

Implementation of an Air Traffic Control Simulation Environment for Human Multitasking Research

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Business Informatics

eingereicht von

Ing. Matthäus Spindelböck, BSc

Matrikelnummer 00731224

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Univ.-Prof. DI Dr. Philipp Wintersberger

Wien, 18. März 2026

Matthäus Spindelböck

Philipp Wintersberger



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.



Implementation of an Air Traffic Control Simulation Environment for Human Multitasking Research

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Ing. Matthäus Spindelböck, BSc

Registration Number 00731224

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.-Prof. DI Dr. Philipp Wintersberger

Vienna, March 18, 2026

Matthäus Spindelböck

Philipp Wintersberger



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Ing. Matthäus Spindelböck, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT-Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 18. März 2026

Matthäus Spindelböck



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

Firstly, I want to thank Univ.-Prof. Philipp Wintersberger for supervising this thesis and giving me advice on how to optimize the created software, as well as the written work. Additionally, I also want to thank Dinara Talyapova for helping me in organizing and conducting the user study with the students and experts.

A big acknowledgment goes to the students and experts, who offered their time for free, to take part in the user study.

Last but not least, I'd like to thank my family, friends and Scotty† the Siberian Husky, for their excellent emotional support during the last years of studying and creating this work.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

In today's work environments, the frequent change between different tasks of workers is necessary in many jobs, to fulfill given requirements and execute duties successfully. Methods to increase efficiency in such a multitasking environment result in improving overall productivity of organizations of all different kinds. Ongoing research tries to find ways to make human multitasking more efficient by using computer technology as an aid. [Lin23] for instance, proposes in his work, that software agents can aid in finding the best moment for a task switch, to increase overall performance of the worker. To be able to evaluate research concerning multitasking optimization of real-world jobs, a simulation of a real-world working environment is necessary. This thesis proposes the simulation of Air Traffic Control (ATC) and features the implementation and subsequent evaluation of such a simulation in the scope of multitasking research. To ensure research to be as efficient as possible, this simulation environment should be free for research, lightweight and easy to set up and operate. Lightweight means, the simulation environment should run on a standard consumer notebook. Easy means, that setting up the software can be accomplished in less than 30 minutes and without in-depth knowledge of the ATC domain. Furthermore, non-ATC participants can be trained to operate single tasks of the simulation within 15 minutes. To design an ATC simulation environment, realizing the aforementioned characteristics, this work establishes a list of functional and non-functional requirements. As research methodologies, the *systematic literature review* described by [Kit07], the *Design Science approach* developed by Hevner et al. [HMPR04] and *laboratory experiments* in connection with a *user study* are used. The latter features, among others, an in-depth analysis and comparison of the performance of members of two different groups, accomplishing tasks within the created simulation environment. The first group consists of participants with knowledge of the aviation and ATC domain, the second group of participants without domain knowledge. The thesis concludes with the established findings during the user study and a discussion on how they can be integrated into related work.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Abstract	ix
Contents	xi
1 Methodology	1
1.1 Problem Statement	1
1.2 Research Questions	3
1.3 Expected Results and Success Criteria	3
1.4 Research Methods	3
2 Literature Research	7
2.1 Review Protocol	7
2.2 Functional Requirements for the Simulation Environment	8
2.3 Non-Functional Requirements for the Simulation Environment	14
3 Implementation	17
3.1 Description of the Simulation Environment	17
3.2 Technical Realization	28
3.3 Configuration Options	45
4 User Study and Evaluation	51
4.1 Methods	51
4.2 Results	54
4.3 Interpretations	68
4.4 Conclusion	70
4.5 Future and Related Work	71
4.6 Limitations	72
5 Übersicht verwendeter Hilfsmittel	73
List of Figures	75
List of Tables	75
	xi

Glossary	77
Acronyms	79
Bibliography	81

Methodology

1.1 Problem Statement

In today's work environments, the frequent change between different tasks of workers is necessary in many jobs, to fulfill given requirements and execute duties successfully. Methods to increase efficiency in such a multitasking environment result in improving overall productivity of organizations of all different kinds. Ongoing research tries to find ways to make human multitasking more efficient and reliable. For instance, by improving workplace procedures and design as well as using computer technology as an aid.

Simulation environments for multitasking research have already existed for quite some time, and include standardized psychological tasks such as n-back, Tracking Task, or the VCR Task. However, researchers also require real-world simulations for the sake of immersion, playability, and realism. Therefore, a new and more modern simulation environment with similar cognitive and physical demands is necessary. One domain that is undoubtedly challenging employees world-wide, is the ATC workplace environment, especially in ATC towers or centers where the complexity of procedures and amount of air traffic is high [Hil04]. "Factors such as skill, training, experience, fatigue and other "stressors" all mediate the relationship between task demands and the workload experienced by a controller" as analyzed by Hilburn [Hil04]. He also mentions the two main goals of the ATC systems which are ensuring adequate separation between planes and to organize the air traffic flow in a way, that single flights move expeditiously through airspace. Stein [S⁺85] concludes that "An air traffic controller operates in a complex person-machine system in which he is subject to multiple demands or task loads over time. His workload in response to those task loads will be a function of what he brings with him to the situation (knowledge, abilities, and skills) and what he must do in order to maintain a safe and expeditious traffic flow". He also references that although big improvements in the technical aspects of all components involved in the ATC domain have been made, the "human operator" remains "fallible".

Due to the inherent nature of ATC controller's workplaces, it seems to be an optimal domain to be simulated and used for the purpose of multitasking research. The challenges are to come up with similar skill, knowledge, and rule-based activities in this simulation that correspond to the cognitive and physical demands of already well-known established tasks like standardized tasks that have been mentioned at the beginning of this section. Therefore, the main goal of this work is to establish requirements to design such a workplace simulation environment and how it can be evaluated in terms of capabilities to induce cognitive workloads that match those encountered in real-world situations. To ensure research to be as efficient as possible, this simulation environment should be free for that purpose, lightweight and easy to set up and operate. Lightweight means, the simulation environment should run on a standard consumer notebook. Easy means, that setting up the software can be accomplished in less than 30 minutes and without in-depth knowledge of the ATC domain. Furthermore, non-ATC participants can be trained to operate single tasks of the simulation within 15 minutes. It should be possible to integrate different kinds of multitasking optimization algorithms. Also when it comes to altering or extending the simulation environment by means of changing or adding scenarios, this should be possible in a standardized and efficient way. Moallemi et al. [MSJ⁺19] describe scenarios in the context of ATC simulation as "stories or examples from the real world. These may be narratives that bring out a problem situation, or sequences of behavior and possible contextual description. They are used to guide design decisions and are commonly used for guiding software design and system development". Additionally, to perform research in the context of human-machine-interfaces (Human-Machine Interface (HMI)), the environment must allow to alter certain aspects (i.e. positioning of displays and controls). The HMI is a very prominent factor in ATC workplace design, as it ensures that all the important information is readily available and understandable for the ATC staff [MSJ⁺19]. Furthermore, it must allow straight forward operation of present technology and issuing commands to plane in standard tasks like "Separating air traffic using surveillance information or manual procedures", "Detecting and resolving potential conflicts" and "Maintaining flight data, including accepting flight plans and updates from controllers" [MSJ⁺19].

Evaluation that has been done in context of this work and included an in-depth literature research concluded that simulation environments fulfilling these requirements do not exist yet. The existing solutions either lack the implementation of state-of-the-art technologies like Controller Pilot Data Link Communications and Automatic Dependent Surveillance - Broadcast or have their shortcomings when it comes to configurability, possibility of adding or altering scenarios or the HMI. Additionally, many existing simulation environments for ATC are targeted towards training of ATC controllers, and thus being too complex to allow efficient research and user studies. One such user study has been performed within the scope of this work to create a proof-of-concept and ensure the newly created simulation environment fulfills its goals when it comes to multitasking research. The results of the study, that has been performed with participants having ATC domain knowledge and participants without it, have been analyzed according to current standards and included in this work.

1.2 Research Questions

- RQ 1: Which functional and non-functional requirements are needed to design an ATC simulation environment for multitasking research?
- RQ 2: Which perceived cognitive workload can be imposed by the ATC simulation environment, compared to real-world situations?

1.3 Expected Results and Success Criteria

The following results are expected to consider the work within this thesis as a success:

- Establishing functional and non-functional requirements needed to design an ATC simulation environment for multitasking research, backed by most recent literature dealing with ATC and multitasking research.
- Creating an actual implementation of an ATC simulation environment, realizing requirements of RQ 1 within the Unity Game Framework. Furthermore, the simulation environment should be free for research, lightweight and easy to set up and operate. Lightweight means, the simulation environment should run on a standard consumer notebook. Easy means, that setting up the software can be accomplished in less than 30 minutes and without in-depth knowledge of the ATC domain. Furthermore, non-ATC participants can be trained to operate single tasks of the simulation in less than 15 minutes. It should be possible to integrate different kinds of multitasking optimization algorithms.
- The above-mentioned implementation should be capable of inducing similar cognitive workloads than real-world situations.

1.4 Research Methods

The following section describes which research methods are applied in this work and how.

1.4.1 To Answer Research Question 1: Systematic Literature Review

To answer RQ 1, we want to know

- Which functional and non-functional requirements are needed to design an ATC simulation environment for multitasking research?

To answer this question, parts of the *systematic literature review* methodology is used. Kitchenham [Kit07] describes it as "identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of

interest. Individual studies contributing to a systematic review are called primary studies; a systematic review is a form of secondary study".

Kitchenham et al. [KC⁺07] proposes a 3-step approach when it comes to applying the methodology. These are the following:

Planning a Systematic Literature Review

After the need for a review has been identified, a review has to be planned, i.e. properties of the review as review questions, review methods references are specified [KC⁺07]. "After the planning went successful, a review protocol is to be established. A review protocol specifies the methods that will be used to undertake a specific systematic review" [KC⁺07]. In the last planning step, the review protocol should be evaluated. "The protocol is a critical element of any systematic review. Researchers must agree upon a procedure for evaluating the protocol. The adviser of this work, Dr. Wintersberger will also evaluate the review protocol.

Conducting the Review

Once the protocol has been agreed, the review can start. Findings in literature will be systematically collected and aggregated.

Reporting the Review

Finally, the data gathered during the conducting step is summarized and evaluated.

1.4.2 To Create the Concept and Implementation of the ATC Simulation: Design Science

As the main research methodology in this work, the *Design Science in Information Systems Research Framework* from Hevner et al. [HMPR04] is applied. The implementation of an ATC simulation environment for multitasking research, to answer this research question, should result in the creation of an information system. It should lead to an improvement of the effectiveness and efficiency of organizations performing multitasking research. "The main outcome of applying design science research should be an artifact (concept, model, instantiation) which solves an existing problem. The way of solving the problem or the artifact, which solves the problem itself, should impose significant contributions to existing knowledge" [HMPR04].

During the whole research process, well established and accepted evaluation methods must be used, to satisfy the demands on scientific rigor, which is elementary within the design science framework. This build-and-evaluate loop is typically iterated a number of times before the final design artifact is generated" [Mar02].

Design Evaluation As already mentioned above, the utility, quality, and efficacy of a design artifact must be *rigorously demonstrated via well-executed evaluation methods*

[HMPR04]. To evaluate the implemented simulation environment, i.e. the design artifact, the *experimental* approach will be used. Read more about it in subsection 1.4.3.

To evaluate the concept and implementation, the following experiments will be executed with different groups of participants:

- 1) Setting up and running the simulation environment and the implemented ATC tasks with 5 experts of the multitasking research group around Dr. Wintersberger.
- 2) Running the simulation environment and the implemented ATC tasks with 5 experts knowing ATC procedures and interviewing them afterwards as stated in subsection 1.4.3.

Structured interviews with the participating experts will be held afterwards, with the goal to evaluate whether the simulation environment fulfills its defined functional and non-functional requirements as specified in section 1.2. Seaman [Sea99] defines that interviews are "used to collect opinions or impressions about something" and that "a structured interview is described as one in which "the questions are in the hands of the interviewer and the response rests with the interviewee".

1.4.3 To Answer Research Question 2: Scientific Experiments

To answer RQ 2, we want to know

- Which perceived cognitive workload can be imposed by the ATC simulation environment, compared to real-world situations?

To answer it, a scientific experiment will be conducted. According to Dodig-Crnkovic [DC02], "a theory is accepted based in the first place on the results obtained through logical reasoning, observations and/or experiments" and that a scientific experiment in terms of computer science is "most effective on problems that require complex software solutions [...]".

The simulation environment and the implemented ATC tasks will be operated and evaluated by 5 experts knowing ATC procedures. We identified pilots and air traffic controllers as suitable participants. After the experiments, questionnaires are given out to ask the participating experts:

- Whether the simulation environment depicts real-world procedures.
- Whether the simulation environment is capable of inducing cognitive workload, similar to real-world situations. To determine this, the standardized NASA Task Load Index (NASA TLX) questionnaire will be utilized.

1. METHODOLOGY

To conduct the experiment, the task performance of participants taking part within the evaluation phase of the design science approach, depicted in section 1.4.2, will be recorded. Borman et al. [BM97] define task performance as "the effectiveness with which job incumbents perform activities that contribute to the organization's technical core [...]". To measure the task performance of each implemented ATC task, metrics taking into account the very nature of the particular task will be used.

Literature Research

To gather answers to the first research question,

- Which functional and non-functional requirements are needed to design an ATC simulation environment for multitasking research?

a *systematic literature review* has been performed. The detailed methodology used, is described in subsection 1.4.1.

2.1 Review Protocol

The structure of the following review protocol is based on [Kit07] and [GG06].

Title	Which formal and non-formal requirements are needed to design an ATC simulation environment for multitasking research?
Reviewer	Matthäus Spindelböck
Supervisor	Dr. Philipp Wintersberger

Search terms	<ul style="list-style-type: none"> - ATC concepts - ATC workplace - ATC radar screen - ATC tasks - ATC rules - ATC simulation - Functional requirements of simulations - Non-functional requirements of simulations - Non-functional requirements of software - Multitasking research concepts - Multitasking performance
Databases/Sources	- Searching academic publications via Google Scholar and aviation authority publications on the website of EUROCONTROL.
Inclusion criteria	Publications from academia and aviation authorities answering the research question fully or partly.
Exclusion criteria	- Publications that are dealing with ATC simulations as a game solely, without having realism as a goal.
Quality assessment	<p>For assessing the quality of the found content, the following attributes according are used:</p> <ul style="list-style-type: none"> - Relevance - Generalizability - Applicability
Extraction strategy	For extracting information related to the research question, standard templates will be used and filled out for every relevant publication. This will ensure, that the inclusion/exclusion criteria and quality assessment will be obeyed properly.
Synthesis strategy	The information collected during the extraction will be aggregated, summarized and compiled into a review report.

2.2 Functional Requirements for the Simulation Environment

According to Malan et al. [MB⁺01], "Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks or functions the system is required to perform".

To structure the list of functional requirements found in the literature research, the following categorization will be introduced and the respective requirements described below.

2.2.1 Human-Machine-Interface

One of the most important and prominently visible component of every simulation environment is the HMI. Sohnle et al. [STS73] define it as "a simulation monitor that allows interaction with the model" and "provides the facilities for the user to interact with the simulation program directly from the terminal". Clema [Cle78] proposes that the *man-machine interface* should be "powerful, yet simple, easy to learn and use" and "have both *pre-simulator* and *post-processing capability*". With *pre-simulator capability*, he describes the possibility for the user to configure the simulation before starting it. Consequently, *post-processing capability* is described by him as the data reduction and analyses after the simulation.

The HMI typically features input and output (monitoring) functionality [STS73] [Cle78]. Among others, the following requirements for a HMI are stated by Sohnle et al. [STS73]:

- "Provision of good facilities to monitor the status of the simulation is of utmost importance. It must be possible to determine quickly and accurately if the simulation is proceeding as expected or if its progress has been influenced by a deviant factor which must be pinpointed and corrected, or at least observed closely".
- "The user should have control over the execution of the model in simulated time. He should be able either to specify the length of simulated time for which the simulation should proceed or to allow the simulation to run continuously.
- "An interrupt facility should be available to suspend the simulation at any point in simulated time".

Now we want to deal with the specialties of ATC simulation environments: Due to the safety relevant function of air traffic controllers, their highly complex workplace design must meet specific requirements. Chang et al. describe in their article about "Human performance interfaces in air traffic control" [CY10], that the "controllers must interact cooperatively with the components (as human performance factors) of the ATC system to ensure a safe, orderly and efficient flow of air traffic" and that "Uncooperative interactions between the controllers and the system components hold potential for human error and thus safety breakdowns".

Let's continue discussing the most important ones of these components:

Radar Display

The main monitoring component of radar-capable ¹ ATC workstation is the radar display [Kal03] [HE16] [FLN09]. It features a **map** [HE16] [Kal03] of the airport or ATC sector,

¹A radar display is not an absolutely mandatory component of an ATC workstation. Due to financial, organizational or geographical reasons, controllers might be required to separate plane by other means at some locations.

the controller at the workplace is in charge of. Depending on the type of area the controller is managing, the map features different types of underlays. Such underlays as described by [HE16] [FLN09] [CVCB23] are listed in Table 2.2. Furthermore, the radar echoes of the **planes** detected within the range of the radar station are displayed as a symbol on the map at their corresponding position [ARS⁺01]. The controller can now detect the location of plane and also their trajectory while they are moving. To extend this basic information, a **label** is placed next to the plane symbols. The label features usually at least the following basic data according to [ARS⁺01] [Ibr]. See Table 2.2. With the increasing use of Automatic Dependent Surveillance - Broadcast and Controller Pilot Data Link Communications, additional information is attached to the radar label [Ibr]. However, this is out of scope of this work.

Aircraft Communication

Communication between controller and pilot is done by either speech or Controller Pilot Data Link Communications [Kal03], a text-based service [Kal03]. To use the first means of communication, the simulation environment must be capable of speech recognition [VP14] [Kal03].

Simulation Control

The HMI must also feature elements for *simulation control* and *performance measurement* as described later on in subsection 2.2.3).

2.2.2 Simulation Scenarios

According to [MSJ⁺19], simulation scenarios are "stories or examples from the real world", and that these "may be narratives that bring out a problem situation, or sequences of behavior and possible contextual description". Real-world scenarios in the ATC domain are manifold and will differ depending on the type of en-route ATC sector or airport the controller has the responsibility for. Even if we consider one single ATC workplace, the day-to-day scenarios differ widely depending on operational factors. These can be the number of planes within the area of responsibility, type of flights, type of planes, time of the day or meteorological situation. According to Moallemi et al. [MSJ⁺19], "The process of extracting requirements from the given scenarios starts with identifying key training concepts aimed by each scenario". Hilburn [Hil04], Jurivicic [JVB11], Moallemi et al. [MSJ⁺19], Rottermann et al. [RWS⁺17] and Oh et al. [OJCL16] list the following most common scenarios and tasks of controllers. They are depicted in Table 2.2.

