CMPE322/326 Assignment 3 - Feature Proposal
FlightGear

Submitted By: Group 20
Jasper Lim, Laurie Yuzichuk, Jordan Herzstein, Campbell Love, Christopher Seguin Bianchi, Cameron Bennett

Abstract

This report aims to outline a potential feature proposal to FlightGear by the group, by analyzing existing features and community discussions to add an innovation to the system where the need exists. From the group's research and feedback, a feature that would intersect ATC with a speech to text (STT) module and LLM functionality would modernize the current ATC subsystem within FlightGear. Specifically, the group has decided to add STT to the existing Red Griffin ATC open source add-on to add a new option for communicating with the ATC as well as LLM functionality for both local and external API LLMs to create flexibility and further immersion with FlightGear's ATC. This system was decided on after researching other existing add-ons for the ATC system, as well as reading forums of users who want to enact STT functionality within some of these existing systems.

Through SAAM analysis, this report identifies key stakeholders in the project, the necessary architectural style for the feature proposal, and key attributes that need to be considered. In terms of stakeholders, developers will have to consider through the inclusion of LLMs all the complexities that come with it, including model training, inference, and optimization requiring efficient algorithms. Users require that the system be fast, responsive, and accurate, but may be confronted with the resource consumption of a LLM being too much of a strain on performance, to the point of being debilitating for their hardware. Conclusively, a Client-Server architecture model should be adopted, as it is the most logical layout in terms of arranging workload for the components of an LLM. Additionally, Client-Server architecture also closely mirrors the established conceptual architecture of FlightGear.

In terms of impact on subsystems and FlightGear's architecture, the ATC and Input modules will be affected the most. The existing ATC subsystem must be able to distinguish good and bad STT input and generate an appropriate response. For Input, the subsystem must become able to process microphone input. In terms of the conceptual and concrete architecture, FlightGear's Client-Server network must be able to accomodate Numen STT data, and may have to be able to interface with an external API for LLM processing. The ATC module would be modified by the presence of our enhanced Red Griffin add-on.

The largest drawbacks involve the drain on performance depending on whether the LLM is local, or whether an external API is used. The latter option brings security and data privacy concerns to the user. In order to optimize this and ensure functionality, rigorous testing needs to be conducted on STT performance with the Red Griffin ATC interface, the text processing capability of the LLM, and the integration of both aspects with each other.

Introduction

FlightGear is a free, open-source flight simulator in development since 1997, supported on multiple OS's including Windows, MacOS, Linux, IRIX, and FreeBSD [1]. FlightGear has been used in academic research, education, training, and for recreational purposes [1]. Previously, we covered the conceptual and concrete architecture of FlightGear. It was determined through the mapping of the project, specifically dependencies such as the FDM sub-module using Sci-tools Understand that there were some divergences between the conceptual and concrete architecture. Thus, the conceptual architecture was adjusted to reflect the gaps with our concrete architecture in the second report.

In this report we will introduce a new potential feature to the project. In particular, the team was interested in an intersection between Air Traffic Control (ATC), voice speech to text recognition, and Large Language Model (LLM) functionality. The latter two features are not available in any FlightGear addon, and represent a key element of the flight simulation experience that is currently lacking. With takeoff and landing from various real models of airports being a prominent feature of FlightGear, it is reasonable to enhance the immersion and real-training applications by enabling the player to simulate real verbal communication with existing ATC systems. Existing FlightGear specific add-ons, community discussion, and other software projects were researched to include in our proposal.

Overview of New Feature

The enhancement of FlightGear will be an add-on for the ATC subsystem that will allow voice commands for ATC using language learning models that can run locally on the player's computer. Currently FlightGear has a few add-ons that have voice and speech synthesis when communicating with ATC such as ATC-pie, Spoken ATC, and Red Griffin that exist without the use of LLMs [2] [3] [4]. LLMs could improve the immersion of the simulator while also providing flexibility for players and developers to train and use their own models. This can also be beneficial for non-native English-speaking players to run local models that employ their native tongue, increasing the accessibility of FlightGear for an international audience.

