software meets market demands and remains competitive in the industry.

Regulatory Bodies:

Compliance: Ensuring that the software complies with relevant aviation regulations and standards, maintaining safety and reliability.

Suppliers/Partners:

Integration: Ensuring that any hardware or software components provided by suppliers seamlessly integrate with the enhanced software.

Training Institutions:

Adaptability: The software should be flexible enough to accommodate various training scenarios and curriculum requirements.

Customization: The ability to customize training modules and scenarios to meet specific training needs.

Aviation Industry:

Innovation: The enhancement should demonstrate innovation and advancements in flight training technology, reflecting positively on the industry.

Software Users Community:

Community Engagement: Encouraging community involvement and feedback to ensure the enhancement meets the needs and expectations of users.

Accessibility Advocates:

Accessibility: Ensuring that the software is accessible to users with disabilities, providing alternative interfaces or features to accommodate diverse needs.

3. Evaluating for the two suggested ways to realize your enhancement how they impact each identified NFR and stakeholder.

VR Integration

End Users:

Usability: VR integration offers a highly immersive and intuitive training experience, enhancing usability.
Effectiveness: Provides a realistic training environment, improving effectiveness in skill development and proficiency.
Reliability: Requires stable performance to maintain immersion and prevent disruptions, impacting reliability.
Performance: Demands higher processing power for VR rendering, potentially affecting performance if hardware requirements aren't met.

Project Team:

Maintainability: VR integration may require more complex maintenance due to the intricate VR systems involved.
Scalability: Increased hardware requirements and complexity may affect scalability.
Security: Introduces potential security concerns related to VR hardware and software.
Interoperability: May require integration with various VR hardware and software, impacting interoperability.

Project Managers:

Manageability: Requires careful resource management and coordination due to the complexity of VR development.
Cost-effectiveness: Higher development costs due to the need for specialized VR technology and expertise.
Risk Management: Higher risk of delays or technical challenges associated with VR development.

Funders/Investors:

ROI: Potential for a high ROI if VR integration significantly improves training effectiveness and user satisfaction.
Marketability: Enhances marketability by offering a cutting-edge and immersive training solution.

Other Stakeholders:

Regulatory Bodies: May need to ensure compliance with regulations related to VR technology in training simulations.
Suppliers/Partners: Requires collaboration with VR hardware and software providers for integration.
Training Institutions: Offers advanced training capabilities but may require investment in VR equipment.

AR Integration

End Users:

Usability: AR integration provides a moderately immersive and intuitive experience, enhancing usability to a lesser extent than VR.
Effectiveness: Enhances training realism but may not match the effectiveness of VR for skill development.

Reliability: Similar reliability concerns as VR, requiring stable performance for immersion.
Performance: Requires less processing power compared to VR, potentially offering better performance on lower-end hardware.

Project Team:

Maintainability: AR integration may be easier to maintain due to simpler hardware and software requirements.
Scalability: Less demanding on hardware, potentially offering better scalability compared to VR.
Security: Similar security concerns as VR but may be less pronounced due to simpler technology.
Interoperability: May require integration with various AR hardware and software, similar to VR.

Project Managers:

Manageability: Potentially easier to manage compared to VR due to lower complexity.
Cost-effectiveness: Lower development costs compared to VR, offering a more cost-effective solution.
Risk Management: Lower risk of delays or technical challenges compared to VR, reducing overall project risk.

Funders/Investors:

ROI: Still offers potential ROI but may be lower compared to VR due to less immersive experience.
Marketability: Enhances marketability by offering an innovative and immersive training solution, albeit to a lesser extent than VR.

Other Stakeholders:

Regulatory Bodies: Similar compliance concerns as VR but may be less stringent due to simpler technology.
Suppliers/Partners: Requires collaboration with AR hardware and software providers for integration, similar to VR.
Training Institutions: Offers enhanced training capabilities at a lower cost compared to VR, potentially more accessible to institutions with budget constraints.

