2006-06

A teamwork-oriented air traffic control simulator

Sidhom, Mounir

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/2716

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A TEAMWORK-ORIENTED AIR TRAFFIC CONTROL SIMULATOR

by

Mounir Sidhom

June 2006

Thesis Advisor:	Arnold Buss
Second Reader:	Don McGregor

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2006	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE: A Teamwork-oriented Air Traffic Control Simulator	**5. FUNDING NUMBERS**
6. AUTHOR(S) Mounir Sidhom	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Air Traffic Control (ATC) is a complicated domain in which many specialists should collaborate and communicate with each other in order to guarantee safe and efficient air traffic. A significant number of air traffic control errors are associated with either faulty coordination between ATC actors, or a failure of some kind of team coordination. These errors are likely to increase in the future as aircraft density increases. Many researchers suggest that the introduction of team and teamwork concepts during the training phase of the ATC actors will be in help to reduce the amount of these errors.

The objective of this research is to conceive, design, and implement a teamwork-oriented Air Traffic Control simulator that can be easily installed and used in ATC schools. The product of this thesis will be a complete software package that allows trainees in the different ATC specialties to work together in the same manner as they do "on-the-job" in order to collaboratively manage an air traffic situation. This type of simulator should allow air traffic control trainees to acquire more robust coordination skills and reduce the amount of traffic control errors caused by lack of teamwork in actual ATC training situations.

14. SUBJECT TERMS Air Traffic Control, Teamwork, Pilot, En-route controller, Approach Controller, Tower Controller, XML, OOP, RMI, Simulation.	**15. NUMBER OF PAGES** 117
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

A TEAMWORK-ORIENTED AIR TRAFFIC CONTROL SIMULATOR

Mounir Sidhom
Major, Tunisian Air Forces
Engineering Degree, Universita degli Studi Frederico II, Naples, Italy 1992
Master in Computer Science, ENSICA, France 1996

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN
MODELING, VIRTUAL ENVIRONMENTS, AND SIMULATION

from the

NAVAL POSTGRADUATE SCHOOL
June 2006

Author:	Mounir Sidhom
Approved by:	Arnold Buss Thesis Advisor
	Don McGregor Second Reader
	Rudy Darken Chair, MOVES Academic Committee

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Air Traffic Control (ATC) is a complicated domain in which many specialists should collaborate and communicate with each other in order to guarantee safe and efficient air traffic. A significant number of air traffic control errors are associated with either faulty coordination between ATC actors, or a failure of some kind of team coordination. These errors are likely to increase in the future as aircraft density increases. Many researchers suggest that the introduction of team and teamwork concepts during the training phase of the ATC actors will be in help to reduce the amount of these errors.

The objective of this research is to conceive, design, and implement a teamwork-oriented Air Traffic Control simulator that can be easily installed and used in ATC schools. The product of this thesis will be a complete software package that allows trainees in the different ATC specialties to work together in the same manner as they do "on-the-job" in order to collaboratively manage an air traffic situation. This type of simulator should allow air traffic control trainees to acquire more robust coordination skills and reduce the amount of traffic control errors caused by lack of teamwork in actual ATC training situations.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1

II. BACKGROUND AND RELATED WORK....5
- **A. SIMULATION IN ATC....5**
- **B. BENEFITS OF SIMULATION IN ATC....6**
- **C. ATC SIMULATORS....8**
  - **1. The Tower Research Simulator....10**
  - **2. The Total Airspace and Airport Modeler (TAAM)....11**
  - **3. SIMMOD....12**
  - **4. Re-organized ATC Mathematical Simulator (RAMS)....13**
- **D. IMPORTANCE OF TEAMWORK IN ATC....14**
- **E. ATC AS A BIG TEAM....16**
- **F. CONCLUSION....16**

III. GENERAL ISSUES AND USE CASES OF AIR TRAFFIC CONTROL....17
- **A. INTRODUCTION....17**
- **B. A BRIEF HISTORY OF AIR TRAFFIC CONTROL....17**
- **C. AIR TRAFFIC CONTROL....19**
  - **1. Tower Control....21**
  - **2. Approach/TRACON Control....23**
    - ***a. Departure Aircraft....23***
    - ***b. Arrival Aircraft....23***
    - ***c. Over-flight Aircraft....24***
    - ***d. VFR Aircraft....24***
    - ***e. IFR aircraft....25***
  - **3. En-route Control....25**
- **D. USE CASES OF AIR TRAFFIC CONTROL SIMULATOR....26**
  - **1. Tower Controller Use Cases....29**
  - **2. Approach Controller Use Cases....30**
  - **3. En-route Controller Use Cases....30**
  - **4. Pilot Use Cases....31**
  - **5. Instructor Use Cases....32**
- **E. CONCLUSION....33**

IV. SYSTEM SPECIFICATION AND DESIGN....35
- **A. INTRODUCTION....35**
- **B. SPECIFICATION....36**
  - **1. Specification for the Approach Part....36**
  - **2. Specification for the En-route Part....41**
  - **3. Tower Control Specification....43**
  - **4. Pilot Specification....43**
  - **5. Instructor Specification....45**
- **C. APPLICATION DESIGN....46**

1. **Overall Application Design** **46**
2. **The Communication Module** **48**
3. **The Instructor Module** **53**
4. **The Approach Module** **56**
5. **The Tower Module** **58**
6. **The Pilot Module** **58**

D. CONCLUSION **60**

V. IMPLEMENTATION OF THE ATC SIMULATOR **61**

A. MAP REPRESENTATION **61**

B. THE SERVER/INSTRUCTOR MODULE **62**

C. THE PILOT MODULE **64**

1. **The Cockpit Module** **65**
2. **The Ground Module** **66**
   - ***a. Aircraft Mover*** ***68***
   - ***b. The Takeoff Processor*** ***69***

D. THE TOWER MODULE **70**

1. **Placing Algorithm** **71**

E. APPROACH MODULE **72**

1. **Coordinate Conversion** **75**
2. **Runway Representation** **76**

F. THE EN-ROUTE MODULE **78**

VI. CONCLUSIONS AND FURTHER WORK **79**

A. CONCLUSION **79**

B. FUTURE RESEARCH **80**

1. **Usability Study** **80**
2. **Design of Experiments** **81**
3. **Rebuilding the Tower Module** **81**