To introduce this system into FlightGear's existing architecture we plan on using some existing add-ons as dependencies as they already have voice features for ATC such as the aforementioned ATC-pie, Spoken ATC, and Red Griffin. We have decided to develop our proposal based on the Red Griffin ATC specifically. Red Griffin already uses the keyboard to communicate directly with the ATC, though it does not have functionality to deliver speech-to-text capabilities to communicate with ATC [5]. Since 2023, members of the community have been discussing adding such a feature to an existing ATC add-on, in which some have suggesting allowing a project such as Numen to operate with an existing add-on [5] [6]. Numen is a voice control for hands-free computing, which is ideal for a STT module which can be integrated with existing FlightGear add-ons [6]. We will add speech-to-text functionality to Red Griffin using the existing Numen project, and then optionally allow LLM functionality to integrate into the ATC for flexibility with key words when communicating with ATC.

SAAM Analysis

The following includes a Software Architecture Analysis Method (SAAM) analysis for the proposed feature, including identifying stakeholders, chosend architecture, and key attributes.

Stakeholders

The stakeholders involved in integrating voice commands for ATC using LLMs into FlightGear which mainly encompass developers and players. Developers are tasked with ensuring software compatibility, security, and reliability while integrating the voice command feature. Users, including pilots and enthusiasts, prioritize usability, security, and performance in their interaction with the simulator.

Developers

Developers play a crucial role in integrating voice commands using LLMs into FlightGear. They face the challenge of ensuring software compatibility with existing systems and libraries, especially considering the computational resources required for real-time voice recognition and processing. The use of LLMs introduces complexities in model training, inference, and optimization, requiring developers to implement efficient algorithms and data structures to handle large-scale language models effectively. Additionally, developers must address security concerns related to voice data privacy when using external API's.

Users

Users, particularly pilots and enthusiasts, are focused on usability, security, and performance when interacting with FlightGear's voice command feature. The computational challenge here lies in providing a seamless and responsive user experience, which necessitates optimizing the speed of LLMs and improving the accuracy of voice recognition in the STT module. Users expect the system to accurately interpret and execute voice commands in real time, requiring efficient algorithms for voice processing and command execution without causing noticeable delays or mistakes in interpretation. Users also should be concerned with the resource usage that would be consumed by the LLM, which could represent a significant drop in performance, or be untenable to run at all by their machine's hardware if the LLM is local.

Architecture Choice

The chosen architecture for the enhancement would be a Client-Server architectural style. This chosen architecture would make sense as the RGATC add-on would act as our client with keyboard/STT input functionality, whereas the "server" architecture could be the LLM it communicates with. This is clear in the case in which an external service with an API is used, however, even if the LLM is run locally it can act as a server which services the RGATC add-on.

Key Attributes

Key attributes to consider during the integration process include maintainability, testability, performance, and scalability. Maintaining the voice command system's codebase should be prioritized for ease of updates and modifications without disrupting overall functionality. Rigorous unit testing methods should be employed to verify the reliability and efficiency of voice commands. Additionally, performance profiling tools can be utilized to identify and address any bottlenecks that may affect system performance. Scalability testing is crucial to assess the system's ability to handle varying loads of voice commands effectively when working in multiplayer environments.

By addressing these attributes and employing appropriate methods such as comprehensive code documentation, unit testing, performance profiling, and scalability testing, FlightGear can successfully integrate voice commands using LLMs into its architecture while ensuring a well-rounded approach to system development and enhancement.

Impact on Subsystems

The main subsystems that will be affected are those that relate to ATC interactions in FlightGear, such as ATC and add-ons. However, some subsystems such as Input will be lightly adapted in order to properly integrate the new feature.

ATC

Depending on the input received from the user that is then converted from speech into text commands, the ATC subsystem will need to be able to properly understand the input and must output a response accordingly back to the user based on said input.

Additionally, when the speech to text converted response is either not coherent or an irrelevant command, the ATC must be able to recognize this and thus will report back to the user that their message has not been understood and then prompt them to repeat their message more clearly. For instance, when given a reply that the program does not understand, the ATC could reply back with “Transmission is weak and distorted, over” or “Say again, over”. This will further aid in simulating real life radio communication and adds to the immersion of the user.

Add-ons & Red Griffin ATC

Since the enhancement would build upon the preexisting add-on, Red Griffin ATC, it (and thus the Add-on subsystem) must be able to access the user speech input in order to convert it to text. From there, Red Griffin ATC will map the speech-to-text to established requests that can be given to the ATC, such as requesting an engine start or to have the user’s aircraft taxied on or off the runway.