2.2.3 Simulation Control and Execution

After having collected the functional requirements for *simulation scenarios*, we want to continue, and deal with those requirements, that are necessary for controlling the simulation environment and executing the scenarios.

Clema lists [Cle78] in his description of *simulation stages* among others, the additional requirements to *simulation control* and *execution* which are listed in Table 2.2.

Among others, Sohnle et al. [STS73] describe as functional requirements for simulation environments the control mechanisms in Table 2.2.

2.2.4 Multitasking Performance Measurement Strategy

As the work of air traffic controllers imposes heavy cognitive demand and workload on them [Hil04], numerous work exists, targeting this domain. It makes it perfect for any research, tackling topics like human factors and errors, workload and of course multitasking performance. As air controllers are confronted with many different work tasks, which often have to be executed in a simultaneous manner, it is a heavily studied field, especially in the field of psychology. Benbunan et al. [BFAM11] define multitasking as the "result of time allocation decisions made by individuals faced with multiple tasks". Furthermore, they state that "Multitasking research is important in order to improve the design of systems and applications" and that "Since people typically use computers to perform multiple tasks at the same time, insights into this type of behavior can help develop better systems and ideal types of computer environments for modern multitasking users".

As the target of this work is not to enlighten psychological aspects of this workplace, no in-depth evaluation of that kind of will be done. However, in order to establish the functional requirements needed for creating the concept of a simulation environment for multitasking research depicting state-of-the-art ATC tasks, a somehow basic understanding of the relationship between the certain work tasks of ATC staff and their required cognitive demand on workload and multitasking is necessary. In a quite recent study, Socha et al. [SHV⁺20] from the Czech Technical University in Prague, Czech Republic performed in their work an experiment including multiple European air traffic controllers. In a simulation environment, they were given different tasks of current ATC procedures, published by the European Organisation for the Safety of Air Navigation (EUROCONTROL). The cognitive workload of the controllers was measured during performing the different tasks by measuring physical parameters like heart rate variability (HRV). Similar to this study, [MO02], [SVM⁺23] and [BWS96] measured the change of cognitive workload by changing influencing factors like given tasks, number of planes, pilot skills and types of planes during various ATC tasks. In order to collect requirements for a *simulation environment for multitasking research*, we need to introduce a strategy to assess the user's multitasking performance. This will allow to study the effects of introduced changes within the simulation environment. Such changes can target for example the HMI [BFAM11] [MNS⁺20] or the introduction of (software) assistance in task switching [MD00] [ABF15]. Moallemi et al. [MSJ⁺19] specifically mention for ATC simulation environments that a "Scenario Evaluator consists of evaluation and grading categories and acceptable performance data that is used to grade the students' performance" and that "several tasks are selected for the student to be tested on, and a record of their response to each situation is used for grading their proficiency".

Major metrics to measure multitasking performance while executing tasks are listed in Table 2.2. To implement the listed metrics in grading the user's performance realistically, the scenarios and tasks that should challenge them, must be chosen appropriately [FLN09] [ABF15].

2.2.5 Other ATC Simulation Environments for Research

Concerning freely available existing concepts or implementations of ATC simulation environments for research organizations, a variety of proposed solutions targeting different audiences are available. However, none could be found that unites all of the aspects of being

- Freely available for research organizations.
- Lightweight: Runs on a standard consumer laptop without additional hardware.
- Easy to set up and operate. That means, the simulation environment should run on a standard consumer notebook and that setting up the software can be accomplished in less than 30 minutes and without in-depth knowledge of the ATC domain. Furthermore, non-ATC participants can be trained to operate single tasks of the simulation in less than 15 minutes.
- Extendable to integrate different kinds of multitasking optimization algorithms.

All older implementations like ATC-lab(Advanced) [FLN09] and TACUND [HH99] are missing recently introduced modern ATC features like Controller Pilot Data Link Communications and Automatic Dependent Surveillance - Broadcast. Both being systems, changing the way ATC staff works [CACK⁺19] [AOM⁺14]. Other implementations however, e.g. ATCOSima [CACK⁺19] or Shiomi's work [SSF⁺], cannot be considered lightweight due to their amount of additional hardware or size of installation. EUROCONTROL's Escape features a simulation environment that is targeted towards researchers being experts in the ATC domain. For instance, Antolović [Ant20] wrote her entire master's thesis about setting up and configuring Escape as well as creating scenarios for this environment. Also SmartAtms [KMPT97] and BlueSky [HE16] target audiences doing in-depth researching of Air Traffic Management and not for multitasking.

2.2.6 Summary

Table 2.2: Functional requirements for an ATC simulation environment for multitasking research

Radar display map underlays [HE16] [FLN09] [CVCB23]	<ul style="list-style-type: none"> - ATC sector borders - Weather radar echoes - Navigational aids - Airways, standard arrival routes (STARs), and standard instrument departures (SIDs) - Airport layout (runways, taxiways etc.)
Plane labels [ARS ⁺ 01] [Ibr]	<ul style="list-style-type: none"> - Aircraft call sign - Aircraft track - Aircraft groundspeed - Aircraft altitude - Aircraft squawk
Common ATC tasks [Hi104] [JVB11] [MSJ ⁺ 19] [RWS ⁺ 17] [OJCL16]	<ul style="list-style-type: none"> - Issuing approach, landing, taxi or departure clearances - Issuing en-route clearances - Directing planes to hold or change direction, altitude or speed to maintain appropriate separation between them. - Detecting possible collision hazards between planes themselves and issuing instructions to avoid them. - Detecting possible collision hazards between planes and obstacles or planes and the terrain and issuing instructions to avoid them. - Approving of requests from the pilot. - Handing off planes between different ATC sectors and controllers. - Assisting planes in navigating or operational issues. - Assisting planes in emergency situations.
Simulation stages, control and execution [STS73]	<ul style="list-style-type: none"> - Loading the application - Initializing the simulation - Checking in and checking out of the simulation - Dynamic control of the simulation - Prompting and anomaly messages - Displaying of options, parameters and commands - Data acquisition and logging - Performance monitoring and trace messages - Summary reports

Multitasking performance metrics [MD00] [BKPK06] [LCCN16] [ABF15] [BKPK06]	- Speed of execution - Quality/Accuracy of execution - Errors made
--	--

2.3 Non-Functional Requirements for the Simulation Environment

Chung et al. [CNYM12] describe non-functional requirements in software engineering as those that are determined by "operational costs, performance, reliability, maintainability, portability, robustness and the like". As the *systematic literature research* did not reveal all-embracing content relating to non-functional requirements specifically in the simulation domain, we will deal with general non-functional requirements in software engineering and extend them with ATC simulation related remarks. The following categorization of non-functional requirements (NFR) is based on the book "Non-Functional Requirements in Software Engineering" by Chung et al. [CNYM12]:

2.3.1 Performance

Time

The most important time-related requirements of software systems are:

- Response time: The time the software needs to react upon any user input or request. In an ATC simulation environment, or an ATC workplace in general, the HMI must be responsive in a way, it won't interfere with the timely manner of controlling the plane.
- Processing time: The time a software needs to execute a calculation or task. Also here, the software must not introduce unnecessary delays and disruptions to the simulation scenarios.

Space Requirements

Space requirements deal with the consumption of space on the following hardware components of a computer system:

- Main memory needs: The amount of storage in the main memory of the computer system, the software needs during execution.
- Secondary storage: The amount of storage the software needs on non-volatile storage like hard disks.

2.3.2 Reliability and Security

Confidentiality

This requirement denotes that the software environment must not disclose sensitive information [VL09]. In the simulation domain there might be data of the user like name, e-mail address or their performance in solving tasks. The collected data should be kept secret according to applicable law and not disclosed, if not on purpose and agreed upon.

Integrity

- Accuracy: Data accuracy is "The extent to which data correctly represents the real-world entities or events it is intended to describe" [Ala23]. From this definition of *accuracy*, specific non-functional requirements for ATC simulations can be concluded instantaneously: The scenarios, tasks and behavior of simulated objects must be simulated in a correct and realistic way.
- Completeness: "The degree to which all relevant data points are present and available for use" [Ala23]. This is a very challenging requirement in developing a simulation environment, as creating a *complete* virtual model of the real-world is very time- and resource intensive. Being it during the development and testing phase, or when it comes to operating the environment and providing the necessary *performance* and *storage* needs.

Availability

Katukoori [Kat95] defines availability as "a percentage measure of the degree to which machinery and equipment is in an operable and committable state at the point in time when it is needed". Concerning simulation environments, it can be concluded, that the system should be as stable as possible, not producing system errors, that lead to a disturbance of the simulation experience.

2.3.3 User-friendliness

Chung et al. [CNYM12] state that "[...] the design and functioning of the components (human performance factors) of the ATC system, when interacting together with the controller, should ideally be matched with personal attributes and needs of the controller". While in their article they target real-world workplaces, we go without saying, that this must also hold true for the environment that simulates it.

2.3.4 Cost

The cost to design, develop, test and maintain the software system. These can be kept low, by using efficient procedures and tools during all the phases mentioned above.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Implementation

3.1 Description of the Simulation Environment

This section will describe the different components of ATChallenge, being the *HMI* that should simulate an ATC workplace, the scenarios that are depicted within the simulation and the simulation control functions. Additionally, the design of the simulation environment's data structures, which allow a flexible, extensible and efficient way of creating airport layouts and air traffic scenarios, will be described.

When it seems appropriate and necessary for the reader with no aviation background to understand the following sections, background information about the theoretics in ATC and piloting is given.

3.1.1 Human-Machine Interface

As explained in subsection 2.2.1, the *HMII* must provide all functions for the user of the simulated ATC workplace to interact with the simulation environment and its components.

Considering the outlined requirements, the graphical user interface (GUI) shown in Figure 3.1 has been developed.

The GUI, which is optimized for Full HD resolution (1920px x 1018px) is basically separated into two parts:

The first and left part is the **radar display** on the left side. It features the base map of an airport, with its runways (RWY), taxiways (TWY) and buildings. Furthermore, the dotted line represents standard arrival routes (STARs) and standard instrument departures (SIDs) of planes. On the radar display, all the planes that are within the coverage of the radar station are displayed at their corresponding positions. The plane symbol rotates automatically to point into the direction, the plane is currently moving.

3. IMPLEMENTATION

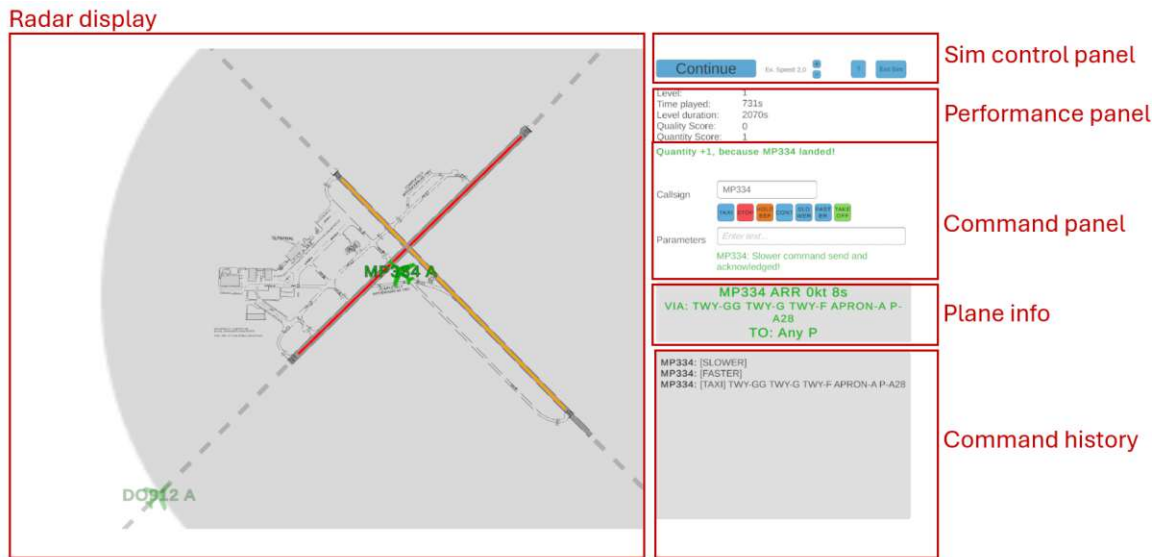


Figure 3.1: HMI of ATChallenge

Every plane has its call sign attached to it. To zoom in and out on the radar display, the scroll wheel can be used. It is possible to drag it in any direction by pressing the right mouse button and moving the mouse in the corresponding direction. See Figure 3.2.

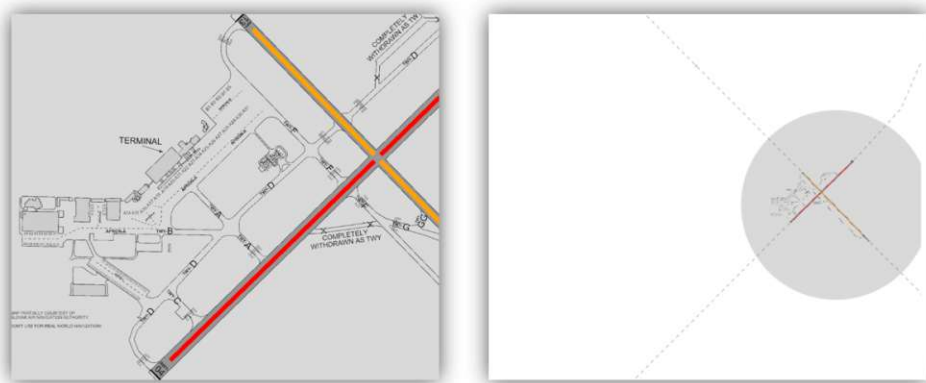


Figure 3.2: Zooming and dragging the radar display in ATChallenge

On the right part of the GUI, the following elements are placed in logical groups, starting from the top:

Firstly, there are the **basic simulation control elements**. They allow the user to pause, continue or exit the simulation. Furthermore, its execution speed can be altered. Read more about it in subsection 3.1.4.

Below the simulation controls, the **game, level and performance information** is

listed. They allow the user to monitor their task performance in real time. The details can be found in subsection 3.1.3.

All the input fields and buttons to assign **commands with their parameters to planes**, are located in the middle of the right pane. All commands are depicted in section 3.1.2.

The two gray panels below the plane commands section, are to display all the **relevant information about a selected plane** and **previously given commands**. The information shown on these areas is listed in more detail in section 3.1.2 as well.

Additionally to the standard GUI, depicted in Figure 3.1, the simulation environment is also capable of displaying **pop-ups** to the user. As an example, the pop-up showing level instructions is shown in Figure 3.3.

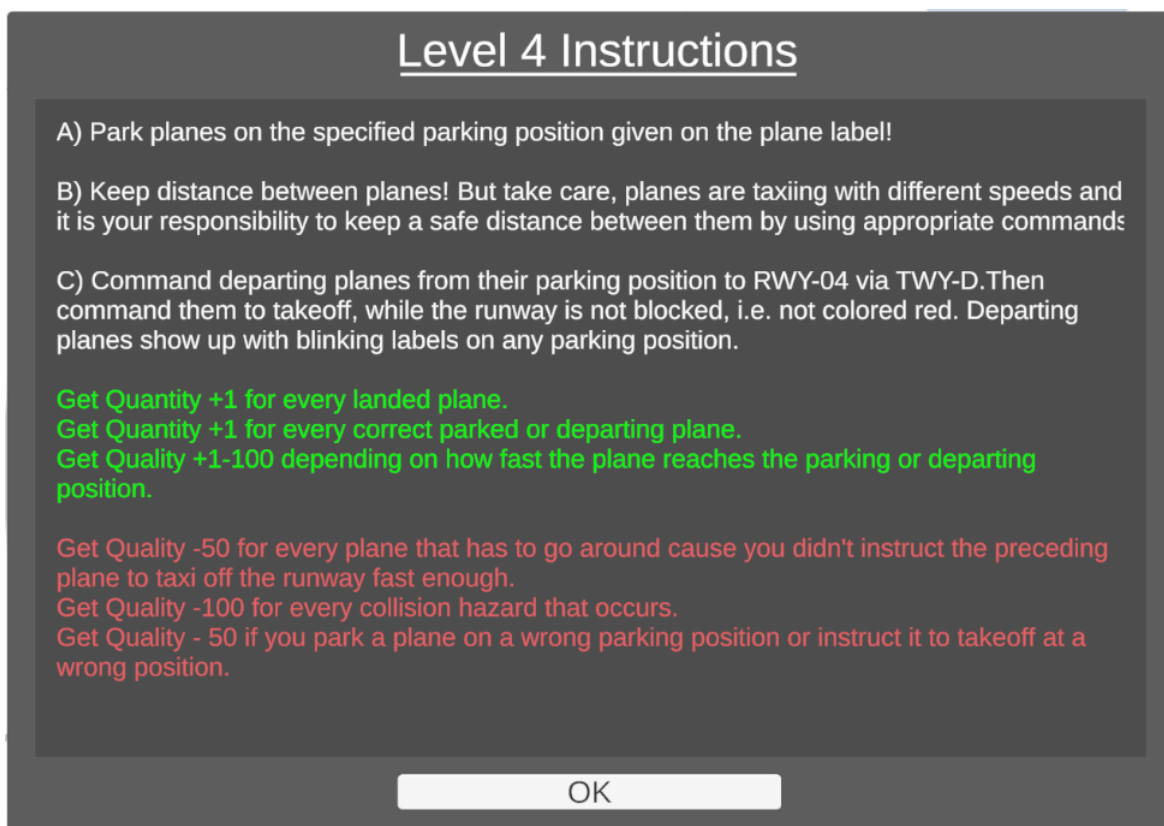


Figure 3.3: Pop-up displaying level instructions.

3.1.2 Scenario

The main purpose of creating the simulation environment ATChallenge was to measure multitasking performance of its users. That being said, the scenarios for the user, which has the role of a virtual air traffic controller, have to be carefully selected, to fulfill the

expectations. As already listed in Table 2.2, main metrics for multitasking performance are speed, quantity and quality of task execution. For effectively measuring these metrics, we propose and implemented the following simulation scenario: The user takes over specific main task of a *tower* air traffic controller. This type of controllers is, depending on the size of the airport, usually in charge of controlling planes during their last section on the approach or their first section in the climb, as well as their movement on runways. At smaller airports, the tower controller is also in charge of controlling planes that taxi between terminals and their parking positions and the runways. Controlling in terms of ATC usually means: Giving instructions to planes and monitoring their proper execution. Furthermore, detecting potential collision hazards between planes themselves or other objects like terrain, is depending on the type of flight, mandatory tasks for the controller. It already can be concluded, that performing all these tasks within a multitasking environment, in a timely and qualitative manner can induce a high cognitive load for the person in charge [S⁺85]. The more planes, that are in the area of responsibility of a controller, the higher the workload usually gets [MO02].