LIST OF REFERENCES **83**

APPENDIX A. GLOSSARY AND ABBREVIATIONS **87**

APPENDIX B. THE SERVICES' INTERFACES **91**

APPENDIX C. SCENE AUTHORING INTERFACE (SAI) **99**

INITIAL DISTRIBUTION LIST **101**

LIST OF FIGURES

Figure 1. The ATC Loop 6
Figure 2. The Different Categories of Air Traffic Control 21
Figure 3. Tower Controller Use Cases 29
Figure 4. Approach Controller Use Cases 30
Figure 5. En-route Controller Use Cases 31
Figure 6. Pilot Use Cases 31
Figure 7. Instructor Use Cases 32
Figure 8. The Waterfall Life Cycle Model 36
Figure 9. Typical PSR output 37
Figure 10. Typical SSR output 39
Figure 11. Aircraft separation in en-route control 42
Figure 12. The overall application design 47
Figure 13. Client/Server interaction in CORBA 49
Figure 14. CORBA communication architecture 50
Figure 15. RMI Client/Server architecture 51
Figure 16. The ATC user service 53
Figure 17. Headings in ATC 55
Figure 18. Overall approach module design 57
Figure 19. The Pilot Module Design 59
Figure 20. Bitmap and XML representation of the map 61
Figure 21. The XML schema of a map 61
Figure 22. Visualization of the xml data in the Approach and En-route application 62
Figure 23. UML diagram of the server module 63
Figure 24. XML schema of the flight plan 63
Figure 25. The server module GUI 64
Figure 26. Cockpit GUI of the pilot package 66
Figure 27. The UML diagram of the cockpit module 66
Figure 28. The ground GUI of the pilot package 67
Figure 29. The UML diagram of the ground part 68
Figure 30. The takeoff verification process 70
Figure 31. The UML diagram of the tower module 71
Figure 32. The placing algorithm 72
Figure 33. The UML diagram of the approach module 73
Figure 34. The approach workstation GUI 74
Figure 35. A simple geographic area representation 75
Figure 36. Runway extensions 77
Figure 37. 3D model of the Tower module 82
Figure 38. 3D model of Tunis airport rendered by SAI 100

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1. Common ATC Simulators' Techniques.....9
Table 2. Main ATC specialist Categories and their Tasks.....28
Table 3. Special purpose SSR codes and their uses.....46
Table 4. A typical instructor's event display GUI.....54
Table 5. Interpretation of the pseudo-code used in the Instructor GUI.....55

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

This research would not have been possible without the encouragement of my wife Faten, my children Seif and Sirine, my father Mokhtar, and my mother Fatma. Thanks a lot from all my heart for your moral support. My family means the world to me and without them I would be nothing. I would like to publicly acknowledge the great sacrifice that my family has endured for the past two years while I conducted research in the United States. Thank you.

This research was conducted in coordination with the MOVES Institute at the Naval Postgraduate School. This institute is comprised of many professional experts in modeling, simulation, and web-based development. I would be remiss if I did not mention a few individuals whose support and knowledge was extremely helpful in the completion of this simulation model. The first person I would like to thank is my thesis supervisor, Professor Arnie Buss. I have known Professor Buss during the classes that he is teaching, and I discovered that he is a caring professor with full of wisdom and experience. From him I learned a lot of skills that certainly will be useful in my life. I am really glad that I came to have him as a teacher, and as an advisor for my thesis. Furthermore, I would like to thank Professors Don Brutzman and Don McGregor, whose knowledge of web-based services and development was absolutely critical to the success of this research. Finally, I would like to thank Professor Rudy Darken and CDR Joseph Sullivan for their efforts in leading the institute. Their expert management and guidance helped ensure my success along the way.

In conclusion, many of my classmates and country mates supported me and encouraged me in implementing this work. In particular, Jon Ellis was in great help in the scheduling and the reorganization of my thesis. Travis Rauch is always happy to participate in executing and debugging the application. Many thanks for them and any other who supported me.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Air Traffic Control (ATC) is a network of facilities whose purpose is to organize aircraft movements in airspaces in order to guarantee safe and efficient air traffic. ATC contains many kind of specialists (ATCS) commonly grouped in three main categories: tower, approach and en-route. To meet the required skills necessary to manage an ever-increasing air traffic density, these specialists should be trained continuously. However, often the training method consists of using separate modules for each air traffic control category. The most critical deficiency of such training method is the lack of the concept of teamwork.

According to some researchers, a significant number of air traffic control errors are associated with either faulty coordination between ATC specialists, faulty coordination between ATCSs and pilots, or a failure in some kind of team coordination. Much research emphasizes the importance of teamwork in ATC. Lintner and Buckles (Lintner & Buckles, 1992) found in their study that more than 30% of operational errors (violations of aircraft separation minima) are associated to some kind of communication deficiency between the ATC actors. In an analysis sponsored by the Federal Aviation Administration's Office of the Chief Scientific and Technical Advisor for Human Factors (Federal Aviation Administration [FAA], 2002), based on a sample of 386 Aviation Safety Report System (ASRS). The results of this analysis show that 90% of the errors are attributed to some category of communication errors. According to a study by Rogers and Nye, coordination between controllers was considered a causal factor in 15% of 1,038 low to moderate severity operational errors from 1988-1991 (Bailey, Broach, Thompson, Enos, 1999).

It is clear, from the examples shown above, that the lack of coordination between the different actors of the air traffic is a direct cause of a non-negligible number of incidents that could have grave consequences. Hence, the concept of team coordination and teamwork are keys to a successful air traffic management. One obvious way to acquire the necessary teamwork skills is through On the Job Training (OJT). However, as stated by O'Hare and Roscoe (O'Hare & Roscoe, 1990), the stress of live performance and the difficulty of providing immediate and accurate feedback makes OJT a less-than-ideal learning environment.

Recent advances in computation and communication technologies can be easily used to expand the options for a training design and methodologies in ATC by producing a simulated system that mimics the real working environment. Simulations can be developed with a great degree of realism and fidelity; hence, they can allow the OJT phase to be avoided or at least reduced. Even though ATC simulators are quickly expanding in number, they still suffer deficiencies, including lack of portability, incompleteness, and in general they are closed systems. In addition, the most accurate simulators require a high cost of production and maintenance, which may not be affordable for many countries. Many developing countries are actually using different applications for their ATC trainees, and often these applications are either incompatible with each other or they do not allow exchange of data. In such conditions, it is difficult, if not impossible, to implement the concept of team work.

The work in this thesis consists of designing and implementing a teamwork-oriented air traffic control simulator that could be used in ATC schools. The goal is to allow different ATCSs to work together, in the same conditions as the job environment, in order to manage an air traffic situation in a collaborative way. A simulator like this should allow ATCSs to acquire further coordination skills, and hence, reduce the amount of air traffic faults due to a lack of teamwork in the actual ATC training system.

A teamwork ATC simulator is a flexible training tool for both trainees and instructors. A simulator like this will help in obtaining a low cost solution that can substitute buying simulators for each category of ATCSs. In addition, the main training advantages of this simulator could be summarized as:

1. Decreasing the OJT time.
2. Help to increase teamwork performance, and hence, reduce air traffic errors.
3. Give insight into teamwork requirements for Air Traffic Control Simulators.