Input

The Input subsystem will need to be able to register speech from the user, which will be relayed to and used by the added speech to text component. Since currently FlightGear’s Input subsystem only uses keyboard, mouse, and buttons as a source of input, a microphone input component within the subsystem must be added in order to do so.

Impact on Conceptual & Concrete Architecture

This section will analyze the impact the proposed feature will have on both the conceptual and concrete architecture of FlightGear.

Impact on Conceptual Architecture

As described in the team's first report, the conceptual architecture is a combination of three architectural styles: High-Level Architecture, Model-View Controller, and Client-Server architecture. In the conceptual architecture, the client interfaces with the network through UDP. This connects to the FDM server, and through a network manager interfaces with flight dynamic calculations, ATC simulation, AI object control, scenery update, audio, and rendering.

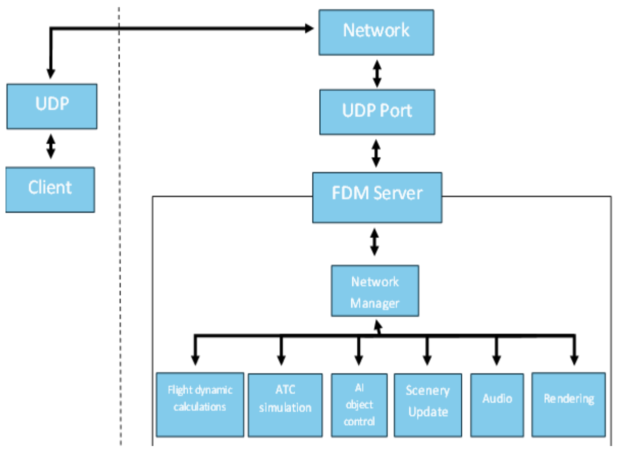

Figure 1: Conceptual Architecture of FlightGear from a previous report.

The proposed addition to the Red Griffin ATC of adding speech-to-text using Numen and incorporating LLM functionality will impact the conceptual architecture in various ways. One such way this will be impacted is through the integration with the network manager and UDP communications. All network interactions will need to be updated to handle speech-to-text features. This can be accomplished either through integrating STT functionality into the existing network flows, or expanding the UDP-based communication protocols to support the additional data that STT brings. If the feature uses an external API for LLM processing, this would also incur much more network traffic. The Client-Server interactions would similarly need to be updated to accommodate the added data flow from speech-to-text and LLM functionalities. Finally, the ATC simulation would use the Red Griffin ATC add-on instead of the base ATC, along with the proposed changes of STT and LLM integration.

Impact on Concrete Architecture

The conceptual architecture of the FlightGear system was outlined in the team's second report. As discussed in the previous section, it involves various subsystems including FDM, Viewer and GUI, Aircraft, Autopilot, Environment and Scenery, Input and Systems, Network, Sound, and Add-ons and Scripting.

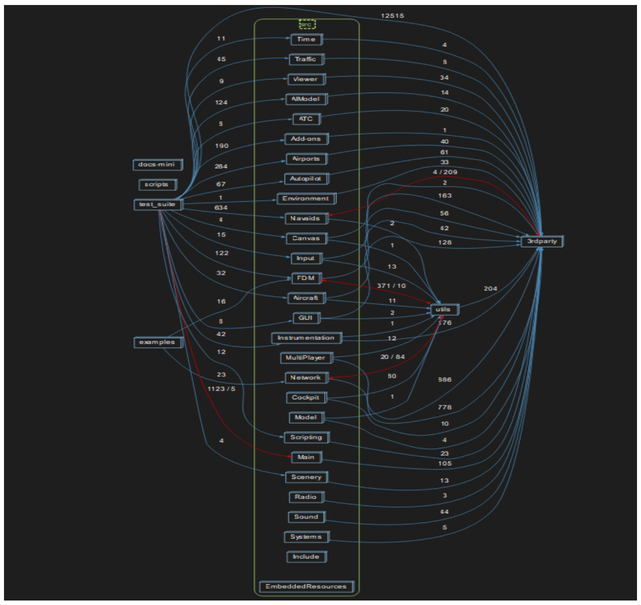

Figure 2: Concrete Architecture of FlightGear from a previous report.