To be able to simulate such a high workload, a typical airport control tower scenario was implemented in ATChallenge. To have more options in creating complex tasks, an airport layout that features crossed runways, is used. The layout is based on that of *Bratislava, Milan Rastislav Štefánik*. Planes are arriving on two *standard arrival routes (STARs)* and departing on two *standard instrument departures (SIDs)* (the dotted grey lines in Figure 3.5). See Figure 3.4 for a drawing.

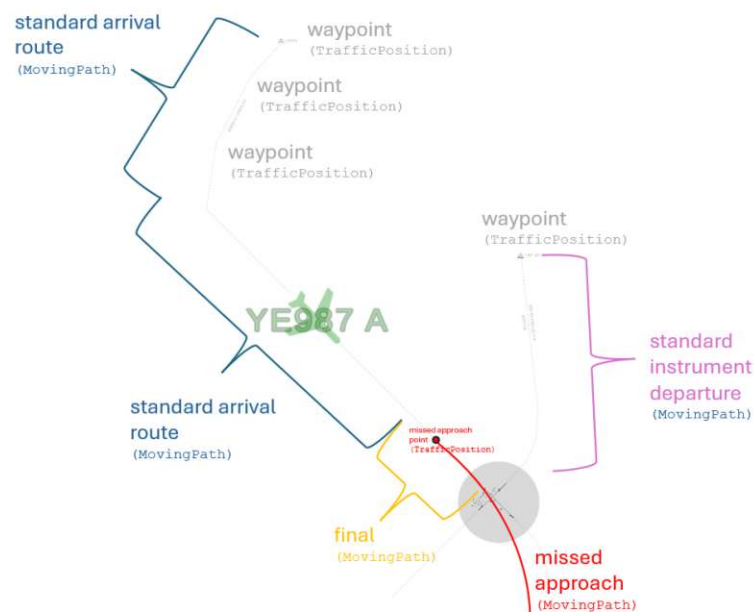


Figure 3.4: Arrival and departure routes of the *GenericAirport1*.

1. Guiding **landed planes** from the point they come to a stand on the runway, via a sequence of taxiways to their parking position.

2. Guiding **departing planes** from their parking position to the right entry of the runway, they are supposed to depart.

Guiding in these cases means, giving them instructions via which sequence of taxiways they should move to reach their desired location. Always taking into account which is the route that is the shortest and where there is no collision with another plane to be expected. Furthermore, to optimize traffic flow, taxi faster or slower commands can be issued. The commands should be given as fast as possible, otherwise the *Plane* accumulates delay or a succeeding plane to land on the same runway must abort take off and go around. Both will reflect on the *performance measurement scores*. When planes are to depart on a runway or cross a such, the controller has to check, if there is no other plane departing or landing that could impose a collision hazard with the departing aircraft. As described in subsection 3.1.3, the speed, quantity and quality of the guiding activities are measured and recorded. The higher the simulation level gets, the interval and number of planes, and the higher the workload for the user gets.

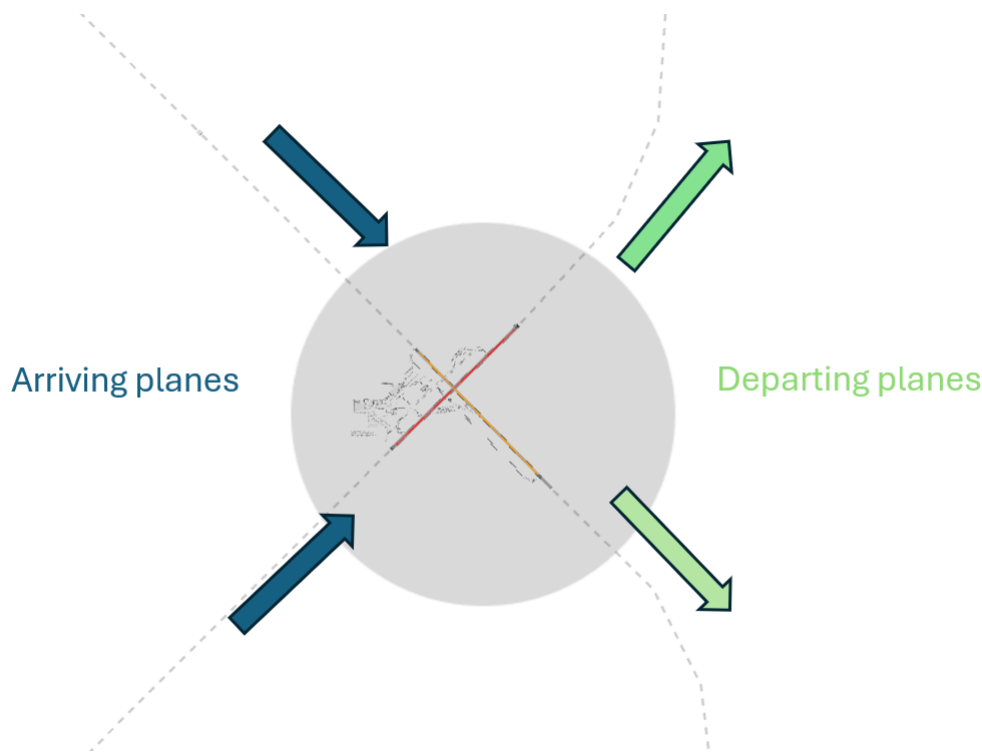


Figure 3.5: Directions of arriving and departing planes in ATChallenge

Guiding Arriving Planes to the Assigned Parking Position

Depending on the simulation level and its configuration, planes are arriving on the runways randomly, within the borders of a configurable interval. The standard settings

take into consideration separation minima defined internationally by the ICAO.

As seen in Figure 3.6, arriving planes show the letter *A*, following their call sign. Furthermore, the runway, the plane is about to land, is automatically colored *red*, which means, it is blocked for any other plane to enter it or depart from it. The other runway that is crossing the first one, is automatically colored *orange*, to mark that other planes can cross this runway, but not depart or land on it. This system of mutual blocking runways, should guide the user to issue instructions to planes in a safe and collision-free manner, the point where the two runways cross, being a crucial one.

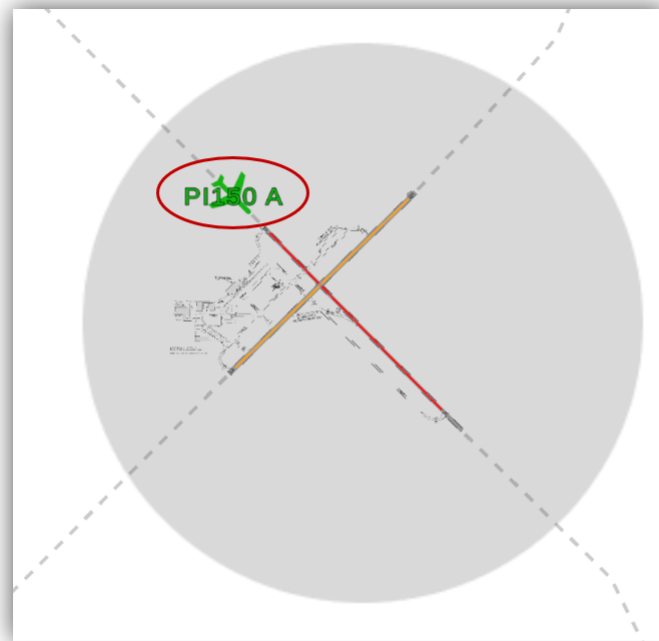


Figure 3.6: An arriving plane is marked by the letter *A*, following its call sign.

If the user clicks on (selects) an arriving plane, its call sign and plane information is shown in the panel on the right (see Figure 3.7). Additionally, the plane's symbol and call sign on the radar display will be highlighted and those of possible other planes in the ATC sector will get less visible. This allows the *selected plane* to be monitored efficiently.

The fields shown are described in detail in Figure 3.8.

The parking position that has been assigned (reserved) for the landed plane is shown on the plane information panel. It's now the task for the user to guide the plane to that position safe and fast. To locate the destination position at the airport, multiple parking positions are grouped locally together and assigned a letter. In the sample airport *GenericAirport1* that is included in ATChallenge, there exist 4 blocks (aprons) that contain parking positions (A, B, C, D). Within every of these aprons, the different positions are numerated. Thus **P-C2** means **Apron-C**, parking position **2**.

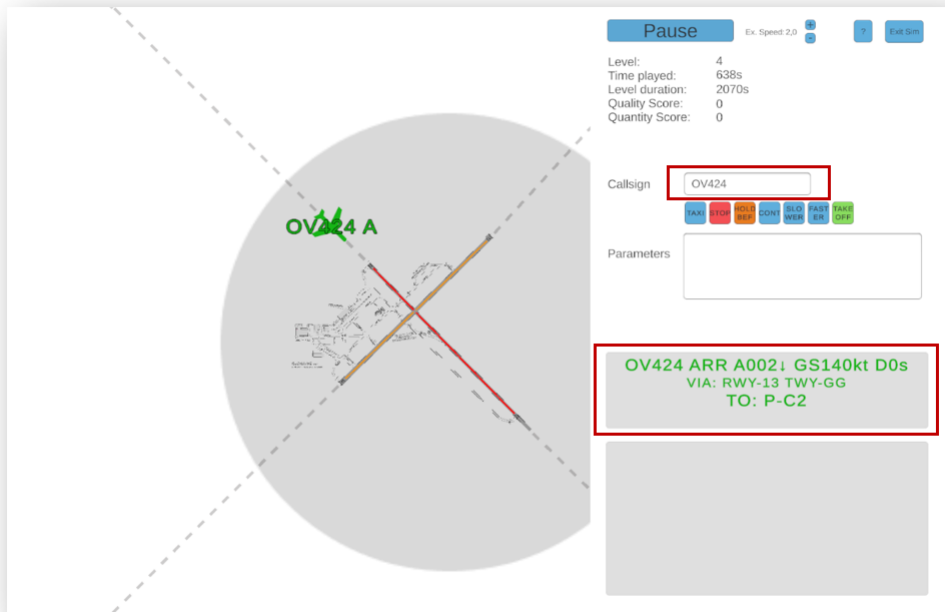


Figure 3.7: The information panel for an arriving plane.

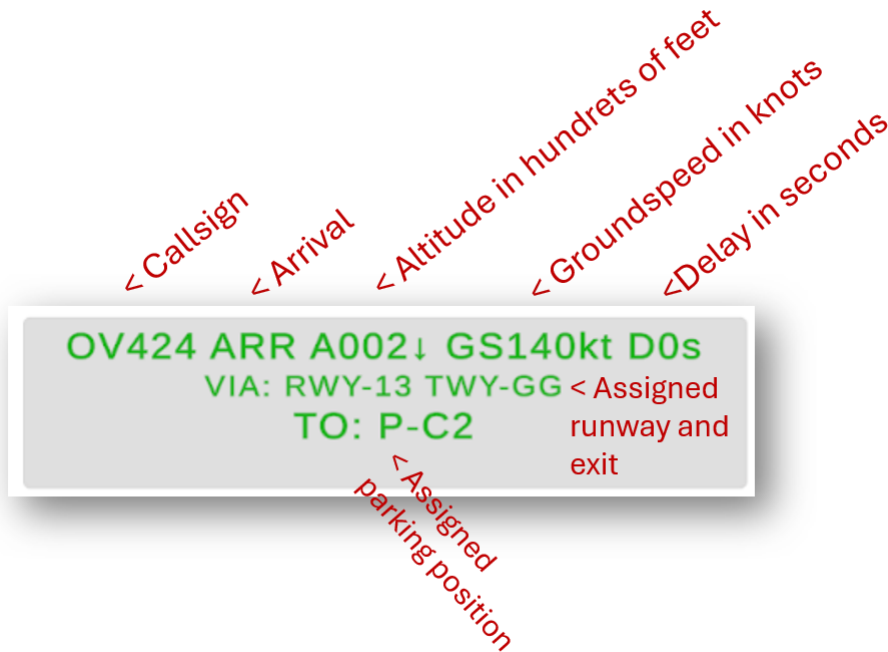


Figure 3.8: Plane information panel in detail.

3. IMPLEMENTATION

It's now the task to find a route on the taxiways of the airport to reach the destination parking position of the plane on the shortest way, without creating a situation, that can lead to a collision with another plane. Furthermore, the commands should be given fast enough, so that the plane vacates the runway on which it has landed, before the next plane that approaches the same runway has to abort and go around. Within the simulation environment we assume that the meteorological situation at the airport introduces low horizontal visibility. Thus, *low visibility procedures* are active at the airport, as the plane's pilots might not see each other well and early enough, to avoid collisions. That means, the plane's pilots have to primarily rely on the instructions of the (air) traffic controller, when it comes to collision avoidance.

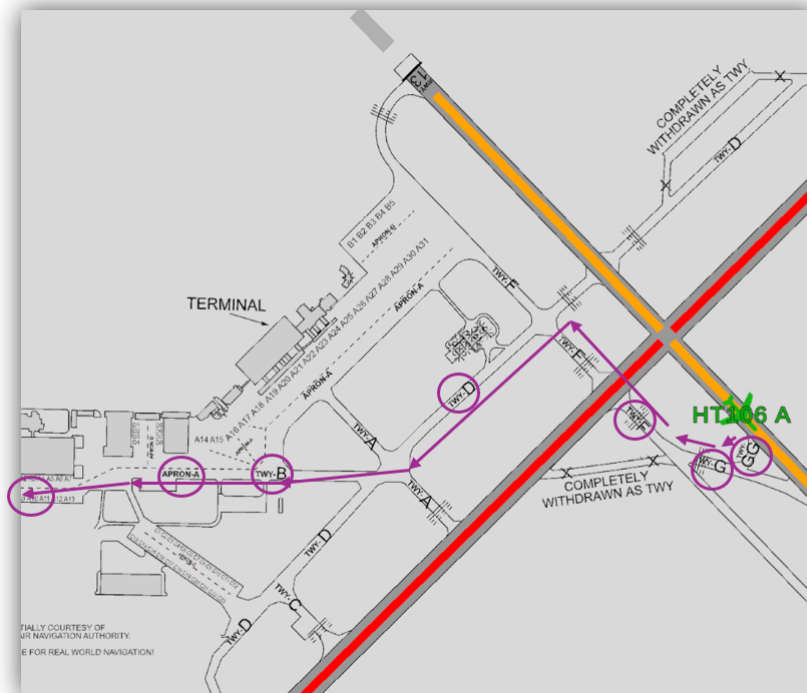


Figure 3.9: A route on the taxiway network of the sample airport *GenericAirport1*.

To send the selected route as a command to the selected plane, there exist two options:

- 1) Click in the right order on the identifier of the taxiways, the plane should take to reach the parking position. The identifiers are placed at the airport map and consist of the **TWY-** indicator, followed by an alphanumeric sequence. E.g. **TWY-A**, **TWY-B**.
- 2) The route can also be entered in the command *parameters* input field by using the keyboard. E.g. **TWY-GG TWY-G TWY-F TWY-B APRON-A P-9** would instruct the plane to follow the route that is depicted on Figure 3.9.

If the selected route is erroneous, e.g. between two taxiways entered in a row there is no intersection, an error message is shown. The user can now correct the route, via editing

the *parameters* input field or delete it entirely, and select it again by clicking on the taxiway identifiers. After the route has been correctly selected or entered, it is about time to send it via the **TAXI** command to the plane (Figure 3.10). The plane will now start to taxi on the route.

While the plane is moving to its parking position, the user can instruct it to **stop it, to continue when it has been stopped, to taxi faster or to taxi slower**. Furthermore, a new route can also be entered, while the plane is already taxiing. It can be done by repeating the steps to assign a route initially.



Figure 3.10: Buttons to send commands to planes.

Planes will automatically hold (stop) before crossing a runway. The user has to check, whether the runway can be crossed. As already described earlier, a runway that is colored red, shall not be crossed, as there is conflicting traffic landing or departing from this runway. Orange-colored runways can be crossed. Once the conditions are met, the user can instruct the plane to **continue**.

If the user wants a plane to stop automatically before an intersection with another taxiway, e.g. to give way to another plane, this can be done with the **hold before** command. While the plane is already in motion, just enter or click on the identifier of the intersecting taxiway into the *parameters* input field, and press the hold before button.

Eventually, when the plane reaches the correct parking position, the task is completed.

Guiding Departing Planes to the Assigned Runway and Intersection

Depending on the simulation level and its configuration, departing planes can show up on any parking position randomly within the borders of a configurable interval. The planes show the letter *D*, following their call sign. See Figure 3.11.

The user has now to click on the plane, to see at which runway and intersection the plane is to depart. In the example shown in Figure 3.12, the plane is to depart at runway **04** and enter it via taxiway **D**. Now the user has to perform the same steps, which were described in section 3.1.2 just in a reversed order. That means, finding a route on the taxiways of the airport to reach the assigned runway via the assigned taxiway (intersection) on the shortest way, without creating a situation, that can lead to a collision with another plane.

The plane automatically stops at the holding point (see Figure 3.13) before entering the runway. If the runway is grey and not colored red or orange, which means it is blocked by another plane, the **takeoff** command can be given.

Eventually, the plane starts the takeoff and the task is completed.

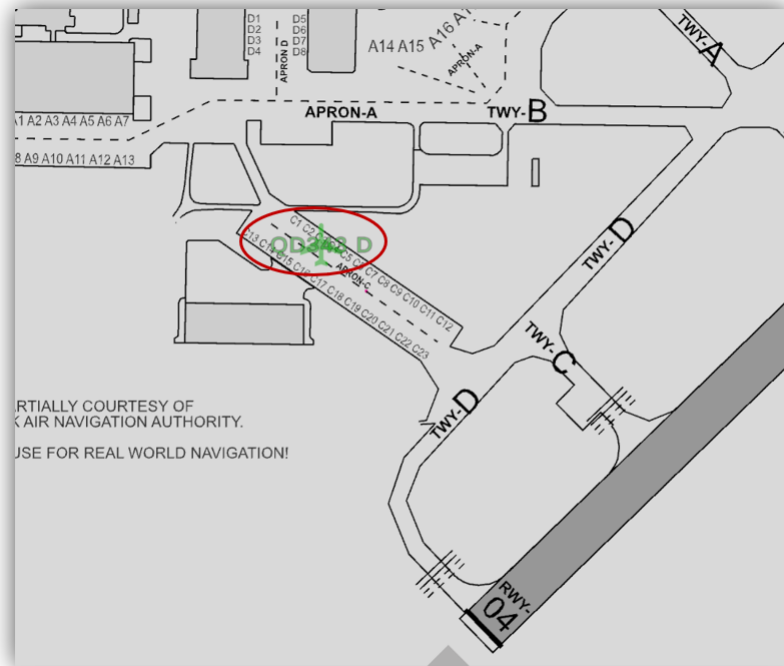


Figure 3.11: A plane ready at its parking position to depart.



Figure 3.12: Plane information of a departing plane, showing assigned runway (RWY-04) and intersection (TWY-D) to depart.

3.1.3 Performance Evaluation

Regarding the performance measurement and evaluation strategy of the user, that has been set up within the simulation environment, we will track the metrics that are listed in Table 2.2 and additional statistics.