This work is organized in six chapters, the first of which is this introduction. The second chapter reviews the most related works about the objective of this thesis. Particularly, it emphasizes the role of simulation in ATC and lists the most important drawbacks about some examined ATC simulators. The end of the chapter illustrates the importance of teamwork and its impact in the ATC activities. The third chapter gives a general overview about the air traffic control domain, including some historical events and the most important concepts of ATC relevant for the scope of the thesis. Since a waterfall model will be adopted as the software development methodology of this work, this chapter is viewed as the acquisition phase of this model. The end of the chapter illustrates the different use cases of the ATC simulator. The fourth chapter is dedicated to the specification and the design phases of the project. It will examine the capabilities and constraints of the intended product. The fifth chapter contains the development phase of the simulator. During this chapter, the different technical aspects of the different modules constituting the developed software will be examined. The sixth chapter concludes this thesis by a summary of the most important benefits of this work. This chapter also illustrates some future steps of the produced simulator such as how it can be used to evaluate trainees, and how it can be used in a designed experiment to demonstrate training improvement.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND AND RELATED WORK

With advances in computer graphics and simulation techniques, many private and governmental companies interested in Air Traffic Control (ATC) have produced very sophisticated ATC simulators. With today's ATC simulators, air traffic control specialists have the opportunity to practice their skills in a simulated environment exactly as they would use a real environment (as in On-the-Job Training (OJT)). As Smolensky and Stein (Smolensky & Stein, 1998) mentioned, the simulated environment can offer samples of any desired density of simulated aircraft that can be "flown" and presented to the controller subjects using scenarios that have fully definable and experimentally controlled levels of complexity.

This chapter is divided into three parts. The first part (the first two sections) discusses the relationship between the real ATC world and ATC simulators and gives some important aspects of simulation in ATC. The second part is a summary of some ATC simulators. Particularly, it illustrates some aspects of ATC simulators, giving their most important characteristics and their limitations. The third part is a collection of ideas and opinions about the importance of team and teamwork in ATC.

A. SIMULATION IN ATC

As stated in the introduction, ATC is an evolving domain with rules and technological mean that change continuously in order to achieve a better degree of safety and to optimize further airflow. Since training should be correlated to the real world, any changes in the real world rules, procedures, or technologies should have a direct impact in the way training is conducted. At the beginning of the ATC era, training was accomplished in two main phases. The first phase is an illustration of the theoretical concepts of the ATC in general and the specific concepts related to a given category of ATC. The second phase is based on the use of some models that represents a specific category of the ATC. For example, in the case of the tower control, often the model consists of a room containing a tower-like shape, and small model of the airport with aircraft figurines with position that can be updated by the trainees themselves. As technology has increased, the figurines of the second phase have been replaced simulation models. Figure 1 shows how simulation is integrated into the ATC loop. Generally new concepts and rules in ATC (e.g., new international accords, flying rules, and human factor concepts) have a direct impact in how the real ATC will be conducted. At this point, simulation techniques could be useful as a means to represent, integrate, and test the new situation. Once a simulation is implemented and tested, it can be used as a training tool and allow controllers to acquire the required skills necessary for the real ATC.

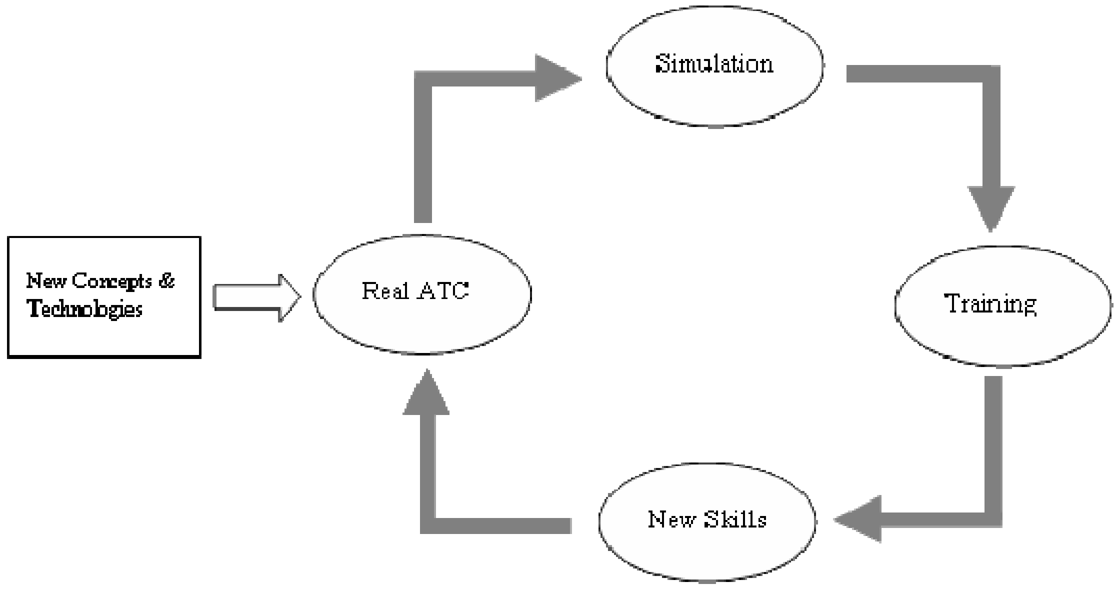


Figure 1. The ATC Loop

B. BENEFITS OF SIMULATION IN ATC

Recent revolutions in computer technologies have made it possible to simulate the ATC process with a level of realism that is far beyond what was envisioned even a few years ago by the most vivid imaginations. It is now possible to present experienced controllers with simulated ATC scenarios that are virtually indistinguishable from the real world to which they are accustomed (Smolensky & Stein, 1998). The National Research Council expressed that simulation can be especially useful in support of investigating how multiple systems do or do not harmonize and of the impacts of new systems on teamwork (National Research Council, 1998). Many research studies have shown the benefits of ATC simulation as a training tool and as a studying environment especially for human behaviors and human factor issues in ATC.

According to Smolensky and Stein there are at least three benefits using simulation in ATC. First, simulation is a cost-effective training solution. Today, even though commercial ATC simulators have a relatively high cost, they often reduce the training time and enhance the Time of Transfer (TOT). In addition, many companies offer a game-like ATC simulator that has a low cost, and these simulators can be useful at least in the pre-training or during the selection phases of ATC. Secondly, ATC simulation is a safe and a well-controlled environment to test, examine, or conduct evaluation studies of new air traffic operational concepts, new international navigation rules, new hardware or software used to automate some parts of the ATC, or simply to test an airport capacity. The third benefit of simulation in ATC is its capability to train the controller and/or controller team. Neophyte controllers can be given the chance to develop without any attendant fear of the potential hazards that could be the result of their inexperience if they were operating within the real world (Smolensky & Stein, 1998). The fourth benefit of ATC simulators is that they can be characterized with a high fidelity with respect to the real operational system. They can mimic most aspects of the real world in detail, provided that all components are well designed and implemented. This capability is very important since it can minimize or even eliminate the OJT phase.