4. Determining which way to realize the enhancement is the overall best one based on the analysis of the previous step.

Based on the analysis of the previous step, the determination of the overall best way to realize the enhancement depends on various factors including stakeholder priorities, resource constraints, and the desired level of immersion and effectiveness for the enhanced training solution.

VR Integration:

- Offers a highly immersive training experience, potentially leading to better training effectiveness and user engagement.
- Requires higher development costs, technical expertise, and resources due to the complexity of VR technology.
- Provides a competitive advantage in the market by offering cutting-edge training solutions.
- May have higher risks associated with project management, technical challenges, and delays.

AR Integration:

- Provides a moderately immersive training experience at a lower cost and with fewer technical barriers compared to VR.
- Offers a more accessible solution for institutions with budget constraints or limited technical expertise.
- May not match the level of immersion and effectiveness provided by VR but still enhances training realism.
- Carries lower risks associated with project management, technical challenges, and resource requirements.

Determining the Overall Best Approach:

Stakeholder Priorities:

- If stakeholders prioritize highly immersive training experiences and are willing to invest in resources, VR integration may be preferred.
- If stakeholders prioritize cost-effectiveness and accessibility, AR integration may be more suitable.

Resource Constraints:

- If the organization has limited budget, technical expertise, or time constraints, AR integration offers a more feasible solution.
- If resources are available to invest in VR technology and expertise, VR integration may provide a higher level of immersion and effectiveness.

Desired Level of Immersion and Effectiveness:

- If the primary goal is to deliver the most realistic and immersive training experience possible, VR integration may be the best choice.
- If a moderately immersive experience is sufficient and cost-effectiveness is a priority, AR integration may be preferred.

Risk Assessment:

- Consider the potential risks associated with each approach, including technical challenges, project management issues, and delays.
- Mitigate risks through careful planning, testing, and contingency measures.

While VR integration offers a higher level of immersion and effectiveness, it comes with higher costs and technical challenges. AR integration provides a more accessible and cost-effective solution but may offer a slightly less immersive experience. If the organization has the necessary resources and stakeholders prioritize highly immersive training experiences, I would recommend VR integration as the overall best approach. While it involves higher costs and technical complexity, VR offers unparalleled immersion and effectiveness in training simulations. This approach aligns with stakeholders' goals of providing the most realistic and engaging training experiences possible, potentially leading to better user engagement, learning outcomes, and competitive advantage in the market.

Current State

Virtual Reality support as it stands in the current release of FlightGear is far from finished. As of March 31st, 2024, the current state of VR support is “highly experimental”. The systems that do exist are built off of the “OpenXR” API using a library called “oxgXR”. There are some objectives that are near or fully completed, but the majority of goals are not currently met. Those that are near completion are: “splash screen”, controller support, hand tracking, and interaction with animated controls. Many development goals with regards to general features and user-based interaction with VR are missing or incomplete [1].

The revised conceptual architecture is seen in Figure 1 below. The OpenXR API is the library that would provide the basis for VR capabilities in the Flightgear system. To compare this new diagram, it was concluded in the previous report that FlightGear uses a repository-style architecture. As per the legend, the new dependencies are depicted with an orange arrow. The additions to the conceptual architecture are centered around the addition of the OpenXR API. This library will have a two-way dependency with the Main Build to communicate information regarding VR simulation and rendering, and a one-way dependency to the I/O Interface to display the rendered VR scenes to the user.