To accomplish this, two main scores have been established: Firstly, the **quantity score**, which reflects the number of tasks (i.e. guiding a plane to its parking position or departure runway), a user accomplished during the given time. For every task accomplished, the score will increase for a defined amount. Secondly the **quality score** which tracks the quality of task execution and errors made. Errors made and collision between planes,

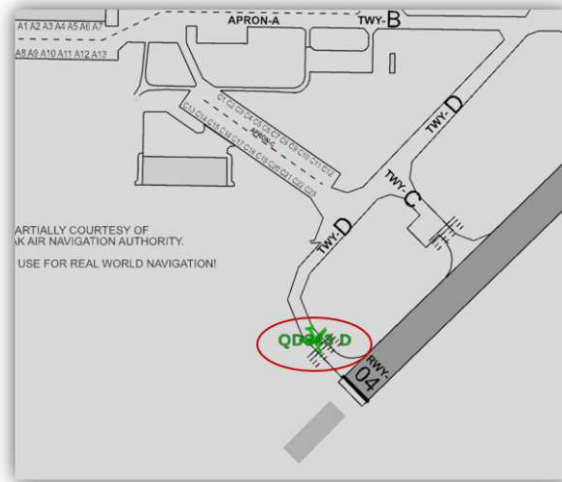


Figure 3.13: Plane ready for departure at the holding point on TWY-D to the assigned runway (RWY-04).

that happened, will reduce the quality score accordingly. Furthermore, every single plane has its own delay counter, which is also displayed on the *plane information panel*. Additionally, the color of the plane symbols on the *radar display* changes according to their accumulated delays. The delay metric will reflect also in the *quality score*. All the time, an event occurs, that alters the scores, a message containing reason and amount of score change is displayed on the GUI. This instant feedback-loop allows the user to learn from their mistakes immediately.

Together with the sample airport *GenericAirport1*, 5 different **simulation levels** were established. The higher the level gets, the more simultaneous tasks were established. Before any level starts, the user gets shown a pop-up containing the level instructions and also details about the level is scored, i.e. which action and events lead to an increase or decrease of scores. See Figure 3.3 for a screenshot of such a pop-up. The initial level that will be executed automatically after starting the simulation environment can be configured as described in section 3.3. Whether the next higher level is automatically executed, after the previous one has ended, can be configured as well.

At the end of every level, a summary of the scores obtained and additional statistics for that level are shown. At the end of the simulation, a summary of the scores obtained and additional statistics for all levels are shown. Furthermore, the scores are written into text files, together with the user's name. The location of the statistics directory will be shown to the user in the summary statement. This makes collecting performance data during studies convenient.

3.1.4 Simulation Control

When the simulation environment is started, the user is asked their name and instruction pop-ups are shown. Thereafter, the simulation starts by itself with the first level that has been preconfigured. When standard settings (see section 3.3) are used, the simulation is sped up significantly until the first plane arrives in the vicinity of the airport. This is to reduce initial waiting time for the user.

After a preconfigured time, the level ends automatically and the level statistics are shown. Depending on the configuration, the simulation environment automatically starts the next higher level after the user confirms the statistics.

As it can be seen in Figure 3.14, the user has also the availability to **pause/continue** the simulation or open the **level instructions again** by clicking on the ? button. Another important feature to mention here is the function to change the **simulation execution speed**. Depending on the configuration, the user can modify the execution speed within predefined intervals. The scenarios have been constructed in a way, that with *execution speed* = 1, the events occurring in the simulation are timed like in real-world situations. *Execution speed* < 1 slows the simulation below real-world speed and *execution speed* > 1 speeds it up beyond real-world speed.



Figure 3.14: Simulation control elements on the GUI.

3.2 Technical Realization

3.2.1 Unity Game Engine

The implemented ATC simulation environment *ATChallenge* is based on the *Unity Game Engine*, version 2022.3 LTS (long term support release). "Unity (commonly known as Unity3D) is a game engine and integrated development environment (IDE) for creating interactive media, typically video games" [Ala23]. It has been first released in 2005 and was meant to be an affordable solution for amateurs and professionals to create computer games out of the box [Ala23]. Unity runs on a variety of platforms. Windows, Unix, Android and iOS are just the most prominent ones. It is also capable of creating a HTML5/Javascript build of most games.

Unity features frameworks to create both 2D and 3D games and simulations. The behavior of the so called *GameObjects*, can be altered by using different scripting languages. *ATChallenge* has been programmed with using scripts written in C# exclusively. The game engine allows advanced graphical rendering in 2D and 3D as well as a machine learning environment.

To allow easy sharing of created work among other creators, Unity provides integrated cloud storage and hosting functionality, called *Unity Asset Store*. Unity is released under different licenses. For educational purposes and use in research, Unity offers a free plan.

Why Unity for Creating an ATC Simulation Environment

One of the main reasons for using Unity as a platform to implement the simulation environment is its out-of-the box capability of moving objects, in Unity called *GameObjects*, in a 3D space, while displaying them projected on a 2D map. This is exactly what we need, when we think of an ATC workplace: On the one hand, there are the planes moving in x, y and z (height) direction while on the other hand we have the radar display map, that shows them basically projected on a map and moving in x and y direction only. The movement in z-direction is not projected on the map but shown as an additional textual information next to the displayed plane symbols. Without a lot of effort, the creation of a radar display map was quickly realized. Furthermore, there is the functionality in Unity to easily assign scripts to *GameObjects* to give them a unique behavior. This comes in handy, as usually when one wants to create a simulation environment containing individual planes, all of them might behave differently. For instance, in which direction they move, how fast they fly, how much delay they might have and so on. If desired, it is even possible to change these values during the run-phase via the *Unity Inspector* menu, making it a perfect tool to debug behavior of planes in the development phase of the simulation. We also want to mention the high performance Unity offers: As there might be up to roughly 100 planes, all with their own unique behavior, being displayed simultaneously on the radar display map of the simulation, efficient run time calculation must be possible. Frame rate must be sufficiently high to allow a smooth simulation experience, which was the case in all scenarios tested during developing the simulation environment. Additionally, there is the capability of Unity to create a HTML5/Javascript based build out of most projects. If the build is hosted online, everybody with a suitable end-device can access the simulation environment from anywhere. As a last point to mention, the free license Unity offers for educational purposes, keeps the necessary budget needed in developing the simulation environment low.

Used Extensions

There was just one extension necessary, to realize the simulation environment. Otherwise, only basic features of Unity have been used. The *Vector Graphics Plugin* was necessary to map ground layouts of airports to the radar display map. As these layouts must be enlarged to a significant high resolution, the Scalable Vector Graphics (SVG) format was used to create them. Although this plugin has been marked "experimental" by the author, it performed well so far without any bugs or performance impairment.

3.2.2 Design Principles and Patterns

Martin [Mar00] wrote about the different levels in software architecture: "At the highest level, there are the architecture patterns that define the overall shape and structure of software applications. Down a level is the architecture that is specifically related to the purpose of the software application. Yet another level down resides the architecture of the modules and their interconnections. This is the domain of design patterns, packages, components, and classes". Wherever possible, the Model-View-Controller (MVC), which is the topic of the following section, has been used.

Model-View-Controller (MVC)

"In MVC, the View displays information to the user and, together with the Controller which processes the user's interaction, comprises the application's user interface. The Model is the portion of the application that contains both the information represented by the View and the logic that changes this information in response to user interaction" [LR01]. The consequent usage of this design pattern allows software system to be constructed and maintained easily, as it can already be deferred from the structure or pattern, where some function or code might be located. Furthermore, it allows the change of certain modules or code (e.g. in the view) without necessarily having to touch code in one of the other modules (model, controller).

Due to the nature of the Unity Game Engine and the interaction of its components, it was in certain cases necessary to contravene the strict usage of MVC. Otherwise, it would have made some function unnecessarily complicated, which would act against the original reason to choose MVC as a pattern. However, this was seldom the case, as can be seen in Table 3.1

Table 3.1: Categorization of classes according to MVC

Model	<ul style="list-style-type: none"> - Airport - BlockingSituation - InvalidMovingSequenceException - LevelStatistics - MovingPath - TrafficPosition
View	<ul style="list-style-type: none"> - MapZoom - CameraDrag
Controller	<ul style="list-style-type: none"> - MainManager - ClickToInput - PlaneLabelClickProcessor - UserInputListener - UserInputProcessor - Utils

Fitting into multiple categories	- GenericAirport1 - Plane
---	------------------------------

C-Sharp Code Conventions

"Coding standards encourage programmers to follow a uniform set of guidelines determined by the requirements of the project and organization rather than by the programmer's familiarity or preference" [Sea08]. Benefits of using coding standards are an increase in maintainability of the source code, especially when more than one programmer or team is involved [Sea08]. Examples for rules within coding standards are naming and possible capitalization of classes, methods and variables as well as how the code is formatted in general. I.e. when and how much indentation is used when it comes to structuring of code blocks and statements like *if* and loops like *for*, *while*.

Throughout the source code of ATChallenge, the *coding conventions for C#*, that are published online by Microsoft, are followed.

C-Sharp XML Documentation Tags

According to Nielebock et al. [NKK⁺19], "Maintaining a program is a time-consuming and expensive task in software engineering. Consequently, several approaches have been proposed to improve the comprehensibility of source code. One of such approaches are comments in the code that enable developers to explain the program with their own words or predefined tags".

Additionally to applying the *coding conventions for C#*, the source code has also been extensively commented, using the *Recommended XML documentation tags*, published online by Microsoft. As the domain of *HMI* and aviation in general, makes extensive use of domain-specific terms, an effort has been made, to add explanations in the source code documentation, where such terms are used.

Automated Creation of HTML Source Code Documentation Package To ease the understanding of the *HMI* simulation environment's architecture, package structure and source code even more, the software *Doxygen* has been used to convert the comments within the source code were automatically converted into a HTML documentation package. Doxygen automatically recognizes references between classes and methods and therefore, generates links between them in the documentation package by itself. Furthermore, a navigation menu, listing all classes is generated. This makes navigation within the documentation package very efficient.

Example using XML documentation tags and automated HTML pack generation:

```

/// <summary>
///     Constructs a new aircraft with the given params and inserts it into the
simulation.
///     This method just inserts the plane, but doesn't instruct it to start moving.

```

3. IMPLEMENTATION

```
/// Plane.SetMovingSequence() and Plane.StartMoving()
/// has to be called after this method.
/// </summary>
///
/// <param name="startPosition">
///     The TrafficPosition at which the plane should appear the first time in the
///     simulation.
/// </param>
///
/// <returns>
///     A MovingPath or null if not found.
/// </returns>
///
public Plane PlaneEntrySim(TrafficPosition startPosition)
```

The following HTML page will be generated by Doxygen:

[!h]

Figure 3.15: Documentation page created by Doxygen

The screenshot shows a Doxygen-generated documentation page for the `PlaneEntrySim()` method. The title is `◆ PlaneEntrySim()`. Below the title, the signature is `Plane MainManager.PlaneEntrySim (TrafficPosition startPosition)`. The description states: "Constructs a new aircraft with the given params and inserts it into the simulation. This method just inserts the plane, but doesn't instruct it to start moving. `Plane.SetMovingSequence` and `Plane.StartMoving()` has to be called after this method." The **Parameters** section lists `startPosition` as "The `TrafficPosition` at which the plane should appear the first time in the simulation." The **Returns** section lists "A `MovingPath` or null if not found." At the bottom, it says "Definition at line 424 of file `MainManager.cs`."

3.2.3 Human-Machine Interface

All components of the GUI, that have been introduced in the last paragraphs, were constructed and positioned via Unity's graphical WYSIWYG editor. This editor features a library of different GUI elements. Being to display or input information. All GUI elements used in ATChallenge, stem from this library. The behavior of all these elements is defined in the *controller* sections of the source code.

Additionally, sound effects are used to support the user in recognizing different events, e.g. successfully accomplished tasks or errors made.

3.2.4 Data Structures and Controllers

The entire simulation environment has been designed with a focus on easy extensibility and configurability. The most important data structures are described in the following sections.

TrafficPosition

Planes in ATChallenge move **exclusively** on predefined *MovingPaths*, which are defined by two or more *TrafficPositions*. See Figure 3.16.

A *TrafficPosition* represents a three-dimensional location for any plane, that can be positioned at any point within the simulation environment. Either on the ground or in the air. Planes move on a *MovingPath* from one *TrafficPosition* to the next one on a straight trajectory that is interpolated by Unity.

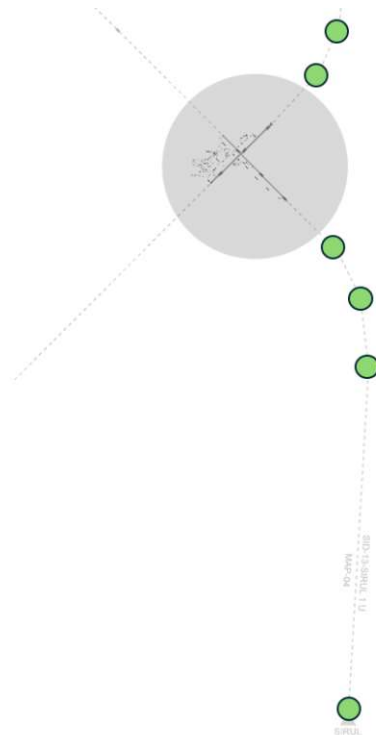


Figure 3.16: *TrafficPositions* (green) forming a *MovingPath* (gray dotted) which is representing a Standard instrument departure (SID).

Every *TrafficPosition* can have two names. The names are set when the *TrafficPosition* represents a significant point at the airport or in the air. Such points can be terrestrial features like cities or mountains or holding points on airports. The property *path* is storing the name of the *MovingPath* this *TrafficPosition* belongs to. X , Y , Z coordinates

are describing the location of the *TrafficPosition* in the three-dimensional space of unity. *TrafficPositions* can also represent intersection between multiple *MovingPaths*.

```
public class TrafficPosition
{
    private string name1;
    private string name2;

    private string path;

    private float coordX;
    private float coordY;
    private float coordZ;

    private string intersection1;
    private string intersection2;
    private string intersection3;
    private string intersection4;
    private string intersection5;
}
```

MovingPath

As described before, a *MovingPath* represents a predefined path, either straight or not, planes can move along. Such paths can be runways and taxiways on an airport or standard instrument departures and standard arrival routes in the air.

It can store a name and all the *TrafficPositions* that form the *MovingPath* in an ordered *List*. First *TrafficPosition* on the *MovingPath* will be at *index = 0* in the *List*, last one at the highest index of the *List*.

```
public class MovingPath
{
    private string name;
    private List<TrafficPosition> positions;
    private List<BlockingSituation> blockingList;
}
```

Furthermore, a *MovingPath* can be blocked by one or more planes simultaneously. For every plane blocking it, a *BlockingSituation* entry is inserted into a *List*.

For an explanation of how *TrafficPositions* and *MovingPaths* are related to each other, see Figure 3.4.

Simulation Control

For the overall simulation environment control functionality, the *MainManager* class is responsible. From within, the loading of the airport, its scenario and planes are governed.

```
/// <summary>
///     Constructs a new aircraft with the given params and inserts it into the
///     simulation.
```

```

/// This method just inserts the plane, but doesn't instruct it to start moving.
Plane.SetMovingSequence and Plane.StartMoving()
/// has to be called after this method.
/// </summary>
/// <param name="startPosition">The TrafficPosition at which the plane should appear
the first time in the simulation.</param>
/// <returns>
/// A MovingPath or null if not found.
/// </returns>
public Plane PlaneEntrySim(TrafficPosition startPosition)
{
    if (GetCurrentPlaneCount() < maxConcurrentPlanes)
    {
        // Finding a unique callsign
        string callsign = null;
        while (callsign == null)
        {
            string callsignToTest = Utils.GetRandomString(2).ToUpper() +
                UnityEngine.Random.Range(100, 999);
            if (GameObject.Find(callsignToTest) == null)
                callsign = callsignToTest;
        }

        currentStatistics.CreatedPlanes++;

        Plane newPlane = Plane.ConstructNewPlane(startPosition, callsign);
        return newPlane;
    }
    else
        return null;
}

```

The basic functions like **starting/pausing/continuing** the simulation and control of simulation execution speed are realized as well.

```

/// <summary>
/// Setting execution speed of the simulation
/// </summary>
/// <param name="timeScale">A float [0.0, 1.0]</param>
public void SetTimeScale(float timeScale)
{
    this.executionSpeed = timeScale > 0.1f ? timeScale : 0.1f;
}

```

The *HMI* and its *event listeners* are initialized.

```

airport = GameObject.Find("Airport").GetComponent<GenericAirport1>();
inputProcessor = GameObject.Find("UserInputProcessor").GetComponent<UserInputProcessor
>();

```

Furthermore, the class ensures correct recording of performance measurement-related metrics and forwarding it to the volatile (session) and non-volatile (disk) storage.

```

overallStatistics = new LevelStatistics();
currentStatistics = new LevelStatistics();

```

Every time, the *Plane* is encountering a situation that leads to an increase or decrease of any *scores*, it reports the event back to the *MainManager*, which then passes it on to the data structure *LevelStatistics* that is storing all the scores. The amount of increase or decrease for every different event is also defined at this location.

```
/// <summary>
/// The specified plane reports that it has tried to park on a parking space that was
/// not assigned to it
/// </summary>
/// <param name="p">The reporting plane.</param>
public void ReportWrongParking(Plane p)
{
    int qualityModifier = -30;
    currentStatistics.QualityScore += qualityModifier;

    scoreMessageType = "FAIL";
    scoreMessage = "Quality_" + qualityModifier.ToString("+#;-#;000") + ",_because_" +
        p.name + "_tried_to_park_on_a_wrong_position!";

    GameObject.Find("ScoreMinusAudio").GetComponent<AudioSource>().Play(0);
}
```

Airport

The standard terminology of areas, routes and points on an airport is depicted in Figure 3.17.

Architecture and Extensibility Airports are represented in ATChallenge the following way: There is *Airport* as base class, which contains all the properties and methods that are required to define an airport. To implement a specific airport, this base class has to be inherited and several abstract methods implemented. This architecture allows to extend ATChallenge with new airports, without changing code anywhere else outside the class that will represent the new airport. As scenarios depend on the airport implemented, they have to be implemented alongside within the new class.

Definition of Airport Layout As a first step, the base structure of an airport should be defined in the *UnityInspector* as shown in Figure 3.18. A good advice is, to copy that one from the sample airport *GenericAirport1*, place it under the parent *Airport* and give it the name of the new airport. After that, deactivate the whole *GenericAirport1* tree in the inspector. To load the new airport at the start of the simulation, copy the name you gave it, into the inspector field *airportName* of the component *MainManager*. Major graphical components that can be seen on Figure 3.18 are the *BlockedLines* which are necessary for the simulation environment to automatically block a runway as explained later on in section 3.2.4. The *ClickableDesignators* are necessary to make *MovingPaths*, e.g. taxiways selectable by clicking on their identifiers. See also section 3.1.2. *RunwayIntersection* and *ReferencePoints* are graphical elements that helped the author in constructing the Airport layout, but are not obligatory.