The ATC loop exposed in Figure 1 has another benefit. It can indirectly reduce the total air traffic cost if the different controllers are well trained and integrated in it. According to a document of the Federal Aviation Administration (FAA), the modern transportation system is characterized by a large amount of congestion and delay. The FAA long range forecasts have projected nearly 50% growth in commercial aircraft operations (domestic and international carriers) between now and the year 2025. The U.S. Department of Transportation (DOT) stated that commercial aviation delays are estimated to cost airlines over $3 billion per year. Passengers are directly affected by missed flight connections, missed meetings, and loss of personal time. There are approximately 20 congested airports, each averaging over 20,000 hours of flight delay per year (DOT, 2002). Obviously, reducing aviation time delay isn't a matter of only air traffic controllers; it requires a conjoint effort from all aviation sectors. However, well-trained air traffic controllers will significantly reduce the cost of air delay. In this case, simulation appears to be an ideal solution because it can generate any hypothetical traffic, and trainers will gain insight and skills to deal with such congested traffic.

C. ATC SIMULATORS

Today's ATC simulators both governmental and commercial offer a myriad of options and settings, allowing coverage of the major parts of training requirements. First, it is possible to control the density of aircraft and their relative characteristics such as the speed and the flying level to meet the trainees' needs. Secondly, in some cases it is possible to switch from one airport to another, which is helpful for the user to gain insight about a particular airport such as number of runways, type of communication devices, navigation facilities, and airport conditions (wind, visibility, etc.). Thirdly, most ATC simulators offer the possibility to introduce some unexpected events and abnormal aircraft conditions. Although such events have generally low frequency in a normal air situation, all ATC trainees should know their symptoms and how to handle them. These anomalies may include one or more of the following patterns:

- Unexpected changes in weather conditions
- Communication failure between controller and aircraft
- Communication failure between controllers

- In-flight emergencies (such as aircraft hijack, communication failure, engine failure, or fire). These anomalies are generally signed out to the controller by a change of the aircraft Secondary Surveillance Radar (SSR) code.

The various ATC simulators differ by the category or type of air traffic control they simulate (VFR, IFR, ILS, etc) as well as in their implementation method and technology. Table 1 gives a summary of the most important ATC categories and their possible implementation techniques.

ATC simulator	Visualization	Implementation techniques
Tower Control	- 2D screen - 3D environment - cave technology -Virtual environment (VE)	- simple simulation - expert systems - agent based systems - neural network - voice recognition - blackboard systems
Approach Control	- 2D screen	
En-route Control	- 2D screen	

Table 1. Common ATC Simulators' Techniques

Air traffic simulators also differ in costs and in the modes in which they operate. For example, some of them require a pseudo-pilot or a real pilot, and others require the installation of a voice recognition module to interpret the controller commands and provide an adequate response to these commands. Even though the limitation of the operational modes can be overcome, the cost still remains a significant issue, especially for developing countries. In an article of Canadian Business (McCleam, 2003), Matthew McCleam wrote, "Until recently, technology has not offered a viable solution. Previous simulators required human 'pseudo-pilots' to interact with trainees and fly the virtual aircraft according to their instructions. Three or more could be required to generate enough traffic for just one student, adding considerably to operating costs. Moreover, owing to a lack of processing power and crude video display technologies, the simulated scenarios didn't even begin to look real. Add in a sticker price often in the millions of dollars, and it's no wonder few organizations could afford a single simulator, let alone deploy them at every facility”. To further understand the world of ATC simulators, the next paragraphs give some ideas about some commercial ATC versions of them. It will also be seen that price is one of the major drawbacks of commercial ATC simulators.

1. The Tower Research Simulator

In the domain of tower control simulators, The National Aerospace Laboratory NLR of the Netherlands operates a real-time Tower Research Simulator (TRS) for advanced research and development on airport control tower operations and systems under a variety of meteorological conditions. The TRS[1] is capable of simulating Tower Control and Apron Control activities at airports under nearly realistic operational conditions, with air traffic controllers and pilots in the control loop. The outside visuals, provided with a projection screen of 13 by 4 m, supports daylight and nighttime situations, winter conditions including snow and a variety of meteorological phenomena. This simulator offers a multitude of features such as (NLR, 2006):

- Validation of Surface Movement Guidance and Control Systems (SMGCS), including strategic and tactical support tools and associated controller procedures.
- Validation of Human Machine Interfaces for controller working position design.
- System integration studies on industrial SMGCS equipment, reducing implementation risk.
- Studies of airport capacity, safety and efficiency under dense traffic and marginal visibility.
- Testing and optimization of future tower procedures and airport infrastructures, including legislation and safety assessment.

[1] http://www.nlr.nl/eCache/DEF/250.html, accessed date January 2006.

- Development and validation of ATM automation tools, including data link applications, by Collaborative Decision Making (CDM) and Gate-to-Gate operations.
- Studies and training of airport emergency situations, including rescue operations.
- Safety critical runway operations, for example taxiway/runway crossings with heavy traffic and low visibility.

TRS is one of the most prominent tower simulators, and can be used as a training or research tool for testing controllers' workload, tower team coordination, and airport capabilities. However, it is not clear that TRS can be used in full scale ATC teamwork training or analysis. Furthermore, no information is available about its technical aspects or financial terms.

2. The Total Airspace and Airport Modeler (TAAM)

Another commercial simulator widely used in ATC is the Total Airspace and Airport Modeler (TAAM)[2]. TAAM is developed by the Preston Aviation Solutions in Australia. TAAM was described by its company as a system that can simulate traffic at an extremely detailed level from the departure gate of one airport to the arrival gate of another airport in the same city or midway across the globe. The scale of simulation can vary, ranging from local to national to inter-continental. In one seamless application, TAAM can model an entire airside and airspace environment, taking into consideration gates, terminals, pushback, taxiways, runways, terminal airspace, en-route and oceanic airspace. However, Plaettner-Hochwarth, Zhao, and Robinson (2000) reported that one of the major drawbacks of this software is the price tag. In 1997 a single site license cost about $350,000. In addition to its high price, TAAM also lacks stochastic options and does not cover all ATM components. The rule set it uses is fixed and thus cannot be used for testing new ATM algorithms or concepts. Because of its complexity it requires

[2] http://www.preston.net/Documents/taambrochure.pdf, accessed date January 2006.

substantial resources and training to set up (Plaettner-Hochwarth, Zhao & Robinson, 2000). In a work dedicated to improve TAAM performances, Sood and Wieland wrote that conflict detection in TAAM has the slowest running algorithm compared to the other ATC simulator of the same category. It comprises almost 50% of the total run time. Also they mentioned that another TAAM bottleneck resides in the aircraft navigation algorithm (Sood & Wieland, 2003).