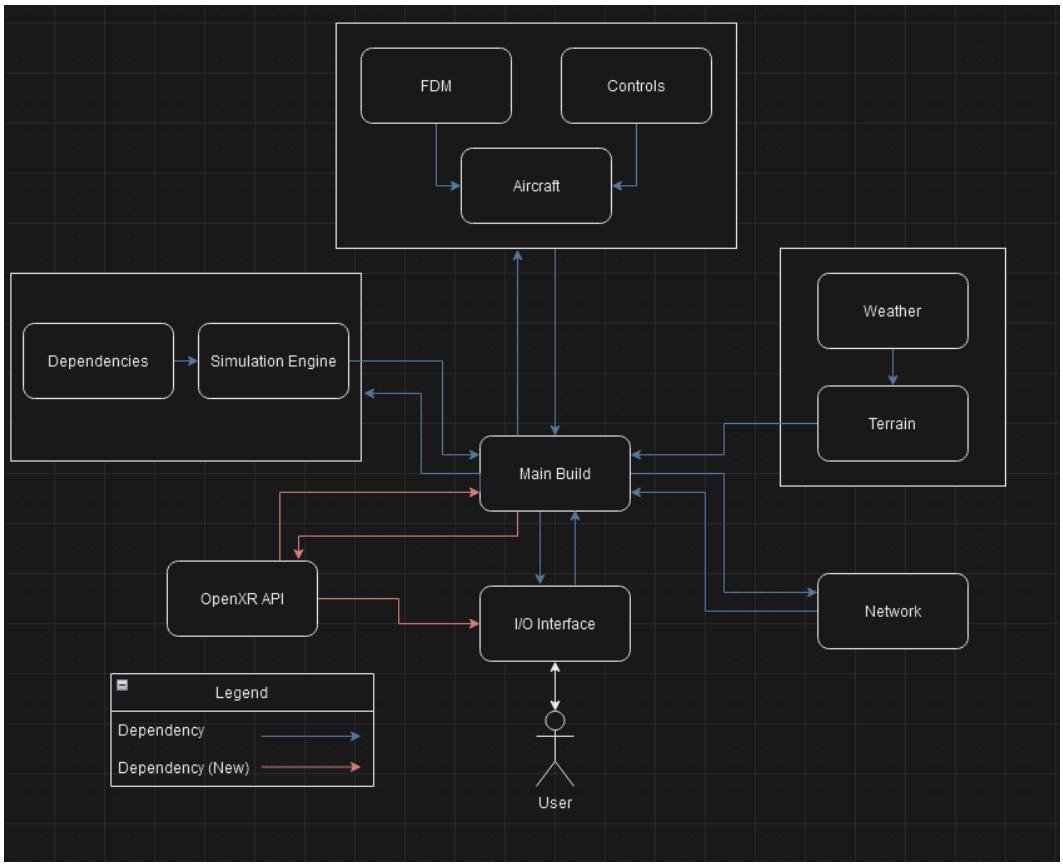

Figure 1: Revised Conceptual Architecture with VR enhancement.

As for AR, there is currently no framework or implementation in place in the current build of Flightgear. While seemingly straightforward to implement, this is likely absent because the

available AR technology is not as developed or popular [2]. Thus, it would not be worthwhile to put development resources towards it.

Effects of Enhancement

Conceptual Architecture

With a repository-styled architecture, it is simple to add modules to the shared data to the Main Build subsystem. To integrate VR capabilities to the rest of the system, no additions need to be made to the simulation of the world itself. This means no modifications to the Aircraft, Environment, or Network subsystems need to be made. The reason for this is that once the world is simulated internally, it is the I/O Interface subsystem in conjunction with the Main Build that renders the simulation to the desired medium for the user.

Evolvability

Virtual Reality technology is being researched and developed. The ability for FlightGear to evolve its VR implementation is limited by the development of the VR frameworks and APIs that are required. As the technology becomes stronger and as the existing VR support software grows, FlightGear can adopt these to evolve its VR support.

Performance

As mentioned previously, it is recommended to use a high-performance graphics card when rendering scenes in VR. With optimizations, some performance gains can be made. However, VR is very costly in terms of computing-power requirements, and thus the performance of the system would be heavily impacted when rendering scenes in VR.