The proposed additions would incur various modifications to the existing elements of the architecture. One area that would need to be modified is the ATC module. Since the Add-ons subsystem will feature the new ATC, Red Griffin ATC, the main ATC subsystem will not need to interface with the other components. The dataflow between all the subsystems will also need to be updated to include additional dependencies such as the Numen STT and potentially external LLMs.

New subsystems would also need to be incorporated into the architecture. One such subsystem is the speech recognition engine that interfaces with Numen. This would be the component that processes the STT, taking audio from Input and Systems and processing it into text. Another subsystem that would be added is for language processing. This would integrate the LLM functionalities that have been described in the proposed additions. A data interface would also be needed to manage the flow of data between the two mentioned subsystems, enabling synchronization and reliable data transmission.

Potential Risks and Limitations

The integration of an LLM powered ATC system presents many new and interesting opportunities in theory, however, poses many risks and limitations. The most obvious risk is that LLM's are computationally expensive and will consume a lot of processing power. The player using this add-on thus has two options; rely on an external service with an API or run their own model locally with their own GPU resources. While an external API would allow the user to use none of their own GPU processing, it would be slower than running the model locally. Using the example of ChatGPT, a player using the ChatGPT API would have to communicate with the ChatGPT servers first before generating a response and sending it back. Running the model locally would not have to rely on server response or availability, however, would consume a lot of VRAM on the user's GPU depending on the model. Thus, this option would not be viable for users without a powerful GPU on their own PC or would have to sacrifice accuracy (using smaller LLMs) for resources.

Another aspect to consider in terms of risk is data privacy. A lot of users do not trust companies such as OpenAI and Google to hold their information and use it correctly, so using their external LLM services can raise privacy concerns for them. Although the actual audio recording isn't being sent any external server, the output text is still being sent to these servers and can be used to improve their LLM. Users don't know where or how their information is being used, hence the concern. This claim is backed up by International Governments such as the Italian Digital Protection Authority (IDPA). The IDPA has raised its concerns about Open AI's data processing, specifically about if it complies with the bloc's General Data Protection Regulation (GDPR). These concerns focused on the legal basis of collection and processing of personal data of training algorithms in the LLM, and the times the AI tool produces inaccurate information about its users [7]. A solution to this could be the use of a local LLM such as Mistral, which would limit these concerns and keep all the information stored on the computer, removing some tensions on the data privacy concerns. Mistral is Europe's biggest contender for the AI global race, and "subscribes itself to the idea that AI software should be open source" [8]. The idea of an open source LLM matching FlightGear's open-source nature will promote safe updates and reuse of the product.

Another concern is while our proposal improves flexibility and ease of use of ATC communication in the simulator, some accuracy might be sacrificed in using a speech to text models with an LLM. LLM's rely on a vast amount of training data to operate with precision. While a system in which key words are identified to activate actions from the ATC is less flexible, it is a lot more predictable than an LLM which might interpret things incorrectly. Additionally, any speech to text program might hear certain words incorrectly due to mic issues or not understanding certain accents. While these programs tend to become more precise and robust over time, a keyboard is always going to act more predictably than a person's voice.

Testing

If this feature were to be implemented, there would be many different procedures that could be conducted on the LLM component designed to integrate within Red Griffin's Air Traffic Control add-on. It will specifically focus on testing speech-to-text functionality with the RGATC interface, testing of the LLM with text input only, and the integration of these two components.

For Testing the STT component with the RGATC interface, there would be testing involving Speech Recognition accuracy and Interface Compatibility. Testing if the speech is being accurately transcribed into pilot voice commands in text is the primary problem. Various voice inputs representing typical ATC commands would be provided to gauge the system's recognition accuracy under different conditions, such as background noises, accents, and varying speech patterns. With the LLM, a text input validation would first need to be done. The test would focus on the LLMs responsiveness and accuracy when provided with text inputs directly, using different ATC commands and viewing the results the LLM spits out to deem if it is accurate or not.

Once text-only input validation is up and running, interface compatibility between the LLM and RGATC would be next. This would include testing from the recognized speech to text output to be correctly interpreted and acted upon by the RGATC add-on, resulting in appropriate ATC responses and interactions within the FlightGear environment.