3. SIMMOD

SIMMOD is a discrete event based model developed by the Federal Aviation Administration (FAA). It can be categorized by its main feature as an airfield and terminal area airspace model. The personnal edition of SIMMOD is offered with an affordable price (around $6,000 plus additional basic annual support fee of $3,000 for a single user license). The professional edition, SIMMOD PRO, includes an advanced simulation engine that greatly expands the simulation capabilities and enhances the graphical user interface. SIMMOD PRO is offered with a price around $60.000[3]. SIMMOD is described by Aviation Test and Analysis Corporation[4] (ATAC) as a system that was designed to "play out" airport and airspace operations within the computer and calculate the real-world consequences of potential operating conditions. It has the capability and flexibility to address a wide range of "what if" questions related to airport and airspace capacity, delay, and efficiency, including questions associated with:

- Existing or proposed airport facilities (e.g., gates, taxiways, runways, pads)
- Airport operating alternatives (e.g., taxi patterns, runway use, departure queuing)
- Existing or proposed airspace structures (e.g., routes, procedures, sectors)
- Air traffic management/control technologies, procedures, and policies

[3] http://www.dlr.de/esug/meetings/25th/minutes.pdf. (page 7), accessed date January 2006.

[4] http://www.atac.com, accessed date January 2006.

- Aircraft separation standards parameters (e.g., weather, aircraft type, flight state)
- Airline operations (e.g., flight schedule, banking, gate use and service times)
- Current and future traffic demand (e.g., volume, aircraft mix, new aircraft types).

The major drawback of this model is that it uses a node-link system on which all the aircraft move, as opposed to a 3D simulation like in TAAM (Plaettner-Hochwarth et al., 2000). Because of this implementation type, it is inflexible and cannot be used to simulate new concepts like free flight.

4. Re-organized ATC Mathematical Simulator (RAMS)

Another example of ATC simulator is the Re-organized ATC Mathematical Simulator (RAMS). RAMS is a fast-time, discrete-event computer simulation model developed and supported by the Model Development Group (MDV) at Eurocontrol, France. RAMS is distributed in many versions. The RAMS Plus is the most popular version. The vendor is currently licensing RAMS Plus as a community-supported tool. There is an initial licensing fee, and yearly continuing support fees which allow access to support and new releases/patches. In 2003 the price was $15,000 for a single-machine license, and includes the ATM Analyzer. Continuing support is currently priced at $5000 per year. In their work, Plaettner-Hochwarth and his companions described RAMS as a General purpose ATC modeling environment for en route and terminal airspace as well as controller workloads. They added, "Since this software is only available through Eurocontrol and since it uses a closed structure, its assumptions are basically unknown" (Plaettner-Hochwarth et al., 2000).

In conclusion, the majority of ATC simulators generally offer good features such as the possibility to play training exercises according to the International Civil Aviation Organization (ICAO) standards, the possibility to record and playback the exercises, and the possibility to develop new scenarios. However, the majority of ATC simulators present some drawbacks on one or more of the following categories:

- Difficult to be adapted to new scenarios.
- They are confined to a specific category of ATC.
- Difficult to be used and maintained.
- Not designed initially to support teamwork coordination.
- Portability issues.
- High prices.

D. IMPORTANCE OF TEAMWORK IN ATC

The idea that some kind of coordination or communication between the different ATC actors (controllers and pilots) is not a new concept. Attempts to develop international air traffic control (ATC) rules addressing language and pilots' needs to communicate date back to 1922 (Ruiz, 2004). Nowadays, it is quite common when reading any book about air traffic control or surfing the net looking for articles related to safety and human factors issues in ATC, to discover that teamwork is an important factor in the success of the overall air traffic. It is also not hard to determine that the lack of training methodologies in it generally lead to the increase in the number of air incidents and errors. The following sections give an illustration of the work of some researchers in the domain of team and teamwork in ATC, and the statistical conclusions of their analyses.

Serious studies about communication, coordination, and teamwork in ATC date back nearly two decades. Adams and Lynn conducted an analysis of operational errors between 1985 and 1988 and found that more than half of the errors were associated with either faulty communication between air traffic controllers or with faulty coordination between controllers and pilots (Smolensky & Stein, 1998). Almost in the same period, Lintner and Buckles (1992) concluded that the ATC system can not work correctly unless pilots and controllers can communicate effectively and understand each other (Ruiz, 2004).

The FAA's long range forecasts have projected nearly 50% growth in commercial aircraft operations (domestic and international carriers) between now and the year 2025. This will cause much preoccupation for all air traffic service providers as to how to meet the forecast demand. This problem and others were discussed during a workshop organized by the European Organization for the safety of Air Navigation in 1998. Prior to this meeting, the European Air Traffic Control Harmonization and Integration Program (EATCHIP) was charged to analyze this problem and suggest recommendations. The EATCHIP recognized that increasing capacity in European airspace while maintaining a high level of safety is not simply a technical solution but also should involves human factor concepts. EATCHIP concluded that ATS providers could produce some rapid and efficient results by taking into consideration the human resource issues, and teamwork is one of these issues (EUROCONTROL, 1998).

The same conclusion and recommendation were also mentioned by Smolensky and Stein in their book where they mentioned that ATC is likely to benefit most from a coordination training intervention that is specifically tailored to enhance the teamwork skills incumbent in ATC tasks. They also added that without training on the specific coordination skills that constitute effective ATC team coordination, Air Traffic Control Specialists (ATCSs) are likely to develop coordination patterns that are less than optimal (Smolensky & Stein, 1998).

Controllers themselves are aware of the importance of teamwork and coordination skills in ATC. In 2001, D'Arcy and Della Rocco conducted a survey study with the participation of 103 ATCSs. In their executive summary, they mentioned that many participants' reports emphasized the collective nature of ATC. In particular, the participants declared that controllers must coordinate their actions and plans with many other actors, such as pilots and controllers working with and around them. Also, their results suggest that controller situation awareness generally includes knowledge of the skills and preferences of the other controllers. The importance of teamwork was also emphasized when participants reported fighting boredom by watching other sectors and protecting other controllers. Helping without a specific request corresponds to the highest level of team coordination (D'Arcy & Della Rocco, 2001).

E. ATC AS A BIG TEAM

All the examples shown in the previous section emphasized how important teamwork is in ATC. This appears somewhat obvious since air traffic controllers are not alone in their daily routine, but each of them is a critical element of a big chain representing the ATC Team. Each of them makes decisions based on the situation of other team members. This fact can help immensely to compensate for differences in performance between the team members; hence teamwork skills lead to a more efficient and safer management of the air situation.