Testability & Maintainability

For testing new features developed in Flightgear, this new enhancement will increase the work needed to develop the systems and interactions for VR. For each additional feature, the mechanisms and interactions occurring in the background are already being handled by the software, therefore only the VR rendering and interactions need to be implemented when introducing a new feature to the system.

File & Structure Modifications

To add VR rendering, many source file changes would be within the src/Viewer folder, which is encompassed by the I/O Interface subsystem. In particular, there are files to do with the Camera management which would be responsible for determining what the player can see, and a VR managing file named “VRManager.cxx”. These files and related paths are where the majority of additions would be placed to complete VR support [3].

Use Case Diagrams

The first use case outlined will be the case where a user attempts to go on a singleplayer flight using a VR headset as their interface. In this situation, the sequence of events is generally symmetrical to how the system would previously handle a singleplayer flight, and as such the diagram and this section will increase importance and attention on the proposed additions of VR handling to highlight the processes.

First the user wearing the VR headset will be in constant communication with the I/O interface and through it the main build as they will be using the controllers and sensors associated with the hardware to communicate their inputs. The User will select the settings for the flight from the interface elements and use the VR controllers to select what they desire. The main build initializes all the needed subsystems which will allow the user to select the aircraft and scenery through the I/O Interface. Main will proceed to get data from the Terrain and Aircraft subsystems to send to the OpenXR API subsystem that will render the initial VR scene of the chosen setting to send to the I/O Interface for the User. The I/O Interface will continue to receive inputs from the User from their VR controllers and motion sensors of the headset for when they attempt to look around the cockpit and view their surroundings. These inputs will be sent to the OpenXR API through Main to continue updating the VR scene based on the head tracking data and controller inputs. As the flight continues Main will continue interacting with the Simulation Engine to coordinate between the Terrain, Aircraft, and FDM subsystems to continue gathering the requisite simulation data for the OpenXR API to render the VR scene accurately. Then the updated scene will be sent through the I/O interface to the User for display. This cycle will repeat throughout the flight until stopped by the user.