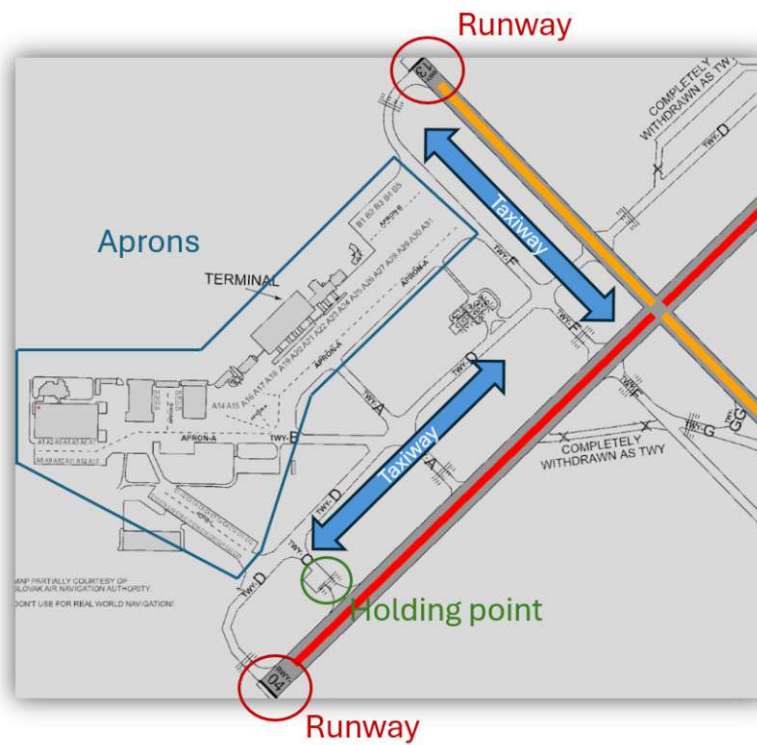


Figure 3.17: Naming of areas, routes and points on airports.

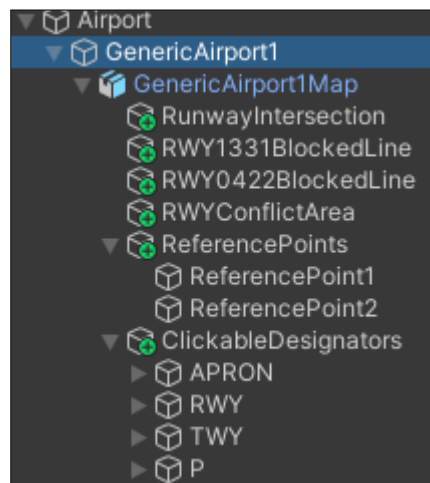


Figure 3.18: Base structure for a new airport in the Unity Inspector.

Thereafter, the base map of the airport has to be constructed and exported in the SVG format. The SVG image has to be imported in Unity, as a *sprite* component (see *GenericAirport1Map* component) and aligned and sized within the Unity 3D-space

accordingly.

Some class members have to be overwritten, to reflect the specialties of the new Airport. The simulation environment automatically puts the *camera focus point* initially at the middle of the *radar display*.

```
public override void Awake()
{
    base.Awake();

    // Values of the base class Airport have to be adapted specifically for this new
    // Airport.
    levelInitTimeScale = 30f;
    cameraFocusPointX = 440f;
    cameraFocusPointY = 459f;

    // Levels are defined in this class, the base class doesn't know about the number.
    nrLevels = 5;
}
```

To size the airport base map accordingly, ensure that a distance or line on the map (e.g. a runway), which length in meters in real world is known and calculate the length in Unity's internal object space distance by applying the following formula:

$$DistanceSimSpace = Distancereal - worldSpaceMeters * 0.217636$$

This conversion constant is defined and fixed within ATChallenge.

Now resize the base map, so that the line chosen previously on the map equals the length of *distance sim space* within the Unity Inspector.

As a next step the name, positions and intersections of runways, taxiways and parking positions at the airport have to be logically defined by overwriting the method *ConstructScenary()*. The whole airport layout is defined within this method. Now all standard arrival routes (STARs), standard instrument departures (SIDs), runways, taxiways, aprons, parking positions have to be defined as *MovingPaths*. The sample airport *GenericAirport1*, see also Figure 3.4 can be used as a reference, how the different *TrafficPositions* and *MovingPaths* should be named and are connected.

When one MovingPath should be connected to another, their last and first TrafficPosition must have the same coordinates and have mutual references as intersections to each other.

All *MovingPaths* have to be defined two times. One for each direction. This is necessary, as taxiways can be one-way, or the name and type of *TrafficPositions* might be different, depending in which direction the plane is moving on the path. For instance, one *TrafficPosition* that is located on coordinates (10, 10) can mark a holding point in one direction and no holding point in the opposite direction.

The following example shows how the taxiway **A**, of the *GenericAirport1* is correctly defined. The standardized identifiers (RWY, TWY, HP, P) must be used consistently

when naming *MovingPaths* for the simulation environment to work correctly. Compare with the depiction of *TWY-A* on Figure 3.17.

```

MovingPath twyASE = new MovingPath("TWY-A>SE"); // Name of MovingPath = Taxiway name (
    TWY-A) + direction (SE, i.e. south east)

twyASE.AddPosition(233.2f, 434.3f, 0f, "APRON-A>NW", "APRON-A>SW"); // First Traffic
    Position of the MovingPath, which is also an intersection to MovingPath "APRON-A>
    NW".
twyASE.AddPosition(279.4f, 386.8f, 0f, "TWY-D>NE", "TWY-D>SW", "TWY-B>E", "TWY-B>W");
    // Intersections with multiple other taxiways.
twyASE.AddPosition("HP-RWY-04", "HP-RWY-22", 305.4f, 360.1f, 0f); // This
    TrafficPosition represents the holding point to runway 04 and 22.
twyASE.AddPosition(321.6f, 343.8f, 0f, "RWY-04", "RWY-22"); // Last TrafficPosition of
    the MovingPath.

movingPaths.Add(new KeyValuePair<string, MovingPath>(twyASE.Name, twyASE)); // The
    newly defined MovingPath is added to the overall repository of the Airport.

// Definition of the opposite direction MovingPath
MovingPath twyANW = new MovingPath("TWY-A>NW"); // Name of MovingPath = Taxiway name (
    TWY-A) + direction (NE, i.e. north west)

twyANW.AddPosition(321.6f, 343.8f, 0f, "RWY-04", "RWY-22"); // Intersection with
    runways 04 and 22
twyANW.AddPosition(279.4f, 386.8f, 0f, "TWY-D>NE", "TWY-D>SW", "TWY-B>E", "TWY-B>W");
    // Intersections with several taxiways.
twyANW.AddPosition(233.2f, 434.3f, 0f, "APRON-A>NE", "APRON-A>SW"); // Intersection
    with APRON-A>NE and APRON-A>SW.

movingPaths.Add(new KeyValuePair<string, MovingPath>(twyANW.Name, twyANW));

```

Additionally to *MovingPaths* on the ground, the *MovingPaths* in the air have to be defined.

To troubleshoot the definition of *MovingPaths*, the debug option *coordinatesDebug*, which can be set in the Unity Inspector window of the component *Airport* comes in handy. It displays all *TrafficPositions* that have been defined as red points on the map.

Defining Conflicting Runways Conflicting runways are runways that don't allow arrivals and departures independently from each other. I.e. on the *GenericAirport1* (Figure 3.17) we have two runways that cross each other. This leads to the implication, that e.g. when there is currently a plane landing on runway 13, there is no departure on runway 04 possible, as this would create the risk of those two planes colliding at the intersection between those two runways.

ATChallenge features a complex algorithm, that decides depending on the position of arriving and departing planes, which runways are *blocked* for certain operation. The algorithm leads to different colorings of the runways on the radar display. See section 3.1.2 for the meaning of the different colors. However, for this algorithm to work, the conflicting runways have to be defined in the methods *BlockConflictingRunway* and *BlockConflictingRunway*. See source code comments on more details.

Finally, some static properties of the new airport class have also to be set. See the source code comments for more descriptions.

Scenario

As scenarios are closely tied to the airport layout, they are also defined within the implementation of the *Airport* class for each airport. All the data and behavior of the scenario have to be defined in the methods *InitLevel*, *ShowLevelInstructions*, *UpdateLevel* and *ReportLevelPhase*.

For illustrating purposes, some crucial code of the sample airport's *GenericAirport1* implemented scenario is explained in further detail. When implementing new scenarios, this scenario and its source code comments are always a good starting point and reference.

InitLevel Method This method is automatically called before the execution of any level. For every different level, the actions to initiate it, have to be defined. In the following listing, the initial camera position and *simulation execution speed* are set. The latter is set to a quite high value at the beginning of every level to shorten the time, the first planes arrive in the vicinity of the airport.

```
// The level phase instructs the scenario to behave differently depending on which
// phase the level is advanced.
// I.e. arriving planes are moving very fast, until the first plane arrives at the
// airport.
levelPhase = "CONSTRUCTING_SCENARIO";
oldLevelPhase = string.Empty;

// We speed up the game at the beginning of each level.
// Will be lowered in ReportLevelPhase() as soon as the first plane reaches the airport
.
mainManager.ExecutionSpeed = LevelInitTimeScale;

// We are adjusting the camera zoom to well fit the airport and surrounding area in the
// field of view.
mainManager.SetCameraFOV(9000);
```

Thereafter, the interval of arriving planes in seconds is configured.

```
nextArrivalRWY13 = 0; // As soon as the level starts, the first arriving plane starts
// to fly towards the airport for runway 13.
nextArrivalRWY04 = Single.MaxValue; // Initially, no arrivals for runway 04 are
// scheduled.

nextDepartureRWY13 = 630;
nextDepartureRWY04 = 730;
```

Additionally, the interval within planes arrive with a randomly chosen time difference has to be defined.

```
if (level > 0 && level < 4)
{
    // The intervals in seconds within planes are arriving at the airport
```

```

// WARNING: 240 sec is minimum for the scenarios to work correctly and
// realistically, but should be a lot higher than on the other runway
// otherwise the airport will just be overloaded, which would result in a lot of go
// -arounds and almost
// impossible to find a window between arriving planes to let planes takeoff.
minIntervalArrivalsRWY13 = 240f;
maxIntervalArrivalsRWY13 = 360f;
}
else if (level >= 4)
{
// We don't want departures on RWY-13 in level 4, departures on RWY-04 are
// sufficient.
if (level == 4)
nextDepartureRWY13 = Single.MaxValue;

// See remark under (level < 4) above.
// Longer interval of arrivals from level 4 up, cause we now also have to squeeze
// the departing planes into the
// traffic flow.
minIntervalArrivalsRWY13 = 500f;
maxIntervalArrivalsRWY13 = 700f;

// The intervals in seconds new planes for departure show up
minIntervalDeparturesRWY13 = 120f;
maxIntervalDeparturesRWY13 = 300f;
minIntervalDeparturesRWY04 = 240f;
maxIntervalDeparturesRWY04 = 360f;
}

```

UpdateLevel Method This is the method that is called for every simulation frame and is in charge of defining the behavior of the scenario, after it has been initialized. It also contains functionality to randomly create new planes and assign and position them for arrival or departure.

```

Plane newPlane = mainManager.PlaneEntrySim(GetMovingPath(assignedParkingPos + ">OUT").
    GetFirstPosition());
if (newPlane != null)
{
    newPlane.VTaxi = UnityEngine.Random.Range(Plane.MinTaxiSpeed, Plane.MaxTaxiSpeed);
    newPlane.CurrentSpeed = newPlane.VTaxi;
    newPlane.FlightType = "DEPARTURE";
    newPlane.AssignedParkingPosition = assignedParkingPos;
    newPlane.Status = "PARKED";
    newPlane.AssignedRwy = "RWY-" + newPlaneRwyNr;
    newPlane.AssignedRwyIntersection = randomIntersection ? Utils.RandomChoose(
        newPlaneRwyNr.Equals("13") ? "TWY-F>NW_TWY-D>NE_TWY-G>SE" : "TWY-D>SW_TWY-C>
        SE_TWY-A>SE_TWY-F>SE") : "TWY-D>SW";
}

```

Plane

All data, behavior and display of *Planes* is contained in this class.

Several constants define *Plane* speeds during the different phases of flight and display options. The latter defines how large the symbols of planes appear on the *radar display*.

3. IMPLEMENTATION

Although the class has been prepared to define different characteristics like speed for different planes, this option is not used in the current version of the simulation environment, as it would need further adaptations in the definition of the scenario.

```
/// <summary>
/// Landing speed of this Plane. Should be the same for every Plane right now as
/// otherwise collisions
/// would occur on the approach paths.
/// </summary>
public const float VRef = 140f;

/// <summary>
/// Maximum airspeed of this Plane. Should be the same for every Plane right now as
/// otherwise collisions
/// would occur on the approach paths.
/// </summary>
public const float MaxAirSpeed = 250f;

/// <summary>
/// Maximum taxi speed of this Plane.
/// </summary>
public const float MaxTaxiSpeed = 30f;
```

```
/// <summary>
/// Customizing standard Plane symbol size.
/// </summary>
public const float StandardSymbolFontSize = 40f;

/// <summary>
/// Customizing Plane symbol size while Plane is parked.
/// </summary>
public const float StandardParkingSymbolFontSize = 5f;

/// <summary>
/// Customizing Plane label size
/// </summary>
public const float StandardLabelFontSize = 20f;
```

The implementation of the functions that allow the *Plane* to move correctly on the predefined three-dimensional *MovingPaths* (see section 3.2.4) is defined within the method *Move*.

As described in section 3.1.2 and section 3.1.2, the user has the task to instruct the *Plane* via which route of *taxiways* the plane has to taxi, i.e. move. Such a route on the sample airport *GenericAirport1* could be as displayed in Figure 3.9. There was now a function to implement, which converts this route on multiple *MovingPaths* into a correct sequence of coordinates, the *Plane* has to move along. This algorithm imposed a significant unforeseen challenge during implementation and testing, which took considerable time to finish. When one considers for instance, the intersection between **TWY-F** and **TWY-D**, the algorithm has to extract solely from the route that has been defined by the user (**TWY-F TWY-D TWY-B APRON-A P-A1**), whether the *Plane* has to move left or right at this intersection to follow the route. It has therefore, to *think ahead* on the route and

consider the next taxiways and other situations on the route to calculate this decision. Finally, the complex algorithm has been defined in the method *SetMovingSequence*.

The class *Plane* contains also all rules, that are applied, when it transitions through the different phases of flights. Depending on them, the plane has to accelerate, decelerate, stop, turn and so on. All these rules are defined in the method *CheckSegmentTransition*. The following listing shows all the actions that have to be done during the *Plane* transitions from **final approach** to **runway** and the taxiing off from the **runway**.

```

//// BEGINNING LANDING ROLL ////

if (previousTrafficPosition.Path.StartsWith("FINAL") && nextTrafficPosition.Path.
    StartsWith("RWY"))
{
    // Landing roll
    status = "ROLLING";
}

//// ENDED LANDING ROLL ////

else if (status.Equals("ROLLING") && flightType.Equals("ARRIVAL") && stopped)
{
    // We reached the point to exit the runway
    status = "LANDED";
    currentSpeed = VTaxi;
    awaitingInstruction = true;

    airport.UnblockRunway(assignedRwy, this);
    airport.BlockRunway(assignedRwy, this, status);
    airport.UnblockConflictingRunway(assignedRwy, this);

    mainManager.ReportLanding(this);
}

//// TAXIING OFF THE RUNWAY ///

else if (previousTrafficPosition.HasIntersectionOrNameStartsWith("RWY") &&
    nextTrafficPosition.Path.StartsWith("TWY"))
{
    // Taxiing off the runway
    status = "TAXI";
    airport.UnblockRunway(assignedRwy, this);
    airport.UnblockConflictingRunway(assignedRwy, this);
}

```

Performance Measurement

When it comes to performance measurement, there are two elementary steps to describe: Whenever a *Plane* finds itself in a situation that could influence the performance scores, it reports this to the *MainManager*, which then modifies the scores in the corresponding data structure *LevelStatistics*. *MainManager* holds two instances of this class. One for the current level that is played, and one for the entire session of the simulation environment

3. IMPLEMENTATION

This real-world to record performance scores both on *level base* or on *simulation session base*. The following scores and statistics are recorded:

```
/// <summary>
/// Time, the level has been started
/// </summary>
private DateTime startTime;

/// <summary>
/// The level this LevelStatistics belongs to
/// </summary>
private int level = 0;

/// <summary>
/// The name of the user
/// </summary>
string userName = string.Empty;

/// <summary>
/// Planes the user parked successfully
/// </summary>
private int parkedPlanes = 0;

/// <summary>
/// Planes the user dispatched for departure successfully
/// </summary>
private int departedPlanes = 0;

/// <summary>
/// How many tasks (e.g. parking planes) the user has accomplished in this level
/// </summary>
private int tasksCompleted = 0;

/// <summary>
/// Quality score (errors)
/// </summary>
private int qualityScore = 0;

/// <summary>
/// Quantity score (speed, efficiency)
/// </summary>
private int quantityScore = 0;

/// <summary>
/// Collision hazards created by the user
/// </summary>
private int collisionHazards = 0;

/// <summary>
/// Overall planes that were created during the level by the simulation
/// </summary>
private int createdPlanes = 0;

/// <summary>
/// The time in seconds, the user played the level
/// </summary>
private float levelSeconds = 0f;

/// <summary>
```

```
/// The initial execution speed of the level
/// </summary>
```

At the end of every level and the simulation session, these values are displayed on the *HMI* in a pop-up and also written on disk. See also subsection 3.1.3.

3.3 Configuration Options

Table 3.2: Configuration options for the simulation environment

MainManager (Unity Inspector)	<ul style="list-style-type: none"> - <code>airportName</code>: The name of the Airport to be loaded for the simulation session. - <code>startLevel</code>: The starting level of the simulation. - <code>initalExecutionSpeed</code>: The initial execution speed of the simulation. - <code>secondsToCompleteLevel</code>: The maximum duration of a level in seconds, after it is completed. - <code>tasksToCompleteLevel</code>: The number of tasks (e.g. parking, taking off planes) that have to be accomplished, to fulfill conditions to complete a level. - <code>automaticLevelIncrease</code>: If set true, the simulation automatically switches to the higher level, after the end of the previous one. - <code>automaticCollisionResolution</code>: If set true, collided planes are removed automatically after time specified in <code>CollisionResolutionTime</code>. - <code>maxConcurrentPlanes</code>: Maximum number of planes that exist within the simulation at any given time. - <code>clickableRoutes</code>: If true, then routes (e.g. taxi routes) can be constructed by clicking on the designators (names) of the movement paths on the map. - <code>debugMode</code>: Switching on debug messages.
MainManager (Class Header)	<ul style="list-style-type: none"> - <code>CollisionResolutionTime</code>: The time it takes, after collided planes are automatically taken off the simulation if <code>automaticCollisionResolution</code> set to true.
MainManager (Methods)	<ul style="list-style-type: none"> - Adapting score calculation algorithms for single events: See methods beginning with name "Report...".
Airport (Unity Inspector)	<ul style="list-style-type: none"> - <code>coordinatesDebug</code>: If true, then all the coordinates that are used for defining the airport <code>MovingPaths</code>, parking position etc. will be shown as small points on the map.