One safe and efficient way to acquire teamwork skills in ATC can be achieved through the use of ATC simulators. Stimulating teamwork via ATC simulators has many benefits. The most important of them is the reduction or even the elimination of the OJT phase of the ATC training cycle.

F. CONCLUSION

In this chapter, it was shown that team and teamwork are important concepts in air traffic control. These concepts can be easily embedded in a simulation-based ATC training system. Unfortunately, existing simulators have some limitations, and none of them is teamwork-oriented.

III. GENERAL ISSUES AND USE CASES OF AIR TRAFFIC CONTROL

A. INTRODUCTION

Air Traffic Control (ATC) is a set of rules and facilities having roots that extend to the first decade of the last century. Today, ATC is a huge domain headed by the International Civilian Aeronautical Organization (ICAO) and it is represented in each country by a local organization. ACT contains many specialists to cover the control of the airspace at any time and in any circumstance. Among these specialists, there are three categories of interest: the tower, approach, and en-route specialists. The description of these specialists and the aspect of their tasks will be explained in section D below. However, first a brief history of the ATC will be presented. The objective of this history is to give an idea in how the ATC evolved form the first decades of the last century until the present. It will also help by defining and understanding some concepts in the requirement phase of this work. It is not a complete story; however, it cites the most important milestones and events having a great influence in today's ATC situation. Next, the different parts and rules of ATC will be explained**.** The end of this chapter will be dedicated to use cases of the projected simulator.

B. A BRIEF HISTORY OF AIR TRAFFIC CONTROL

When the Wright brothers made their brief successful flight, nobody thought that the aviation domain would become one of the world's most powerful "team" with millions and millions of dependents and aircraft. Furthermore, everyone thought that the sky was so vast there was little or no risk for one aircraft to collide with another. However, this belief was short-lived. Only seven years after the Wright's experiment, many countries realized the necessity to regulate the aviation domain by introducing some navigation rules and some ground facilities to guide pilots from their departure to their destination location in a safe and efficient way. This necessity becomes more urgent after four mid-air collisions in 1910 and six in 1912 (National Air Traffic Service, 2005). As a consequence of all these events and the increasing use of the shared airspace, especially in Europe where the density of industrialized countries is high and the shared airspace is relatively small, the International Commission for Air Navigation (ICAN) was created in Paris in 1919. Among the main purposes of ICAN were the establishment of uniform rules and standards for aircraft registration and identification, personnel licensing, maps and charts, and most importantly, for the ATC domain, establishing rules and giving solutions for air and flying procedures.

The first reported solution to control air navigation was a very simple one, consisting of some arrows drawn on the ground to help pilots correct their direction. In addition, flights were restricted to daytime operation with good weather conditions and sufficient visibility. In these cases, given the technological level of the aircraft (limited speed and altitude), the pilot was generally able to distinguish the relevant landmarks (such as big plants, roads, railway lines, and rivers), and it was fairly straightforward for him to successfully guide his plane from one point to another. In such low density air traffic, the only limitations on safe flight, excluding mechanical problems, were bad weather or darkness. A first important step to reduce these limitations was the use of simple ground signaling lamps to communicate to the pilot information such as the limit of the runway, the landing direction, and permission of takeoff or landing. The lamp-based signaling solution was a significant milestone in the world of ATC, and it still is used in all regulated airports around the world. Even though today a pilot can navigate without the necessity of looking outside his or her cockpit, the signaling system remains the ultimate backup system in case of communication device failure or a total electricity blackout.

The next step in the advance of ATC was the use of the electromagnetic waves. These waves can carry much information by changing one or more of their characteristics such as the frequency, the phase, or the amplitude. This concept was first proved by

Marconi on December 1894, and was successively used as an ATC facility in the 1930's when Marconi Company first installed a single-channel radio at Croydon, UK (Aviation Sri Lanka, 2005). Although the main purpose of this radio communication was not to serve the ATC, it turned out that it is indeed an efficient solution to regulate air traffic. In fact, it was easy for pilots, with this new facility, to communicate precious information such as position, estimated arrival time, weather conditions, and any other information relevant for the flight safety.

Another important milestone was the invention of the radar, which not only made it possible for the pilot to know his position, but for Air Traffic Controllers also to know the aircraft's position as well. It is clear that, especially to prevent air-crashes, this was a very important invention. Although the radar was invented in 1922 by Marconi, it took many years before it became an essential hardware piece in the Air Traffic Control. It was only after the Second World War that the radar and other communication devices were made available to the civil aviation for air traffic regulation and management. The international flight rules and regulations were further finalized in 1947 after the establishment of the ICAO. After that date the sky became a regulated space with well defined airways, Flight Instruction Regions (FIR), and many other rules to facilitate navigation and air control.

Nowadays, the ATC contains a complicated set of communication, detection, and visualization equipments making it a highly automated domain. However this automation does not come for free. It requires a large amount of knowledge and practice from the specialists of ATC, and often automation implies the introduction of new skills, and new training methodologies, as explained in the paragraph "ATC simulator."

C. AIR TRAFFIC CONTROL

After this brief history of ATC, the next logical step is to present the basic ideas about air traffic control activities. Since the aim of this thesis is to develop simulation software that can be used in ATC training phases with emphasis on the concept of teamwork, the following paragraphs describing the different categories of ATC that can be retained as part of the requirement and specification phases of the software. The overall development of the simulator will follow the waterfall model. Requirement and specification are the first two steps of this model. This model was preferred to other approaches (prototype, spiral, extreme programming, etc) because of its simplicity as well as ease of understanding and implementation. In addition, the waterfall model forces a verification activity in between each two adjacent steps which guaranties a high degree of fidelity and robustness to the final software.

Air Traffic Control (ATC) is the organization of aircraft movements in airspace, including methods and procedures used to manage and safeguard air traffic. Since air traffic is shared by all countries in the world, it is obvious that flight rules and organization should be standardized in some way so all the pilots in the world follow the same procedures independently of the country or the region in which they are flying. The International Civil Aviation Organization (ICAO), mentioned earlier, is the official worldwide center responsible for this standardization. In addition, within each country, there is an organization that works in collaboration with the ICAO to produce the necessary air documents and specific rules for that country. In the United States, for example, that organization is the Federal Aviation Administration (FAA).

The actors in ATC (which will be described in detail in paragraph D) are the crew of the aircraft and the different air controllers. ATC is a collaborative domain and it is based on the cooperation between the crew and the specific air traffic controller. An essential part of it is the exchange of communication and data. The objectives of ATC can be summarized as following:

- Maintaining in any circumstance a reasonable separation between all aircraft in a given air domain.
- Preventing collision between aircraft.
- Preventing aircraft from crashing into obstacles on the operating ground of the controlled domain.