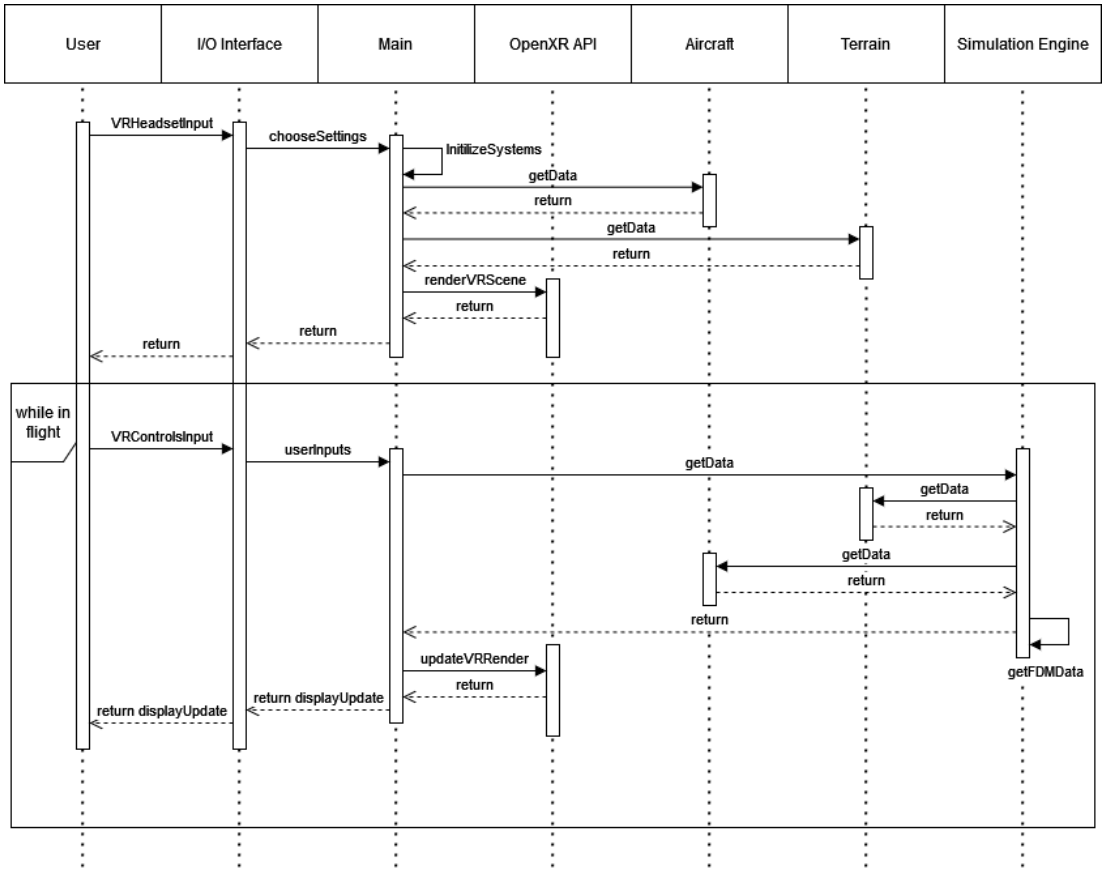

Figure 2: Sequence Diagram for singleplayer flight simulation use case.

The second use case outlined will be the case where a user wishes to practice an emergency landing. Once again the virtual reality implementations will be highlighted to put focus on them due to the nature of the enhancement having limited effect on the sequences. Like the previous use case, the user wearing the headset will start the simulation and Main will initialize all necessary subsystems. The User will select the same settings through the VR interface but in this case will include the option for a scripted simulated aircraft failure. As such the steps of initial flight takeoff mimic those of the previous case. The Main will inform the Simulation Engine of the chosen emergency scenario and then communicate that to the FDM .The FDM receives the notification and will alter the requisite data and return it to the Simulation Engine to simulate the failure accurately. The I/O will handle the appropriate audio and visual cues and send them to the user, using the VR Handling component. The User will then deal with the emergency situation based on the audio and visuals transmitted to them through the VR headset and act accordingly.