Airport (Class Header)	<ul style="list-style-type: none"> - LevelInitTimeScale: The execution speed at level init to speed up the arrival of the first planes. Is then subsequently reduced by the simulation till the preset one. - RwyBlockedLineWidth: When a runway is blocked, a colored line is indicating it. Specify the width here. - Creating a new airport and scenario as described in section 3.2.4. Interval of arriving and departing planes can be configured then individually.
Plane (Unity Inspector)	<ul style="list-style-type: none"> - MinPlaneReactionTime: Minimum reaction time a Plane needs to execute a command given by the user. - MaxPlaneReactionTime: Maximum reaction time a Plane needs to execute a command given by the user.

**Plane
(Class Header)**

- VRef: Landing speed of this Plane. Should be the same for every Plane right now as otherwise collisions would occur on the approach paths.
- MaxAirspeed: Maximum airspeed of this Plane. Should be the same for every Plane right now as otherwise collisions would occur on the approach paths.
- MaxTaxiSpeed: Maximum taxi speed of this Plane.
- MinTaxiSpeed: Minimum taxi speed of this Plane.
- VRefToVApproachFactor: Which factor to multiply VRef (landing speed) with to obtain this Plane's speed on the approach segment of the flight.
- VRefToVInitialClimbFactor: Which factor to multiply vRef (landing speed) with to obtain this Plane's speed on the climb segment of the flight.
- VRefToVCruiseClimbFactor: Which factor to multiply vRef (landing speed) with to obtain this Plane's speed on the cruise segment of the flight.
- DelayScoringInterval: The interval within the delay status of this plane changes with its accumulated delay in seconds.
- StandardSymbolFontSize: Customizing standard Plane symbol size.
- StandardParkingSymbolFontSize: Customizing Plane symbol size while Plane is parked.
- StandardLabelFontSize: Customizing Plane label size.
- HoldBeforeDistance: The distance a Plane will hold before reaching an intersection, it should hold before.

MapZoom (Unity Inspector)	<ul style="list-style-type: none"> - minFieldOfView: Sets the minimum field of view, which is basically the zoom level of the camera. The higher the fov, the smaller the zoom level. - maxFieldOfView: Sets the maximum field of view, which is basically the zoom level of the camera. The higher the fov, the smaller the zoom level.
MapZoom (Class Header)	<ul style="list-style-type: none"> - PovAdjustmentFactor: The factor which governs the size of plane symbol and label in regard to current POV of the camera.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

User Study and Evaluation

To evaluate the concept and implementation of the simulation environment against the success criteria (section 1.3) a user study has been performed.

4.1 Methods

4.1.1 Participants

To gather additional insights into possible performance differences between study participants with ATC domain knowledge (**DK group**) and participants without domain knowledge (**NDK group**), two groups were established. DK group (**n=7**) consisted of licensed and active pilots, while the NDK group (**n=6**) consisted of students at Vienna University of Technology.

Table 4.1: Study participant's demographics and self-assessed computer gaming experience.

Group	ID	Gender	Age	Computer Gaming per Week	Computer Gaming Skills Self Assessment [1 is "Don't play at all" and 7 is "Professional level"]
NDK	1	Female	31	<1 hour	2
NDK	3	Male	26	6 - 10 hours	6
NDK	4	Male	32	<1 hour	6
NDK	5	Male	35	0	1
NDK	6	Female	33	0	1
Average:			31.40	~1.80 hours	3,20

Table 4.1 continued from previous page

Group	ID	Gender	Age	Computer Gaming per Week	Computer Gaming Skills Self Assessment [1 is "Don't play at all" and 7 is "Professional level"]
DK	D1	Male	36	<1 hour	3
DK	D2	Male	56	<1 hour	2
DK	D3	Male	66	0	2
DK	D4	Male	66	1 - 5 hours	5
DK	D5	Male	73	<1 hour	4
DK	D6	Male	60	0	2
DK	D7	Male	56	0	1
Average:			59	~0.64 hours	2.71

4.1.2 Materials

Eye-tracker recordings were taken of the participants while performing tasks of the simulation environment. These recordings have not been analyzed in the scope of this work but might be used for further follow-up work. After the participants finished each level, they were given a short online questionnaire, containing self-assessment questions, derived from the **NASA-TLX** questionnaire. This questionnaire, that has been developed by Hart and Staveland [HS88] in 1988, covers the six scales *Mental Demand*, *Physical Demand*, *Temporal i.e. time pressure*, *Performance*, *Effort* and *Frustration*. The participant is asked to self-assess their perception between *very low* and *very high*. Following the last level, all participants took part in an oral interview that also contained detailed questions about personal habits and demographic data of the participants.

For all study participants, the same hardware (standard consumer type) was used.

4.1.3 Procedure

To gather additional insights into possible performance differences between study participants with ATC domain knowledge (**DK group**) and participants without domain knowledge (**NDK group**), two groups were established. DK group (**n=7**) consisted of licensed and active pilots, while the NDK group (**n=5**) consisted of students, at Vienna University of Technology. After a 15 minutes interactive introduction to the simulation environment with the possibility to ask for questions, all participants were asked to perform a standardized set of levels and tasks (**Levels 2, 4, 5**) within the sample *GenericAirport1* scenario. This ensured that the cognitive demand to accomplish the tasks was increased with each level. For more details on the simulation environment's performance evaluation strategy, see subsection 3.1.3.

Played levels:

- Level 2: Parking arriving planes.
- Level 4: Parking arriving planes and guiding departing planes to the departure runway via always the same taxiway intersection.
- Level 5: Parking arriving planes and guiding departing planes to the departure runway via a randomly assigned taxiway intersection.

All levels have been played **2070 seconds** with **simulation execution speed 2.0**. The whole study session took 1 hour for every participant, and was organized at a location at Vienna University of Technology and in a room of a flying school at Graz Airport.

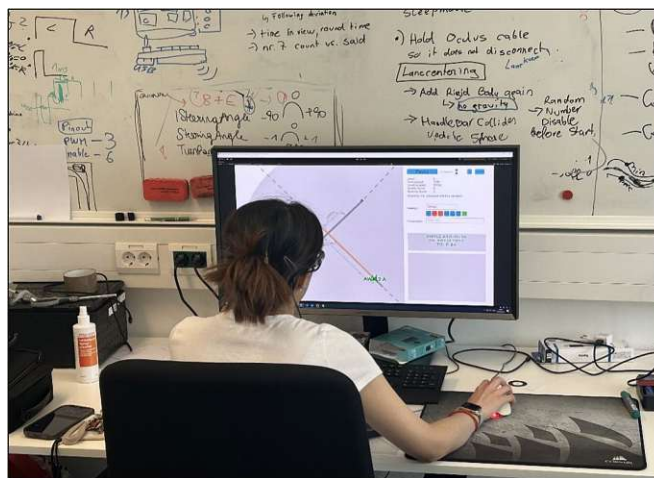


Figure 4.1: One NDK (student) participant in a room at Vienna University of Technology.



Figure 4.2: One DK (pilot) participant in the room of the flying school in Graz.

4.2 Results

4.2.1 Missing Data

The results of the participant *NDK02* from the *no domain knowledge group* has been excluded from the study analysis, as due to an error of the study assistant, the wrong order of levels, that did not match the study protocol, has been selected.

4.2.2 Descriptive Statistics

Performance Scores

Table 4.2 shows the averaged results of the performance evaluation for every single study participant.

Table 4.2: Study participant's performance scores.

ID	All Levels Avg. Quality Score	All Levels Avg. Quantity Score	All Levels Avg. Successful Parkings	All Levels Avg. Successful Takeoffs	All Levels Avg. Created Collision Hazards	All Levels Avg. Successful Tasks	All Levels Avg. Percentage Successful Tasks
01	126.33	11.00	3.00	2.33	2.67	5.33	0.38
03	324.67	13.33	4.33	3.33	1.33	7.67	0.48
04	228.33	11.33	4.33	1.00	2.67	5.33	0.37
05	-182.00	8.67	2.00	1.00	2.00	3.00	0.20
06	112.67	11.00	4.00	1.33	0.67	5.33	0.36
D01	356.00	13.67	5.33	2.33	0.67	7.67	0.52
D02	37.00	11.67	4.33	1.67	0.00	6.00	0.40
D03	35.33	10.00	3.00	1.33	0.67	4.33	0.33
D04	-156.00	10.33	3.33	0.33	2.67	3.67	0.24
D05	-178.67	6.67	1.67	0.67	0.00	2.33	0.17
D06	226.67	11.67	4.00	1.33	0.00	5.33	0.36
D07	180.67	10.67	3.33	1.33	0.67	4.67	0.32
Avg. NDK	122.00	11.07	3.53	1.80	1.87	5.33	0.36
Avg. DK	71.57	10.67	3.57	1.29	0.67	4.86	0.33

NASA-TLX Questionnaire

Table 4.4 shows the averaged results of the self-assessed NASA TLX questionnaire and Table 4.5 the results at the highest level 5. Table 4.3 shows the conversions scale that has been used to convert the options given in the survey, to NASA-TLX scale values.

Table 4.3: The conversion scale that has been used to convert the options given in the survey, to NASA-TLX scale values.

Value in Survey	NASA TLX Number Value
Very low	0
Moderately low	16.66
Slightly low	33.32
Neutral	50
Slightly high	66.66
Moderately high	83.33
Very high	100

Table 4.4: The averaged results of the NASA-TLX survey.

ID	All Levels Avg. Mental Demand	All Levels Avg. Physical Demand	All Levels Avg. Temporal Demand	All Levels Avg. Effort	All Levels Avg. Frustration	All Levels Avg. Performance
NDK01	55.55	44.43	55.55	55.55	38.88	72.22
NDK03	50.00	27.77	44.44	44.44	55.55	55.55
NDK04	61.11	16.66	55.55	77.77	61.11	33.33
NDK05	72.22	55.55	66.66	77.77	83.33	16.66
NDK06	61.10	22.21	61.10	61.10	66.67	27.77
DK01	66.66	38.89	61.10	61.10	61.11	44.44
DK02	66.66	50.00	61.11	66.66	66.66	44.44
DK03	49.99	72.22	55.55	66.67	77.77	38.89
DK04	55.55	44.43	66.66	66.66	50.00	49.99
DK05	61.11	61.11	61.11	77.77	55.55	38.88
DK06	72.22	61.10	72.22	72.22	77.77	61.11
DK07	88.89	83.33	83.33	88.89	83.33	16.66
Avg. NDK	60.00	33.33	56.66	63.33	61.11	41.10
Avg. DK	65.87	58.73	65.87	71.43	67.46	42.06

Table 4.5: The results of the NASA-TLX survey for the highest level 5.

ID	Level 5 Mental Demand	Level 5 Physical Demand	Level 5 Temporal Demand	Level 5 Effort	Level 5 Frustration	Level 5 Performance
NDK01	66.66	66.66	83.33	83.33	50.00	50.00
NDK03	83.33	50.00	83.33	83.33	100.00	66.66
NDK04	100.00	16.66	100.00	100.00	100.00	0.00
NDK05	83.33	66.66	83.33	83.33	100.00	0.00
NDK06	83.33	33.32	83.33	83.33	100.00	16.66
DK01	83.33	50.00	83.33	83.33	100.00	16.66
DK02	83.33	50.00	83.33	83.33	83.33	16.66
DK03	66.66	100.00	83.33	100.00	83.33	16.66
DK04	83.33	66.66	100.00	100.00	83.33	16.66
DK05	66.66	66.66	83.33	83.33	66.66	50.00
DK06	83.33	83.33	83.33	100.00	100.00	16.66
DK07	100.00	100.00	100.00	100.00	100.00	0.00
Avg. NDK	83.33	46.66	86.66	86.66	90.00	26.66
Avg. DK	80.95	73.81	88.09	92.86	88.09	19.04

Oral Interview

Immediately after the participants played the pre-selected levels of the game, a verbal interview has been conducted with each of them. No information about the performance within the simulation has been given to the participants ahead of the interview. The following paragraphs list the questions that have been asked with a summary of the answers given. The summaries have been generated by a Large Language Model. The model was fed by the transcribed questions and answers of the participants. The prompt "Summarize the answers for the question [question]" was given to the Large Language Model *GPT-5* on 2025-11-03. The output was cross-checked with the original interview answers for plausibility by the author.

How did you feel playing the game? Participants' feelings while playing the game ranged from enjoyment to frustration. Many described the experience as **fun, interesting, and engaging**, appreciating the challenge and the realistic simulation of managing aircraft. However, as the game progressed, several participants reported feeling **stressed, confused, or overwhelmed**, especially in the final levels where task complexity increased. Common frustrations included **difficult controls, small clickable areas, and unclear differences between taxi and takeoff commands**. While some players found the rising difficulty **motivating and satisfying**, others felt hindered by technical issues or lack of practice. Overall, most participants enjoyed the concept and challenge of the game but found it **mentally demanding and occasionally stressful**, particularly toward the end.

How stressed or challenged did you feel during the game? Participants generally reported that their **stress and challenge levels increased with each round of the game**. The first level was described as **easy, calm, and manageable**, while the later levels became **progressively more demanding and fast-paced**. Many participants felt **stressed, confused, or overwhelmed** as the number of planes increased and the game required faster reactions. Common causes of stress included **control issues**, such as difficulty clicking on small targets or confusing the taxi and takeoff commands, as well as the **pressure of multitasking** under time constraints. Some players described moments of frustration or loss of control, while others viewed the challenge as **stimulating and enjoyable**. Overall, stress levels ranged from **low to moderate in the beginning to high and cognitively demanding in later rounds**, with players' experiences shaped by both their familiarity with gaming and their ability to handle multitasking pressure.

How would you assess stress/challenge during the different levels? Participants consistently reported that the **difficulty and stress increased with each level**. The **first level** was widely described as **easy, relaxed, and manageable**, serving as a warm-up phase. The **second level** introduced a **moderate challenge**, with some participants noting the first signs of stress or confusion due to increased traffic and the need for faster reactions. The **final level** was almost universally seen as **the most difficult**

and stressful, often described as overwhelming or chaotic, with players struggling to manage multiple planes simultaneously. Many attributed the higher stress to **the steep increase in complexity, limited control precision, and time pressure**. A few participants, however, found the progression motivating and viewed the rising challenge as **engaging and rewarding**. Overall, stress levels followed a clear upward trend, from **low in the first level to high in the last**, reflecting the game’s escalating cognitive and multitasking demands.

Is there something you specifically like about the game? Most participants highlighted that they **enjoyed the overall concept and design of the game**, particularly its **realistic simulation** of airport operations and the **strategic challenge** it offered. Many appreciated the **management and planning aspects**, such as organizing aircraft movements, coordinating takeoffs and landings, and keeping traffic under control. Several players described the game as **mentally stimulating and satisfying**, especially when they successfully avoided collisions or completed tasks efficiently. A few mentioned enjoying the **visual clarity and simplicity of the interface**, which made it easy to follow aircraft routes. Others valued the game for giving them **insight into the work of air traffic controllers** and allowing them to experience decision-making under pressure. Overall, participants liked the game’s **realism, structure, and cognitive challenge**, finding it both **engaging and educational**.

Is there something you specifically didn’t like? Many participants pointed out **technical and interface-related frustrations** as the main aspects they did not like about the game. The most common complaints involved **imprecise clicking, small icons, and difficulty selecting or controlling planes**, especially under time pressure. Several players also mentioned confusion between the **“taxi” and “takeoff” commands**, which led to mistakes or blocked runways. Others criticized the **lack of responsiveness** when trying to speed up or slow down planes and the **limited ability to correct errors** once they occurred. Some participants found the **controls non-intuitive** and sometimes struggled with the **mouse settings or zooming mechanics**. While most participants appreciated the game’s concept, they agreed that **technical improvements and smoother interaction** would make the experience more enjoyable and less frustrating.

How would you assess your performance? Most participants rated their performance as **average or below average**, acknowledging mistakes and moments of confusion during the more complex levels. Many felt that they **performed well in the first level**, where the tasks were simple, but struggled as the game became faster and more demanding. Several players admitted to **making errors**—such as clicking the wrong command, blocking runways, or causing collisions—due to time pressure or control difficulties. Some participants noted that with **more practice or better familiarity with the interface**, their performance would have improved significantly. A few players considered their performance **satisfactory** despite the challenges, while

others felt disappointed or frustrated by their results. Overall, participants viewed their performance as **adequate but improvable**, emphasizing that success depended largely on practice, precision, and the ability to stay calm under pressure.

Did you use any specific techniques to perform better/manage the pressure of multitasking in this game? Please tell me in more details how did you proceed! Most participants reported that they **did not use any specific or deliberate techniques** to manage performance or multitasking pressure during the game. Many explained that they relied on **basic logical organization and focus**, trying to prioritize urgent tasks or handle one plane at a time. A few mentioned using **structured approaches**, such as managing arrivals and departures sequentially, grouping similar tasks, or planning routes to avoid conflicts. Some participants described an effort to **stay calm and concentrate**, focusing on visual cues and step-by-step actions rather than multitasking instinctively. A small number attempted to **mentally map** aircraft positions or create a “pipeline” of movements to anticipate problems, but most admitted that rising stress and speed made it difficult to sustain such strategies. Overall, participants tended to act **intuitively and reactively**, with limited use of formal coping or planning methods, though several recognized that **practice and experience** would help them develop more effective multitasking strategies in future sessions.

Do you think you had enough time to practice before the start of levels? Most participants felt that they **did not have enough time to practice before starting the levels**. Many mentioned that a **longer introduction or additional trial rounds** would have helped them better understand the controls, commands, and overall logic of the game. However, a small number of participants—mostly those with gaming or aviation experience—said the provided time was sufficient and that learning through mistakes was part of the challenge. Overall, the majority agreed that **more practice would have improved their performance and reduced early frustration**.

Would you consider yourself good at multitasking in your everyday life? Participants’ self-assessments of multitasking ability varied, but most described themselves as **moderately good or average** at handling multiple tasks. Several participants said they **regularly multitask in daily life**, such as cooking while talking on the phone, managing several work-related tasks, or driving while conversing. However, many emphasized that their multitasking usually involves **switching quickly between tasks rather than performing them simultaneously**, describing themselves more as “task switchers” than true multitaskers. A few participants, especially those with **professional experience in demanding or high-responsibility roles** (e.g., pilots, managers), rated themselves as confident multitaskers. In contrast, others admitted they **struggle with managing several complex tasks at once** and prefer to focus on one thing until it is completed. Overall, most participants viewed their multitasking ability as **functional but limited**, depending on the nature and complexity of the tasks rather than on pure ability.