- Maintaining a reasonable and orderly air traffic flow.

There are three main interdependent cooperative activities or areas of responsibilities in ATC. Each area is operated by a set of controller category and has a set of specific rules. These areas (shown in Figure 2) are:

- Tower control Area.
- Approach or Terminal Radar Control (TRACON) area.
- En-route Area.

The next three paragraphs give the most important characteristics of each area, such as its specific definition, and the some of its rules.

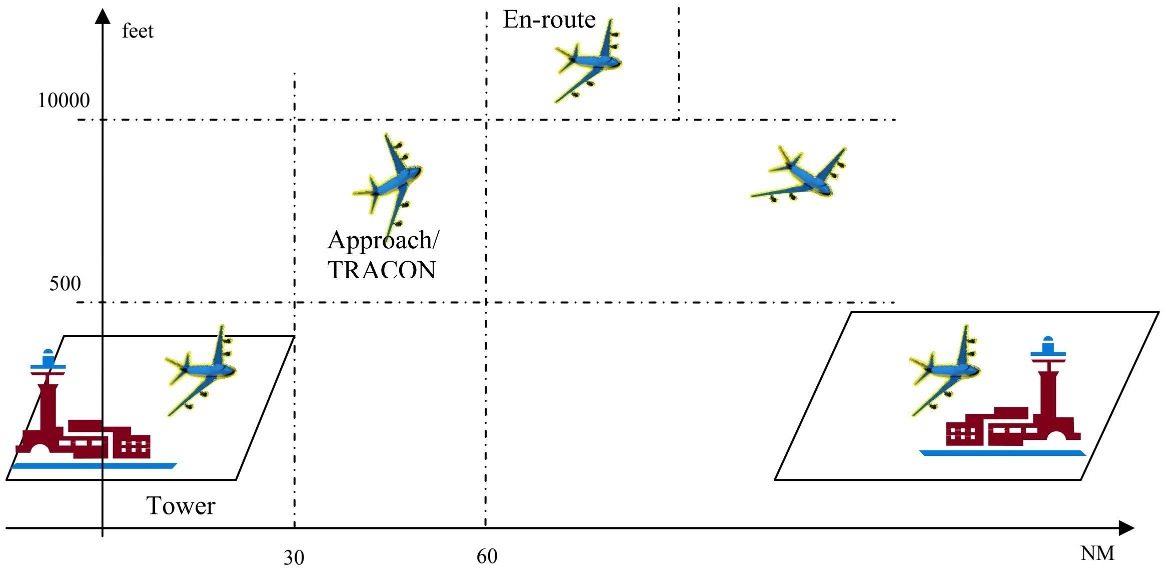


Figure 2. The Different Categories of Air Traffic Control

1. Tower Control

Tower control is the set of activities and procedure to control any movement (ground vehicles or flying objects) within the area of the airport. This area includes the airport itself and the surrounding space. It is generally defined as an airspace volume with a radius of 2 to 30 miles and a height of 1,500 to 2,000 feet centered in the airport. The main facility for the tower control is the control tower from which the tower control specialists conduct their control activities. Control towers are typically higher than other structures at the airport in order to give air traffic controllers a view of aircraft moving about on the ground and in the air around the airport. They usually have windows that circle the entire top floor, giving 360 degrees of viewable area. Larger airports usually have space for several controllers to work and operate 24 hours a day, 365 days a year. Small airports may have only one person staffing the control tower, and may not even keep the tower open 24 hours per day. Control Towers usually contain the following equipment:

- Communication devices allowing internal (between controllers) and external (with the aircraft crew) communications.
- A light gun for signaling certain events to aircraft crew in the event of a radio failure.
- One or more radar systems that controllers use to track aircraft.
- A land-line telephone or possibly even a direct line to fire and ambulance services.

Airport Tower Controllers regulate a specific airport's traffic. They use two-way radios to give pilots permission to take off and land. They also direct ground traffic, which includes taxiing aircraft, vehicles, and airport workers. Outside the Tower Control area aircraft is subject to either Approach or En-route Control. There are several categories of ATCS in the tower including:

- Flight Data Controller
- Clearance Delivery Controller
- Ground Controller
- Local Controller

Later, in the user case analysis, more details about different ATCSs, their tasks, and their needs will be given.

2. Approach/TRACON Control

In ATC, often Approach Control is used interchangeably with TRACON, an acronym for for Terminal Radar Approach Control. Approach Control is usually located within the vicinity of a large airport (a tower control with the coordination with en-route control is sufficient for a small airport with very low traffic). Typically, the TRACON controls aircraft within an airspace defined by a radius of 30-50 nautical miles (56 to 93 km) from the center of the airport, and a height between the surface and 10,000 feet above the airport. Approach Control is responsible for providing all ATC services within the above airspace. Generally, there are four types of traffic flows controlled by TRACON controllers (Wikipedia, 2005). These are departures, arrivals, overflights, and aircraft operating under Visual Flight Rules (VFR) or Flight Instrument Rules (IFR). The following paragraphs give some clarifications about these categories of air control.

a. Departure Aircraft

Departure aircraft are handed off from the tower to the TRACON when they are between 1,000 feet to 2,000 feet high, climbing to a pre-determined altitude. The TRACON controller working this traffic is responsible for clearing all other TRACON traffic and, based on the route of flight, placing the departing aircraft on a track and in a geographical location (sometimes referred to as a "gate") that is pre-determined through agreements for the en-route center controller. This positioning is designed to allow the en-route center to integrate the aircraft into its traffic flow easily.

b. Arrival Aircraft

Arrival aircraft are handed off from the en-route center in compliance with pre-determined agreements on routing, altitude, speed, spacing, etc. to the TRACON center. The TRACON controller working this traffic will take control of the aircraft and handle it with other aircraft entering the TRACON from other areas or "gates" into a single file or final for the runway. This spacing is critical to ensure the aircraft can land and clear the runway prior to the next aircraft touching down on the runway. The tower may also request expanded spacing between aircraft to allow aircraft to depart or to cross the runway in use.

c. Over-flight Aircraft

Over-flight aircraft are aircraft that enter the TRACON airspace at one point and exit the airspace at another without landing at an airport. They must be controlled in a manner that ensures they remain separated from the climbing and descending traffic that is moving in and out of the airport. Their route may be altered to ensure this is possible. When they are returned to the en-route center, they must be on the original routing unless a change has been coordinated.

d. VFR Aircraft

VFR or Visual Flight Rules are a set of aviation regulations under which a pilot may operate an aircraft, providing weather conditions are sufficient to allow the pilot to visually control the aircraft's attitude, navigate, and maintain separation with obstacles such as terrain and other aircraft. Pilots flying VFR aircraft assume responsibility for their separation from all other aircraft (IFR & VFR) and do not have set routes and altitudes. They fly on their own using a "see and be seen" separation criteria. In certain airspaces, VFR aircraft are required to have a transponder. This amplifies the radar signal, as well as broadcasting altitude level and SSR code, and is used to allow controllers to warn IFR aircraft of any potential conflict. Governing agencies establish strict VFR "weather minima" for visibility, distance from clouds, and altitude to ensure that VFR pilots can see far enough.