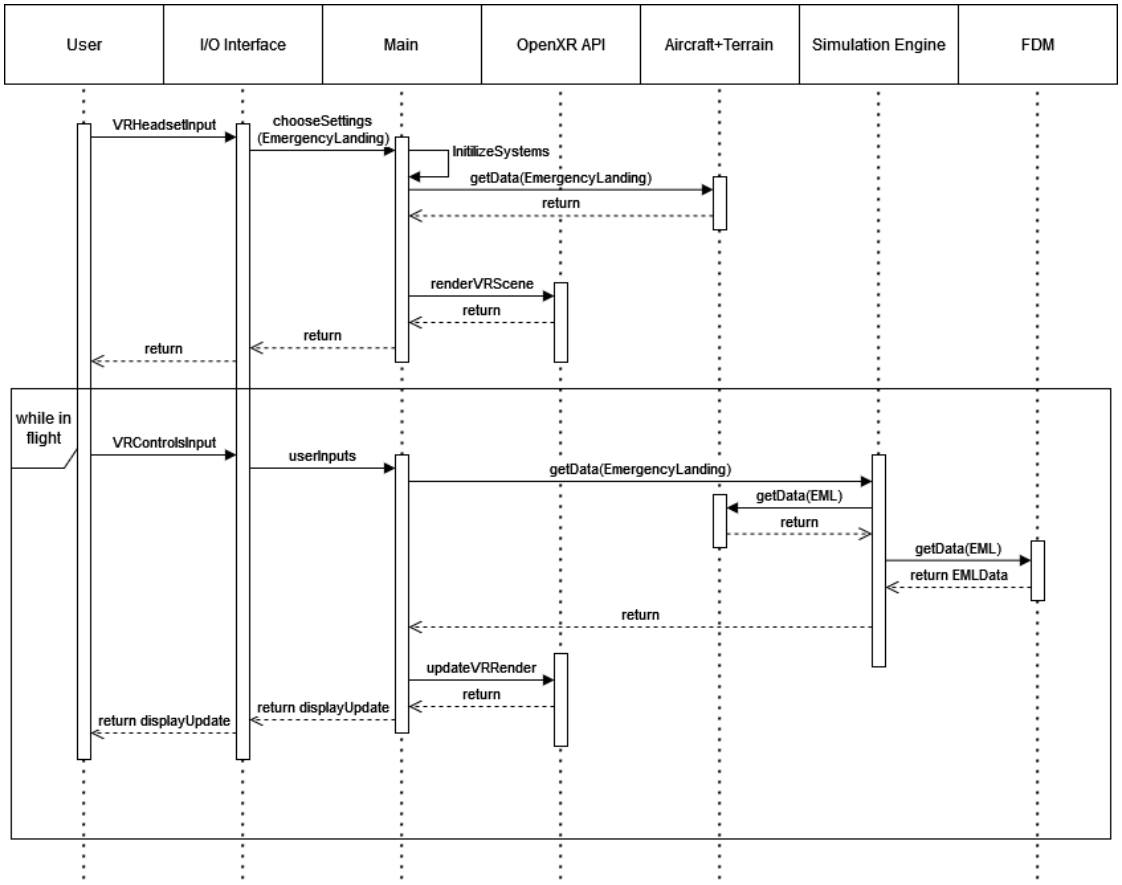

Figure 3: Sequence diagram for emergency landing practice use case.

Potential Risks

When it comes to potential risks, we've identified several key areas that include physical risks, maintainability, scalability and performance. These new factors will need to be tested and monitored to ensure the success of the product.

Physical Risks The shift to VR/AR introduces a wide variety of physical risks that both users and developers must be aware of in order for safe and successful operation of the program. Physical risks like motion sickness, nausea, eye strain and more are standard hazards when using augmented or virtual reality applications for a prolonged time. Due to the increased risk in this form of operation, users should be made aware of these possibilities before stepping into the simulation and efforts should be made to give warnings when using the application for too long. Other indirect physical risks could also include impact injuries due to a lack of space or electrical hazards in relation to setting up the proper equipment for VR/AR. These risks do have to be taken into account however should not be our main focus as it is more hardware dependent. To ensure safe usage, warnings could be included to ensure a proper environment is maintained but ultimately these risks are not the primary concern.

Maintainability Adding VR/AR support also means relying on 3rd party hardware to operate our software thus introducing additional problems in the event of hardware malfunctions

or updates that affect the core functionality of the hardware. Platform support must also be improved upon to account for the various types of hardware that support the new features. While hardware updates and malfunctions are generally out of our control, further efforts need to be made to mitigate the risks. This could involve additional compatibility testing to ensure issues are accounted for before moving to production or improving communication with hardware providers to ensure the development teams are aware of any potentially impactful changes.

Scalability The introduction of VR/AR capabilities is a very big change and due to the nature of the update, it can be likened to creating a whole new application in general. With this, any updates to the core application must also ensure compatibility with the new VR/AR feature, making the development time for new features longer. This affects the scalability of the application as developers cannot roll out new updates as quickly as before.

Performance VR/AR naturally require more power to run smoothly thus increasing the performance cost of the application. Users will most likely require a stronger setup in order to accommodate the new features. The backend must also accommodate for the increase of data received which could potentially slow down the entire system. Furthermore, previously functional subsystem connections would have to be reevaluated in order to determine if they are still necessary or even capable of handling the new workload.