Do you think the game fairly simulates multitasking experience? Most participants agreed that the game **fairly simulated a multitasking experience**, as it required them to **manage several simultaneous actions, monitor multiple aircraft, and make quick decisions under time pressure**. Many noted that the need to switch attention between planes, avoid collisions, and coordinate arrivals and departures closely mirrored real multitasking situations. Some described the simulation as **realistic and cognitively demanding**, reflecting the kind of divided attention required in complex real-world tasks. A few participants, however, pointed out that the game involved **similar types of actions**, which made it feel more like **rapid task repetition** than true multitasking involving distinct tasks. Overall, participants agreed that the game provided a **good approximation of multitasking**, effectively testing their ability to manage competing demands, prioritize actions, and stay focused under pressure.

Do you think the game fairly assesses your multitasking skills? Most participants believed that the game **partly or fairly assessed their multitasking skills**, as it required constant attention, coordination, and quick decision-making. They felt that managing several aircraft simultaneously provided a good indication of their ability to **handle multiple tasks under pressure**. Some participants mentioned that the game measured **reaction speed and focus** rather than multitasking in a broader sense. A few players with aviation or gaming experience agreed that the game reflected multitasking demands realistically but still found the difficulty jumping in between levels too steep for fair evaluation. Overall, participants considered the game a **reasonable but imperfect measure** of multitasking, providing useful insights into attention management but influenced by interface challenges and learning curve effects.

Is there anything you'd like to add about your experience? In their final reflections, participants generally described the experience as **interesting, educational, and enjoyable**, despite some challenges. Many appreciated the opportunity to **experience a realistic simulation** of air traffic control and noted that it gave them a better understanding of the **complexity and pressure** involved in such tasks. Several participants reiterated that **technical improvements**—such as more precise clicking, smoother controls, and clearer instructions—would significantly enhance the game play. Some expressed that **additional practice time** or a short tutorial would help reduce confusion and improve confidence in managing tasks. A few mentioned that the game demanded **high concentration and attention management**, which they found mentally stimulating but also tiring. Participants with aviation backgrounds commented positively on the **realism and authenticity** of the simulation, while non-gamers found it challenging but insightful. Overall, most participants concluded that the game was a **valuable, thought-provoking, and cognitively engaging experience**, though somewhat limited by usability issues.

How many flight hours did you do? (Question for DK group only) Participants' reported flight experience varied widely, reflecting a mix of professional pilots and trainee

pilots. Among the pilot group, flight hours ranged from approximately 80 to over 2,000 hours, with most falling between 300 and 900 hours.

Are the scenarios depicted in the game realistic? (Question for DK group only) Most participants agreed that the scenarios depicted in the game were generally realistic, especially in terms of airport layout, aircraft movements, and basic air traffic control procedures. Pilots and aviation-trained participants noted that the simulation accurately represented taxiing, holding positions, and runway operations, reflecting real-world logic and sequencing.

4.2.3 Inferential Statistics

Inferential statistics have been calculated and graphs generated by the software *IBM SPSS*.

Performance Scores

Domain Knowledge vs. Performance A Mann-Whitney U test was performed to determine if there were differences in *All Levels Avg. Quality Score* between participants *with domain knowledge, DK group* and *without domain knowledge, NDK group*. The distributions of the scores for both groups were similar, as determined by visual inspection. Significance level was *0.05*. Median scores were not statistically significantly different between the groups, $N = 12$, $U = 15$, $z = 0$, $p = 0.755$, using an exact sampling distribution for U [DB73].

A Mann-Whitney U test was performed to determine if there were differences in *All Levels Avg. Quantity Score* between participants *with domain knowledge* and *without domain knowledge*. The distributions of the scores for both groups were similar, as determined by visual inspection. Significance level was *0.05*. Median scores were not statistically significantly different between the groups, $N = 12$, $U = 16$, $z = 0$, $p = 0.876$, using an exact sampling distribution for U [DB73].

Age vs. Performance A Pearson product-moment correlation was performed to determine the relationship between participant's *age* and *All Levels Avg. Quality Score*. The relationship between the two variables was visually assessed as linear and both variables are normally distributed, which a Shapiro-Wilk's test $p > .05$ confirmed. One outlier (*age = 35*, could be observed and was kept in the analysis. Significance level was *0.05*. The correlation showed that there was a negative, non-statistically significant correlation between *age* and *All Levels Avg. Quality Score*, $N = 12$, $r(98) = -0.524$, $p = 0.080$ [BCHC09].

A Pearson product-moment correlation was performed to determine the relationship between participant's *age* and *All Levels Avg. QuantityScore*. The relationship between the two variables was visually assessed as linear and both variables are normally distributed,

which a Shapiro-Wilk's test $p > .05$ confirmed. No outliers could be observed. Significance level was 0.05 . The correlation showed that there was a negative, non-statistically significant correlation between *age* and *All Levels Avg. Quality Score*, $N = 12$, $r(98) = -0.537$, $p = 0.072$ [BCHC09].

Self-Assessed Computer Gaming Skills vs. Performance Participants were asked to self-assess their computer gaming skills in a questionnaire with values ranging from [1 is "Don't play at all" and "7" is "Professional level"].

An Independent-Samples Kruskal-Wallis test was performed to determine if there were differences in *All Levels Avg. Quality Score* between participants with different *self-assessed computer gaming skills*. The distributions of the scores were not similar for all groups, as determined by visual inspection. Significance level was 0.05 . The mean ranks of *All Levels Avg. Quality Score* were not statistically significantly different between age groups, $N = 12$, $X^2 = 7.827$, *Degree of Freedom* = 5, $p = 0.166$ [MN10].

An Independent-Samples Kruskal-Wallis test was performed to determine if there were differences in *All Levels Avg. Quantity Score* between participants with different *self-assessed computer gaming skills*. The distributions of the scores were not similar for all groups, as determined by visual inspection. Significance level was 0.05 . The mean ranks of *All Levels Avg. Quantity Score* were not statistically significantly different between age groups, $N = 12$, $X^2 = 7.616$, *Degree of Freedom* = 5, $p = 0.179$ [MN10].

NASA-TLX Questionnaire

All Levels Avg. Quality Score vs Participant's TLX Ratings A Pearson product-moment correlation was performed to determine the relationship between participant's *All Levels Avg. Quality Score* vs. their self-assessed 6 NASA-TLX ratings. The relationship between *All Levels Avg. Quality Score* and the separate NASA-TLX ratings were visually assessed as linear and all variables are normally distributed, which a Shapiro-Wilk's test $p > .05$ showed. Outliers were identified and were kept in the analysis [BCHC09]. Significance level was 0.05 . The correlation showed that there were positive, non-significant correlations between *All Levels Avg. Quality Score* and *All Levels Avg. TLX Mental Demand*, *All Levels Avg. TLX Performance*. The correlation showed also that there were negative, non-significant correlations between *All Levels Avg. Quality Score* and *All Levels Avg. TLX Physical Demand*, *All Levels Avg. TLX Temporal Demand*, *All Levels Avg. TLX Effort*, *All Levels Avg. TLX Frustration* [BCHC09]. See Table 4.6 for more details.

Table 4.6: Pearson correlations between All Levels Avg. Quality Score and NASA-TLX ratings.

		AllLevels Avg. TLX Mental Demand	AllLevels Avg. TLX PhysicalD emand	AllLevels Avg. TLX Temporal Demand	AllLevels Avg. TLX Effort	AllLevels Avg. TLX Frustration	AllLevels Avg. TLX Per- formance
	Pearson Correlation	0.057	-0.324	-0.197	-0.366	-0.029	0.275
AllLevels Avg. Quality Score	Sig. (2-tailed)	0.860	0.304	0.539	0.242	0.929	0.387
	N	12	12	12	12	12	12

All Levels Avg. Quantity Score vs Participant's TLX Ratings A Pearson product-moment correlation was performed to determine the relationship between participant's *All Levels Avg. Quantity Score* vs. their self-assessed 6 NASA-TLX ratings. The relationship between *All Levels Avg. Quantity Score* and the separate NASA-TLX ratings were visually assessed as linear and all variables are normally distributed, which a Shapiro-Wilk's test $p > .05$ showed. Outliers were identified and were kept in the analysis [BCHC09]. Significance level was 0.05 . The correlation showed that there were positive, non-significant correlations between *All Levels Avg. Quantity Score* and *All Levels Avg. TLX Performance*. The correlation showed also that there were negative, non-significant correlations between *All Levels Avg. Quantity Score* and *All Levels Avg. TLX Mental Demand*, *All Levels Avg. TLX Physical Demand*, *All Levels Avg. TLX Temporal Demand*, *All Levels Avg. TLX Effort*, *All Levels Avg. TLX Frustration* [BCHC09]. See Table 4.7 for more details.

Table 4.7: Pearson correlations between All Levels Avg. Quantity Score and NASA-TLX ratings.

		AllLevels Avg. TLX Mental Demand	AllLevels Avg. TLX Physical Demand	AllLevels Avg. TLX Temporal Demand	AllLevels Avg. TLX Effort	AllLevels Avg. TLX Frustration	AllLevels Avg. TLX Per- formance
AllLevels Avg. Quantity Score	Pearson Correlation	-0.069	-0.433	-0.248	-0.561	-0.122	0.367
	Sig. (2-tailed)	0.831	0.159	0.436	0.058	0.705	0.241
	N	12	12	12	12	12	12

4.3 Interpretations

4.3.1 Participant's Performance

Performance between participants varied greatly in terms of *Avg. Quality Score* and *Avg. Quantity Score*. Analysis showed that the higher the level, the lower the percentage of accomplished tasks in both groups. Although some participants mentioned in the oral interview, that the duration of the introduction session was, with 15 minutes quite short, all of them managed to complete at least some *aircraft parking* and *aircraft takeoff* tasks. This gives us the conclusion, that the protocol for the introductory session is sufficient and the session long enough to allow participants to perform basic tasks in the simulation. There was no participant, that did not accomplish any task. All participants performed the whole study according to the protocol, without any request to cancel earlier. No extraordinary or unexpected events took place. *NDK (no domain knowledge group)* performed better than *DK (domain knowledge group)* in quality and slightly better in quantity of tasks. The higher cognitive performance due to young average age (almost half of average age than DK group) of the *NDK group* might be a contributing factor to the good performance. However, *NDK* caused more collisions over all levels. *DK (pilots)* prioritized the avoiding of accidents (task prioritization, risk-awareness and accident avoiding are important traits of pilots). Possibly, pilots scored less in other scores in favor of avoiding accidents (i.e. less throughput but also less accidents). The Mann-Whitney U Test showed *no statistically significant difference in quality and quantity scores between the NDK and DK group*. When it comes to the question of whether there is a correlation between the participant's age and performance, the Pearson product-moment correlation showed that there is a *negative, non-statistically significant correlation between age and performance* giving a hint that higher age might influence the performance towards lower values. We also performed an Independent-Samples Kruskal-Wallis Test to determine whether there is a relation between the participant's *Self-Assessed Computer Gaming Skills* vs. their *performance* in the simulation. However, there was no statistically significant finding supporting a possible relation.

4.3.2 NASA-TLX Questionnaire

When it comes to the NASA TLX questionnaire, all participants had to fill right after playing the simulation, we saw that the simulation was capable to induce high to very high mental demands in both NDK and DK groups. NDK (no domain knowledge group) perceived lower demand in almost all categories of the NASA TLX in all simulation levels and scored better in the simulation. DK (domain knowledge group) perceived higher demand in almost all categories of the NASA TLX in all simulation levels and scored worse in the simulation. NDK group perceived much lower average physical demand than DK, which might stem from the much lower average age of this group (almost half of average age than DK group). Average overall experience was assessed neutral by NDK, high (good) by DK. This might be the result of the fact that the DK group is familiar with the depicted scenarios and affine to the aviation topic as such. The Pearson

product-moment correlation showed negative, non-significant correlations between *All Levels Avg. Quality Score* and *All Levels Avg. TLX Physical Demand, All Levels Avg. TLX Temporal Demand, All Levels Avg. TLX Effort, All Levels Avg. TLX Frustration*, leading to the interpretation that the NASA TLX results reflect the measured, objective simulation performance in terms of *quality* quite well. The Pearson product-moment correlation showed positive, non-significant correlations between *All Levels Avg. Quantity Score* and *All Levels Avg. TLX Performance*, indicating that the participants were able to assess their performance adequately. The correlation showed also that there were negative, non-significant correlations between *All Levels Avg. Quantity Score* and *All Levels Avg. TLX Mental Demand, All Levels Avg. TLX Physical Demand, All Levels Avg. TLX Temporal Demand, All Levels Avg. TLX Effort, All Levels Avg. TLX Frustration*, indicating that participants that scored better in terms of *quantity, i.e. task completion speed*, experienced lower *mental, physical, temporal load* and lower *effort and frustration*.

After all we conclude, that the performance measured by the simulation environment in terms of quality and quantity of task accomplishment, reflects the different perceived loads, effort and frustration well. The simulation environment was furthermore capable of inducing high to very high mental and temporal loads at least at the highest played level. Additionally, we got the impression, that the outcome of the NASA TLX survey is plausible.

4.3.3 Oral Interview

Most participants enjoyed the concept and challenge of the game but found it mentally demanding and occasionally stressful, especially at the higher levels. Participants described that the perceived stress increased with increasing level. The lower levels were perceived mostly as "easy, calm and manageable", while at the higher levels some players answered that they feel "stressed, confused and overwhelmed". The latter also aligns with the high perceived mental and temporal loads as indicated by the self-assessed NASA TLX questionnaire. When it comes to multitasking experience, most participants mentioned that the game "fairly simulates multitasking experience", as it required them to "manage several simultaneous actions, monitor multiple aircraft, and make quick decisions under time pressure". Most participants "enjoyed the overall concept and design of the game", particularly its "realistic simulation" of airport operations and the "strategic challenge" it offered. Furthermore, they answered that they find the game "engaging and educational". When it comes to issues that were disliked, it was mentioned several times that there was "imprecise clicking, small icons, and difficulty selecting or controlling planes, especially under time pressure". We took this feedback seriously and improved the function of selecting planes on the map. It should be now easier to perform selections in different zoom levels of the maps. Assessing their own performance in the game, many participants felt that they would most likely perform better if they had the chance to practice more intensively before getting assessed. Just very few participants used specific cognitive techniques to manage their stress levels and perform better in the game. Most participants acted "intuitively and reactively". Participants in the DK group were also

asked whether the scenarios depicted in the game are realistic. There was a consensus that "the scenarios depicted in the game were generally realistic, especially in terms of airport layout, aircraft movements, and basic air traffic control procedures". After all, the answers given in the oral interviews match the answers given in the online NASA TLX questionnaire.

The interview answers have been summarized by a Large Language Model, see more in section 4.2.2. The citations in the above paragraph refer to these summaries.

4.4 Conclusion

After the Problem Statement has been defined and the Research Questions for this thesis have been outlined, the Literature Research helped us to collect the functional and non-functional requirements for a simulation environment for multitasking research. We categorized the functional requirements between requirements targeting the human-machine interface (HMI) with the radar display and simulation control panels, the simulation scenarios to be covered and the simulation control and execution features. Furthermore, we put a focus on those requirements that are necessary to allow multitasking performance measurement within the simulation environment. The literature research revealed numerous publications targeting requirements for such an environment. The most relevant one have been selected and incorporated into the implemented software. For the non-functional requirements we collected demands targeting performance, reliability and security of the software, as well as cost efficiency considerations. In the following step, a concept for the simulation environment, fulfilling all the requirements from the above-mentioned categories, has been created and discussed. We choose the simulation and game development platform *Unity* as a suitable way to implement the concept. After all, *Unity* turned out to be a high-performance and efficient way to implement such types of simulation environments. During the implementation, we ought to use state-of-the art design principles and conventions. As a high-level design principle, the Model-View-Controller architectural principle was applied. Due to the use of code conventions, XML documentation tags and source documentation, we ensured, that the simulation environment can be altered or changed as needed by future research work. A main characteristic of the created simulation environment, is its extensibility and configurability. After having learned the internal concepts, it is straight forward, to integrate new airports and scenarios. The use of thoroughly defined data structures, modularization, interfaces and inheritance allows to do so easily without having to interfere with the simulation environment's core code. After the implementation has been completed and tested, a User Study and Evaluation has been performed, with two groups of participants. One with members having no aviation domain knowledge, and the other with members having it. We wanted to test whether the created simulation environment fulfills the goals set out in the Problem Statement under consideration of the Expected Results and Success Criteria. The user study showed that for both groups it was possible to obtain solid performance in task completion after introducing the users to the simulation for 15 minutes, and thus the availability of the simulation environment

to be suitable for multitasking research. Performances of the different participants varied greatly, with no significant difference between the two groups mentioned above. However, we found correlations between the measured performance of the participants in the simulation environment and their grades in the self-assessed NASA TLX. At the highest levels, the challenges in the simulation were able to create high to very high self-assessed mental and temporal demand of all participants, regardless of their group they were assigned to. Feedback given by the study participants to certain aspects of the HMI resulted in implemented improvements of the software.

4.5 Future and Related Work

The results of the user study showed, that the more tasks the participants have to accomplish in parallel (i.e. the higher the played level in the simulation environment was), the worse the performance got. Similar observation has also been made by Li et al. [LLXC22] who did performance measuring of participants in a simulated plane. Also there the "performance was worse during the high mental workload multitasking condition" compared to "to multitasking in lower mental workload condition". Also Lin et al. [LCCN16] found "significantly lower accuracy and longer completion time in participants when they attempted the combined tasks, suggesting that even concurrent tasks that depend upon different modalities may hinder efficiency". The analysis of the self-assessed NASA TLX questionnaire showed, that the higher the multitasking requirement of the simulation scenario was, the higher the participant's reported scores in the dimensions *mental load* and *temporal demand* were. This observation has also been made by Lin et al. [LCCN16]. When it comes to age as an independent variable for multitasking performance, our results showed, that there might be a negative correlation between these two. Paridon and Kaufmann [PK10] analyzed multitasking studies that deal with influence of age in task speed and precision and came to the same conclusion. Crews and Russ [CR20] concluded "Younger and older participants took more time to complete the experiment, which indicates an inverted U relationship and may mean that people become more effective at multitasking with experience and practice, but then less effective as they get older." Here it might be of future interest, how strong the independent variables *experience* and *age* correlate with multitasking performance and to which extend missing experience can be made up by a younger age. Our study showed, that the significantly older group of participants with domain knowledge, performed worse than the younger group without. Our work did not involve analysis on how certain aspects of the simulation environment's HMI might influence the performance of players. Several work, e.g. the one by Chang and Yeh [CY10] targets the influence of the design of ATC workplaces on the performance of the controllers. Although we did eye tracker recordings during the user study, further work has to be done to analyze them and potentially find possible improvements when it comes to the implemented HMI.