Sequence Diagrams

Included in this section are two different use cases of the proposed feature including use of the feature with and without an external LLM API.

Use Case #1

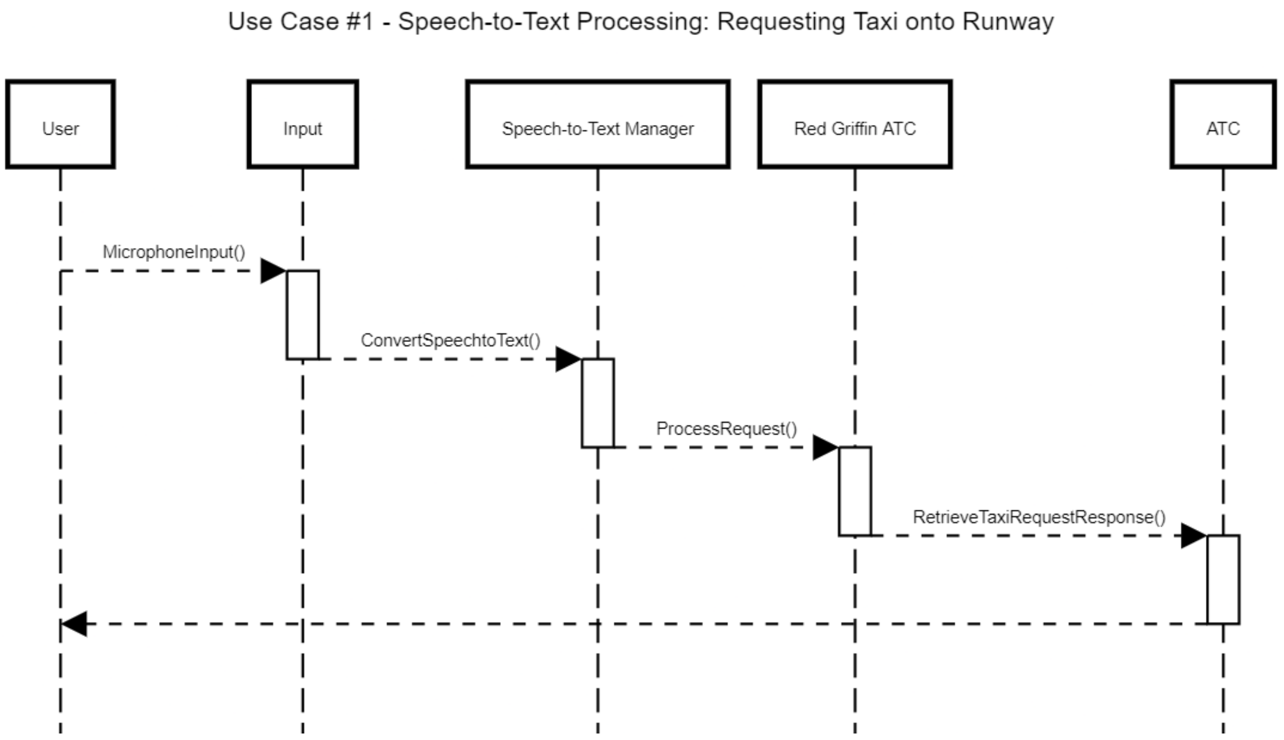

Figure 3: Sequence diagram of using Speech-to-Text in order to communicate with the ATC to request a taxi onto runway.

illustrates a use case for the proposed enhancement, where the user speaks into their computer's microphone to communicate with the ATC and requests that their aircraft be taxied onto the runway. The Input component registers the spoken input, which is then passed onto the Speech-to-Text Manager and converted into text. From that generated text, the Red Griffin ATC will map the text to a specific request that's already been implemented into Red Griffin ATC. That request is given to the ATC and will output the appropriate response to the request back to the user.