Plans for Testing

In order to provide a conclusive test for the new features, the new test plan would need to account for both software and hardware problems. Since the new feature would require a somewhat large overhaul to current subsystems, one of the main focuses of the new testing plan should be ensuring that each subsystem is capable of handling the new data and usage. This would also include testing the interactions between existing subsystems with any new subsystems (OpenXR API) to ensure data is being transferred correctly and efficiently. Specifically, this would include the interactions between the new OpenXR API and the communication between the IO Interface as well as any subsequent information that would travel to another subsystem. This would involve smaller system tests to ensure data sharing between two components are functional while system tests would need to be introduced to test overall system communication. While these tests cover a wide variety of problems and ideally, the simple data transfer problems, it is unrealistic to assume a feature of this nature can be fully covered. By focusing on providing test coverage for the basic functionality of each module, it ensures that when an obscure problem does arise, the solution can be quickly isolated without worrying about a core functionality breakdown. In the case that a core function does stop working, these tests will also help quickly identify the problem.

As previously mentioned, the introduction of third party hardware creates other issues within our platform. Since our application directly relies on outside hardware that is also frequently updated, we need to include compatibility checks to ensure core functionality can be achieved regardless of how the hardware operates. While important, these tests do not need to be as frequent as hardware issues will most likely only occur when hardware developers release their own updates and nightly checks for example, could reduce the efficiency of our application. In order to decrease stress to our own platform, these checks should be scheduled

for longer intervals or whenever a large update has been released for any of the compatible VR/AR devices.

Conclusion

The objective of the enhancement of the FlightGear software was to improve the immersion and training effectiveness of the software.

There were two methods considered to achieve this: Virtual Reality, and Augmented Reality. Overall, Virtual Reality was analyzed to be the better option for higher immersion and more research & development being put into the technology.

Dependencies would be added to the conceptual architecture. Namely, a subsystem managing VR implementation which utilizes the OpenXR API, integrating with the Main Build and I/O Interface subsystems. The evolvability of the enhancement is directly limited by the development of external VR technologies and methods. Performance demand would increase, which is expected because of the rendering demands of VR. This could limit the VR performance, as the third-party libraries may be developed slower than the software itself. The performance could be optimized if required, however, the performance will be impacted regardless. Another minor, more physical risk to be considered with VR, is the possibility for eye strain or other environmental hazards such as the space requirement.

The primary testing needed for this new enhancement would be to ensure that the VR integration is able to properly handle interactions with existing subsystems. As well, when major updates with third-party libraries and software are released, it must be ensured that any new features do not interfere with any existing mechanisms.

Lessons Learned and Team Issues

As mentioned in the previous paper, the state of the group missing a member limited progress. Despite this the group divided the tasks effectively to make up for the absence and successfully completed the objectives. Time management was particularly important in this instance as many members were busy with other responsibilities so making progress early on was vital to the completion of the report. Another difficulty encountered by the members was the expiration of the trial version of the Understand software they had previously used for analysis. This caused some issues for properly analyzing how the addition of a new subsystem would affect the overall software but was handled well in the end through previous reports and logical deductions.

Through researching this enhancement the group was able to learn about the intricacies of VR and AR development, which broadened their knowledge on their respective benefits and drawbacks. This paper also enabled them to learn about the stakeholders and their values for the FlightGear software, and resulted in a positive learning experience on how the business side of real world projects are handled.

References

[1] J. Hogan, "Virtual Reality," 31 March 2024. [Online]. Available: https://wiki.flightgear.org/Virtual_Reality. [Accessed 13 April 2024].

[2] GCFGlobal, "Understanding Virtual Reality and Augmented Reality," [Online]. Available: https://edu.gcfglobal.org/en/thenow/understanding-virtual-reality-and-augmented-reality/1/. [Accessed 13 April 2024].

[3] J. Turner, "FlightGear - Open source flight sim," 30 September 2021. [Online]. Available: https://github.com/FlightGear/flightgear. [Accessed 13 April 2024].