4.6 Limitations

There was a strong requirement for the way reality in ATC should be simulated in the environment to be implemented: For the sake of doing multitasking research efficiently, it has been defined, that users without knowledge in the ATC domain must be successfully trained to execute basic tasks within 15 minutes. This created the need to somewhat simplify certain aspects of the simulation like scenarios in a way, that such a requirement can be fulfilled. However, it was taken care, that the main aspects of cognitive work, that is required in ATC, remain realistic. This fact was also confirmed during the user study, where also users with domain knowledge were tested and interviewed, although some of them gave as a feedback that they would have benefit from a more extensive introduction session into the simulation environment. Allowing participants to familiarize themselves with the simulation for a longer time, might be considered in future work. Another limitation exists when it comes to communication between the simulated ATC workplace and the simulated planes: While in reality, ATC is using voice (radio) *and* textual commands (CPDLC), in the created simulation, textual commands are given exclusively. This might induce in some aspects, differences between real-world and the simulation. Implementing voice commands by using TTS (text-to-speech) technology in future, might close this gap. However, it has to be considered, that real-world ATC uses extensive standard phraseology to give voice commands to planes. For us it seems to be unrealistic, to have the study participants learn these commands. So, if the simulation environment should continue fulfilling the requirement of being understood fast by participants without domain knowledge, the introduction of voice commands seems to be contrarious. A limitation of the user study was the relatively low number of participants (n=12) and the large difference in average age in the NDK group (31.4 years) and DK group (59 years). Due to limited resources, the participants could not be selected in a way to fulfill broader requirements in terms of difference in participant variables. Especially, participants with domain knowledge of aviation, were a challenge for us to find. For future studies, the group of participants should be considerably larger, and the recruiting process should take care, that the samples are representative and reduce bias. Additionally, while the hardware setup was the same for all participants in the study, the environment (location) was different. We asked the participants, whether they have used certain cognitive techniques to improve their performance in the simulation. Almost all of them answered, that they haven't. It might be of interest in future, how the application of such strategies might influence the performance in the simulation. To conclude, we can say, that potential for future user studies in the area this thesis covers is recommended. However, it also has to be mentioned, that the focus of this thesis was to create the concept and implementation of the simulation environment.

Übersicht verwendeter Hilfsmittel

Im Abschnitt section 4.2.2 wurde das Large Language Model *GPT-5* am 2025-11-03 benutzt, um die Transkripte der mündlichen Interviews zusammenzufassen. Der folgende Prompt wurde verwendet: “Fasse den Text zusammen”.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

3.1	HMI of ATChallenge	18
3.2	Zooming and dragging the radar display in ATChallenge	18
3.3	Pop-up displaying level instructions.	19
3.4	Arrival and departure routes of the <i>GenericAirport1</i>	20
3.5	Directions of arriving and departing planes in ATChallenge	21
3.6	An arriving plane is marked by the letter A, following its call sign.	22
3.7	The information panel for an arriving plane.	23
3.8	Plane information panel in detail.	23
3.9	A route on the taxiway network of the sample airport <i>GenericAirport1</i>	24
3.10	Buttons to send commands to planes.	25
3.11	A plane ready at its parking position to depart.	26
3.12	Plane information of a departing plane, showing assigned runway (RWY-04) and intersection (TWY-D) to depart.	26
3.13	Plane ready for departure at the holding point on TWY-D to the assigned runway (RWY-04).	27
3.14	Simulation control elements on the GUI.	28
3.15	Documentation page created by Doxygen	32
3.16	<i>TrafficPositions</i> (green) forming a <i>MovingPath</i> (gray dotted) which is representing a Standard instrument departure (SID).	33
3.17	Naming of areas, routes and points on airports.	37
3.18	Base structure for a new airport in the Unity Inspector.	37
4.1	One NDK (student) participant in a room at Vienna University of Technology.	53
4.2	One DK (pilot) participant in the room of the flying school in Graz.	53

List of Tables

2.2	Functional requirements for an ATC simulation environment for multitasking research	13
3.1	Categorization of classes according to MVC	30
3.2	Configuration options for the simulation environment	46
4.1	Study participant’s demographics and self-assessed computer gaming experience.	51
4.2	Study participant’s performance scores.	55
4.3	The conversion scale that has been used to convert the options given in the survey, to NASA-TLX scale values.	56
4.4	The averaged results of the NASA-TLX survey.	57
4.5	The results of the NASA-TLX survey for the highest level 5.	58
4.6	Pearson correlations between All Levels Avg. Quality Score and NASA-TLX ratings.	65
4.7	Pearson correlations between All Levels Avg. Quantity Score and NASA-TLX ratings.	67

Glossary

Air Traffic Management Unit which is ensuring that flights travel via predefined efficient and secure procedures.. 12

Airway Horizontally and vertically defined routes for planes to fly on. Are continuously replaced in favor of free route airspace, meaning planes are not bound to confined airways anymore, making traffic more efficient. . 13

Altitude The actual altitude measured by plane on-board equipment and transmitted via the on-board radar transponder to the radar station.. 13

Apron Movement area on an airport, that contains one or more parking positions.. 22

ATC Sector An area of responsibility of a single ATC controller [Kal03].. 9, 10, 13, 22

Automatic Dependent Surveillance - Broadcast System that actively broadcasts plane flight data like altitude, speed, heading, autopilot settings to other planes or ATC units.. 2, 10, 12

Call sign A unique alphanumerical identifier for a plane. It can, but must not, match the flight number or the registration of the plane.. 13, 18, 22, 25, 75

Clearance A dedicated instruction to a plane to fly in a horizontally or vertically defined path or area.. 13

Controller Pilot Data Link Communications Systems that allow to exchange text commands unidirectional between ATC units and pilots. This imposes an alternative to traditional voice communications.. 2, 10, 12

Groundspeed The actual speed relative to the ground, the plane is moving.. 13

Holding point A special point and marking on a taxiway, that must not be crossed by a plane without special permission.. 25, 33

ICAO International Civil Aviation Organization, a part of the United Nations.. 22

Large Language Model A model that is used in the artificial knowledge domain to solve certain tasks. . 59, 70, 73

Model-View-Controller Architectural principle that splits system modules according their function in terms of data model, view and controlling. . 70

NASA TLX A multidimensional assessment questionnaire to establish perceived values in the cognitive domain. Among others there are perceived cognitive demand and effort. . 5, 56, 68, 69, 70, 71

Navigational Aid Terrestrial stations sending out radio signals used by planes to determine their position.. 13

n-back A psychological test, where "subjects are asked to monitor series of briefly presented stimuli and have to decide in each trial if the currently presented stimuli is the same as the one presented two or three trials before" [SPW08].. 1

Radar Echo The reflections of the emitted radar beam by objects in the air or on the ground.. 10, 13

Squawk A 4-digit number that can be selected on the plane's on-board radar transponder and will be transmitted back to the radar station. It is used to identify and transmit additional information from the plane to the radar station.. 13

Standard Instrument Departure (SID) Horizontally and vertically defined routes for planes to fly after they departed from a specific runway.. 13, 17, 20, 33, 38, 75

Standard Arrival Route (STAR) Horizontally and vertically defined routes for planes to approach a specific runway.. 13, 17, 20, 38

Taxi The movement of planes on or slightly above (in case of helicopters) a maneuvering area of an airport.. 13, 20

Taxiway Movement areas on an airport, where planes can taxi on or slightly above (in case of helicopters) from one part of the airport to another.. 13

Track The actual direction measured in degrees of the compass rose, the plane is moving.. 13

Tracking Task A psychological test, where the subjects have to track an object displayed on a computer screen via hand-eye coordination.. 1

VCR Task Visual Common Reasoning. A psychological test, where subjects try to recognize with one glance at an image,depicted people's actions, goals, and mental states [ZBFC19].. 1

WYSIWYG What you see, is what you get.. 32

Acronyms

ATC Air Traffic Control. ix, 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 14, 15, 17, 20, 28, 29, 51, 52, 71, 72

EUROCONTROL European Organisation for the Safety of Air Navigation. 11, 12

HMI Human-Machine Interface. 2, 9, 10, 11, 14, 70, 71

SVG Scalable Vector Graphics. 29, 37



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

- [ABF15] Rachel F Adler and Raquel Benbunan-Fich. The effects of task difficulty and multitasking on performance. *Interacting with Computers*, 27(4):430–439, 2015.
- [Ala23] Moses Alabi. Data governance and quality: Ensuring data reliability and trustworthiness. 2023.
- [Ant20] D. Antolović. Development of scenarios on the escape atc simulator, 2020.
- [AOM⁺14] Busyairah Syd Ali, Washington Ochieng, Arnab Majumdar, Wolfgang Schuster, and Thiam Kian Chiew. Ads-b system failure modes and models. *The Journal of Navigation*, 67(6):995–1017, 2014.
- [ARS⁺01] Ulf Ahlstrom, Joshua Rubinstein, Steven Siegel, Richard Mogford, Carol Manning, et al. Display concepts for en route air traffic control. Technical report, William J. Hughes Technical Center (US), 2001.
- [BCHC09] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.
- [BFAM11] Raquel Benbunan-Fich, Rachel F Adler, and Tamilla Mavlanova. Measuring multitasking behavior with activity-based metrics. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 18(2):1–22, 2011.
- [BKPK06] Markus Bühner, Cornelius J König, Marion Pick, and Stefan Krumm. Working memory dimensions as differential predictors of the speed and error aspect of multitasking performance. *Human Performance*, 19(3):253–275, 2006.
- [BM97] Walter C. Borman and Stephan J. Motowidlo. Task performance and contextual performance: The meaning for personnel selection research. *Human Performance*, (10:2), 1997.
- [BWS96] Jeffrey B. Brookings, Glenn F. Wilson, and Carolyne R. Swain. Psychophysiological responses to changes in workload during simulated air traffic control. *Biological Psychology*, 42(3):361–377, 1996.

- [CACK⁺19] Cem Cetek, Fulya Aybek Cetek, Zekeriya Kaplan, Kadir Dönmez, Biljana Juricic, Bruno Antulov-Fantulin, T. Feuerle, and Paul Frost. Atcosima project: Integrated atc radar-flight deck simulations for the assessment of atco trainees. 2019.
- [Cle78] Joe K Clema. General purpose tools for system simulation. In *Proceedings of the 11th annual symposium on Simulation*, pages 37–60, 1978.
- [CNYM12] Lawrence Chung, Brian A Nixon, Eric Yu, and John Mylopoulos. *Non-functional requirements in software engineering*, volume 5. Springer Science & Business Media, 2012.
- [CR20] Derek E Crews and Molly J Russ. The impact of individual differences on multitasking ability. *International Journal of Productivity and Performance Management*, 69(6):1301–1319, 2020.
- [CVCB23] Nicola Cavagnetto, Roberto Venditti, Matteo Cocchioni, and Stefano Bonelli. Sectorx, an en-route atc simulator for ai-based decision support to air traffic controllers: A case study in the mahalo project. In *Proceedings of the 15th Biannual Conference of the Italian SIGCHI Chapter*, pages 1–3, 2023.
- [CY10] Yu-Hern Chang and Chung-Hsing Yeh. Human performance interfaces in air traffic control. *Applied Ergonomics*, 41(1):123–129, 2010.
- [DB73] LC Dinneen and BC Blakesley. Algorithm as 62: A generator for the sampling distribution of the mann-whitney u statistic. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 22(2):269–273, 1973.
- [DC02] Gordana Dodig-Crnkovic. Qualitative methods in empirical studies of software engineering. *Proceedings of the Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden*, pages 126–130, 2002.
- [FLN09] Selina Fothergill, Shayne Loft, and Andrew Neal. Atc-lab(advanced): An air traffic control simulator with realism and control. *Behavior research methods*, 41:118–27, 2009.
- [GG06] Lawrence W Green and Russell E Glasgow. Evaluating the relevance, generalization, and applicability of research: issues in external validation and translation methodology. *Evaluation & the health professions*, 29(1):126–153, 2006.
- [HE16] Jacco M Hoekstra and Joost Ellerbroek. Bluesky atc simulator project: an open data and open source approach. In *Proceedings of the 7th international conference on research in air transportation*, volume 131, page 132. FAA/Eurocontrol USA/Europe, 2016.

- [HH99] Henry Hexmoor and Tim Heng. Atc tower simulator: Tacund. 1999.
- [Hil04] Brian Hilburn. Cognitive complexity in air traffic control: A literature review. *EEC note*, 4(04):1–80, 2004.
- [HMPR04] Alan R. Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28:75–105, 2004.
- [HS88] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.
- [Ibr] Abdullah Sabah Ibrahim. *Air traffic Control Radar Design*. PhD thesis, University of Technology.
- [JVB11] Biljana Juričić, Ivana Varešak, and Diana Božić. The role of the simulation devices in air traffic controller training. In *International Symposium on Electronics in Traffic, ISEP 2011 Proceedings*, 2011.
- [Kal03] Sven Kaltenhäuser. Tower and airport simulation: flexibility as a premise for successful research. *Simulation Modelling Practice and Theory*, 11(3-4):187–196, 2003.
- [Kat95] Vamshi K Katukoori. Standardizing availability definition. *University of New Orleans, New orleans, La., USA*, 1995.
- [KC⁺07] Barbara Kitchenham, Stuart Charters, et al. Guidelines for performing systematic literature reviews in software engineering, 2007.
- [Kit07] Barbara Kitchenham. A systematic review of systematic review process research in software engineering. *Information and Software Technology*, (26:3):2049–2075, 2007.
- [KMPT97] Tak-Kuen Koo, Yi Ma, George Pappas, and Claire Tomlin. Smartatms: A simulator for air traffic management systems. pages 1199–1205, 01 1997.
- [LCCN16] Lin Lin, Deborah Cockerham, Zhengsi Chang, and Gloria Natividad. Task speed and accuracy decrease when multitasking. *Technology, knowledge and learning*, 21:307–323, 2016.
- [Lin23] Alexander Lingler. Smart agent supported task-switching optimization using reinforcement learning and human cognition modelling, 2023.
- [LLXC22] Wenbin Li, Rong Li, Xiaoping Xie, and Yaoming Chang. Evaluating mental workload during multitasking in simulated flight. *Brain and Behavior*, 12(4):e2489, 2022.

- [LR01] Avraham Leff and James T Rayfield. Web-application development using the model/view/controller design pattern. In *Proceedings fifth ieee international enterprise distributed object computing conference*, pages 118–127. IEEE, 2001.
- [Mar00] Robert C Martin. Design principles and design patterns. *Object Mentor*, 1(34):597, 2000.
- [Mar02] M. L. Markus. A design theory for systems that support emergent knowledge processes. *MIS Quarterly*, (26:3):179–212, 2002.
- [MB⁺01] Ruth Malan, Dana Bredemeyer, et al. Functional requirements and use cases. *Bredemeyer Consulting*, pages 335–1653, 2001.
- [MD00] Stephen Monsell and Jon Driver. *Control of cognitive processes: Attention and performance XVIII*, volume 18. MIT Press, 2000.
- [MN10] Patrick E McKight and Julius Najab. Kruskal-wallis test. *The corsini encyclopedia of psychology*, pages 1–1, 2010.
- [MNS⁺20] Malgorzata Marchewka, Janusz Nesterak, Mariusz Sołtysik, Wojciech Szymła, and Magdalena Wojnarowska. Multitasking effects on individual performance: an experimental eye-tracking study. 2020.
- [MO02] Arnab Majumdar and Washington Y. Ochieng. Factors affecting air traffic controller workload: Multivariate analysis based on simulation modeling of controller workload. *Transportation Research Record*, 1788(1):58–69, 2002.
- [MSJ⁺19] Mohammad Moallemi, Christopher Shannon, Shafagh Jafer, Ashok V Raja, and Neal C Thigpen. Building atc simulator through scenario-driven requirements engineering. In *AIAA Scitech 2019 Forum*, page 1482, 2019.
- [NKK⁺19] Sebastian Nielebock, Dariusz Krolikowski, Jacob Krüger, Thomas Leich, and Frank Ortmeier. Commenting source code: is it worth it for small programming tasks? *Empirical Software Engineering*, 24:1418–1457, 2019.
- [OJCL16] Hyeju Oh, Sehun Jeong, Keeyoung Choi, and Hak-Tae Lee. Human-in-the-loop simulation analysis of conflict resolution maneuvers using an air traffic control simulation. In *AIAA Modeling and Simulation Technologies Conference*, page 0169, 2016.
- [PK10] Hiltraut M Paridon and Marlen Kaufmann. Multitasking in work-related situations and its relevance for occupational health and safety: Effects on performance, subjective strain and physiological parameters. *Europe’s Journal of Psychology*, 6(4):110–124, 2010.

- [RWS⁺17] Gernot Rottermanner, Markus Wagner, Volker Settgast, Volker Grantz, Michael Iber, Ursula Kriegshaber, Wolfgang Aigner, Peter Judmaier, and Eva Eggeling. Requirements analysis & concepts for future european air traffic control systems. In *Workshop Vis in Practice-Visualization Solutions in the Wild, IEEE VIS 2017*, 2017.
- [S⁺85] Earl S Stein et al. Air traffic controller workload: An examination of workload probe. Technical report, United States. Department of Transportation. Federal Aviation Administration . . . , 1985.
- [Sea99] Carolyn B. Seaman. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, (25:4), 1999.
- [Sea08] Robert C Seacord. *The CERT C secure coding standard*. Pearson Education, 2008.
- [SHV⁺20] Vladimír Socha, Lenka Hanakova, Viktor Valenta, Luboš Socha, Richard Abela, Stanislav Kusmirek, Terezia Pilmannova, and Jan Tecl. Workload assessment of air traffic controllers. *Transportation Research Procedia*, 51:243–251, 2020.
- [SPW08] Daniela Schoofs, Diana Preuß, and Oliver T. Wolf. Psychosocial stress induces working memory impairments in an n-back paradigm. *Psychoneuroendocrinology*, 33(5):643–653, 2008.
- [SSF⁺] Kakuichi Shiomi, Hiroki Sato, Yutaka Fukuda, Kota Kageyama, Toshihiro Hiwada, Kota Kageyama, and Kazuo Shima. *Development of ATC simulation facility*.
- [STS73] RC Sohnle, J Tartar, and JR Sampson. Requirements for interactive simulation systems. *Simulation*, 20(5):145–152, 1973.
- [SVM⁺23] María Zamarreño Suárez, Rosa María Arnaldo Valdés, Francisco Pérez Moreno, Raquel Delgado-Aguilera Jurado, Patricia María López de Frutos, and Víctor Fernando Gómez Comendador. Is it possible to evaluate the event-based taskload of an air traffic controller using an air traffic control simulator? *Journal of Physics: Conference Series*, 2526(1), 2023.
- [VL09] Axel Van Lamsweerde. Reasoning about alternative requirements options. In *Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos*, pages 380–397. Springer, 2009.
- [VP14] Juraj Vagner and Edina Pappová. Comparison of radar simulator for air traffic control. *NAŠE MORE: znanstveni časopis za more i pomorstvo*, 61(1-2):S31–35, 2014.

- [ZBFC19] Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. From recognition to cognition: Visual commonsense reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.