Use Case #2

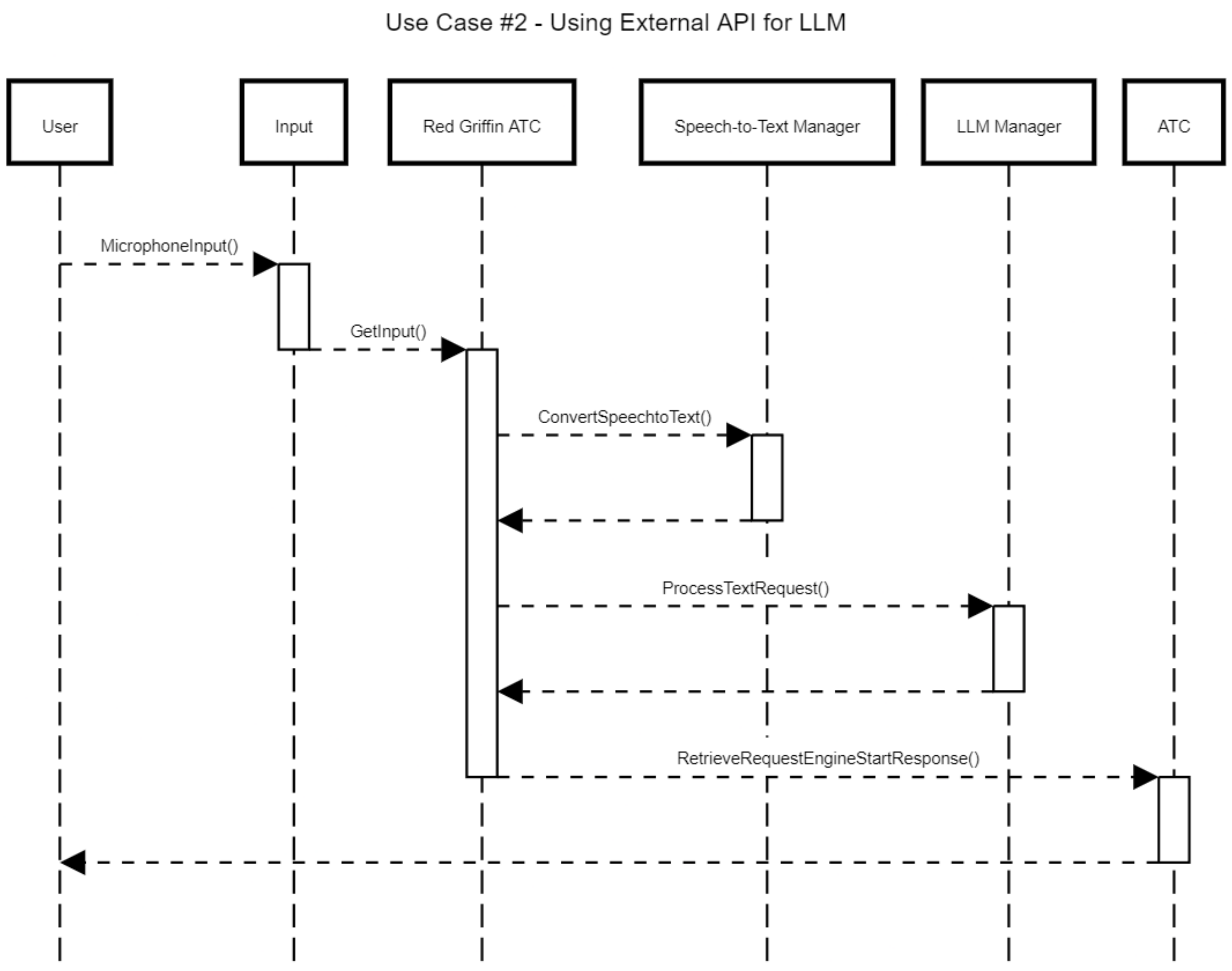

Figure 4: Sequence diagram of using LLM with Red Griffin ATC in order to process the request of starting engine.

Figure 4 above depicts use case #2, where an external API is used for an LLM and is integrated into Red Griffin ATC. Here, Red Griffin ATC would be calling the required components needed when processing the user's spoken input. Like in the previous use case, the registered input is converted into text. However, in this instance, the LLM is used in order to assess the converted text and more accurately understand the user's request regardless of minor discrepancies or synonyms used, which allows the system to be less rigid in what is considered an accepted viable spoken input.

Lessons Learned

When creating our proposal, we first considered creating an STT module with the ATC where we didn't believe it existed in FlightGear proper. With the suggestion of the group's TA and further research into the FlightGear wiki we learned that several add-ons with some voice functionality were already developed by the community. In the consideration of further innovation, we decided to expand our proposal from integrating a STT module into the ATC to also including LLM functionality.

The group's TA made sure to point out the many potential downsides a system that includes both STT and an LLM together might create in terms of performance and accuracy. To combat these issues modularity was a huge consideration in the creation of the proposal so that both players and developers in the FlightGear community can decide what aspects of FlightGear are most important for their use case: performance, flexibility, accuracy and reliability, immersion, and data privacy. For example, if a user is only concerned with performance and accuracy, they can keep their system as is without using the design of our proposal at all. For the subset of players that want to push the immersion of the simulation while being easy to use and without sending data to an external server, the feature proposed can also work for them. The feature can be configured to run locally on the system while using STT to run commands and an LLM to interpret the text flexibly to depict ATC communications more realistically, increasing immersion. For players than want to push immersion and ease of use but do not have the necessary local processing power, external APIs exist and can be used to implement that functionality for them as well.

Conclusion

Overall, we found that by extending the current FlightGear architecture through the add-ons subsystem, we were able to add more immersive features to the ATC component of FlightGear. In order to achieve this, we could rely on existing open source add-ons made by the community, in this case Red Griffin ATC, to improve the system rather than creating everything from scratch or add unnecessary complexity or instability to the core components of FlightGear. The options and flexibility of the extension we are proposing allows the player to decide how much they wish to use their ATC system with more immersion and ease-of-use, or keep a reliable and low-resource intensive simulation.

Key stakeholders were recognized in the SAAM analysis, including developers and players of FlightGear. We also decided on using a client-server architecture in which the RGATC add-on acts as the client and any locally running LLM or external AI service such as

ChatGPT via their API can act as a server which services the RGATC add-on. Analyzing the impact on conceptual architecture, we hypothesize the effect it would have on the networking subsystem with the addition of STT as well as an external LLM incurring even more network traffic. In terms of the concrete architecture, we include the impact it would have on the existing ATC subsystem, as well as an additional STT subsystem that might be added to the existing system. We also analyzed the potential risk factors that an add-on such as this would have to consider, in particular performance and availability, privacy, and accuracy. In terms of testing, we have considered employing the STT and LLM functionality into RGATC separately before integration. Finally, we've shown two cases in which the add-on can be used, including local usage and usage with an external LLM API.

Data Dictionary

ATC: Air Traffic Control

FDM: Flight Dynamics Mode

LLM: Language Learning Model

API: Application Programming Interface

Red Griffin ATC (RGATC): An open-source add-on for FlightGear's ATC, what the proposal is based on

ATC-Pie: An open-source add-on for FlightGear's ATC

Spoken-ATC: An open-source add-on for FlightGear's ATC

STT: Speech-to-text

References

[1] "FlightGear About," FlightGear, [Online]. Available: https://www.flightgear.org/about/. [Accessed 19 02 2024].

[2] "ATC-pie," [Online]. Available: https://wiki.flightgear.org/ATC-pie.

[3] "Spoken ATC," [Online]. Available: https://wiki.flightgear.org/Spoken_ATC. [Accessed 12 April 2024].

[4] "Red Griffin ATC," [Online]. Available: https://wiki.flightgear.org/Red_Griffin_ATC. [Accessed 12 April 2024].

[5] "Spoken ATC: Speech-to-text ?does it exist," [Online]. Available: https://forum.flightgear.org/viewtopic.php?t=41522&p=412865. [Accessed 12 April 2024].

[6] J. Gebbie, "Numen Voice Control," [Online]. Available: https://numenvoice.org/. [Accessed 12 April 2024].

[7] N. Lomas, "ChatGPT is violating Europe's privacy laws, Italian DPA tells OpenAI," 29 January 2024. [Online]. Available: https://techcrunch.com/2024/01/29/chatgpt-italy-gdpr-notification/#:~:text=AI%20model%20training%20lawfulness%20in,temporarily%20suspended%20in%20the%20market.. [Accessed 12 April 2024].

[8] L. A. a. A. Satariano, "Europe's A.I. 'Champion' sets sights on tech giants in the US," The New York Times, 12 04 2024. [Online]. Available: https://www.nytimes.com/2024/04/12/business/artificial-intelligence-mistral-france-europe.html#:~:text=Mistral%20subscribes%20to%20the%20view,to%20copy%2C%20tweak%20or%20repurpose. .