VFR pilots can request, and ATC can elect to provide "VFR Advisory Services" if traffic permits. In this environment, the controllers will use radar to identify the VFR aircraft and provide traffic information and weather advisory services for the VFR pilot. Controllers do not provide any instructions concerning direction of flight, altitude, or speed to the VFR pilot receiving advisory services, and they do not provide separation services. This is an optional service and may be discontinued by ATC or the pilot at any time. It is important to mention that VFR rules are different among countries, but, in any case, pilots should have a VFR certificate to be eligible for VFR flight.

e. IFR Aircraft

Pilots flying under IFR must file a flight plan with ATC and accept any revisions ATC requests to their route or altitude. In return, controllers ensure that pilots flying IFR are separated from all other IFR aircraft and terrain by the appropriate minimum separation. The IFR pilot, however, must maintain a close watch for VFR aircraft since ATC has no control over these aircraft. For this reason, VFR aircraft are restricted to altitudes below 18,000 ft. and must have an operating transponder in congested airspace. Once an IFR aircraft is above 18,000 ft (FL 180) the aircraft is considered in "Positive Control Airspace" where only IFR aircraft are allowed.

3. En-route Control

En-route Control is exercised when the aircraft leaves the TRACON volume and reaches its cruising speed and altitude. En-route Air Traffic Controllers issue clearances and instructions to any aircraft as needed, and pilots are required to comply with these instructions. Controllers adhere to a set of separation standards that define the minimum distance allowed between aircraft; these distances vary depending on the equipment and procedures used in providing ATC services.

En-route air traffic controllers work in facilities called Area Control Centers (ACCs). Each ACC is responsible for many thousands of square miles of airspace known as the Flight Information Region (FIR). ACCs are responsible for climbing the aircraft to their requested altitude while, at the same time, ensuring that all aircraft are properly separated from each other at all time. Additionally, the aircraft must be placed in a flow consistent with the aircraft's route of flight. This effort is complicated by cross traffic, severe weather, special missions that require large airspace allocations, and traffic density. As an aircraft reaches the boundary of an ACC control area it is "handed-off" to the next Area Control Center. This "hand-off" process is simply a transfer of identification between controllers so that air traffic control services can be provided in a seamless manner. Once the "hand-off" is complete, the aircraft is given a frequency change and begins talking to the next controller.

D. USE CASES OF AIR TRAFFIC CONTROL SIMULATOR

The use cases technique provides a simple and efficient way to begin many software projects. It quickly gives an overview of the intended product by the mean of simple diagrams called use cases diagrams. These diagrams are not only very simple, but are also an efficient and effective means of communicating with users and other stakeholders about the system and what it is intended to do (Bennett, Skelton & Lunn, 2001). The use cases presented in this section can also be used as the basis of the specification and design phases of the project (Chapter IV).

As stated in Chapter I, the air traffic control environment contains many users who work together in order to maintain a safe and efficient air situation; that is, all aircraft in the area of interest should accomplish their flight plan while avoiding any conflict with other traffic that may endanger their safety. Moreover air traffic control should also take into consideration some other parameters to enhance the quality of the air traffic management system by avoiding unnecessary actions that can cause flight delay (each flight delay will cause the company owning the aircraft a large loss of money), and by keeping a high degree of fluidity in the air traffic and all the other servitude and safety equipment in the airport.

Air traffic control is a heterogeneous environment that contains many specialists or actors. Table 1 gives a detailed list of these specialists. However, for the purposes of this thesis, and since many of these specialists are using almost the same working environment, it is better to treat them as groups or categories of specialists. This simplification will help to focus more in the design of the simulator, and gain a high level view of the environment to be simulated. This decision was also made because not all the air traffic control specialists illustrated in Table 2 require a specific computer environment to carry out their specific tasks. In fact, many ATCS are executing many of their tasks manually or they are using a common software tool to facilitate their decision and job requirements. As consequence, and for the purposes of this thesis, only four modules were designed and implemented (see the first column of Table 2): Tower Control (TC), Approach Control (AC), En-route Control (EC), and Pilot Interface (PI). One additional module will be added for the instructor. Although the simulator developed in this thesis runs only for the four actors mentioned earlier, the instructor module is very important since it helps the instructor control the progress of his or her students and customize the simulation exercises in a way that best fits the trainees' needs. The following sections give a summary of all actors of the simulator. This information will be the base of the requirement phase of this project, and they are presented as use case diagrams.

ATCS Category	Specialist	Responsibilities	Type of tasks & equipments
Tower Control	Clearance Delivery Controller (CDC)	- issue IFR or VFR clearances - direct the initial aircraft movement - cooperate with GC - elaborate the initial flight strips	- manual (Strips) - communication devices - computer software
	Ground Controller (GC)	- control all movement in the airport	- manual (Strips) - communication devices

		- position aircraft in the correct runway -clear runway for arrival	- computer software
	Local Controller (LC)	- establish a correct sequence of aircraft flow (landing, takeoff) - collaborate with GC to clear runway for approaching traffic - respect aircraft timing - managing the flight strips	- manual (Strips) - communication devices - computer software
Approach Control	Flight Data Controller (FDC)	- communicate all flight plans - insure links with other ATCS - keeps record of flight strips	same as above
	Radar Controller	- maintain at every moment an adequate aircraft separation - assign approach sequences. - decides the departure route for every aircraft	same as above
	Supervisor	- monitor the activities of all approach control specialists - solve communication problems - acts as backup controller - makes decisions regarding some crucial air situations	no special equipments
En-route Control	FIR Controller (FC)	- maintain safe aircraft separation - updates and keeps record of flight strips	- manual (Strips) - communication devices - computer software
	Air Space Controller	- supervise all activities in the national airspace - identify every aircraft in the national airspace - gives alert and pursuit instruction about any non-identified flight	- manual (Strips) - communication devices - computer software

Table 2. Main ATC specialist Categories and their Tasks

1. Tower Controller Use Cases

The role of a Tower Controller (TC) is to watch over all plane movement in the airport's airspace. This includes all ground traffic (aircrafts and other airport vehicles) in the proper airport space, and all aircraft traffic during their final approach. The main responsibility of a TC is to organize the flow of aircraft into and out of the airport. Their tasks are highly related on the field of view from their position in the tower, and on the general traffic situation visualized in a radar screen. A tower controller should closely monitor each plane in the airport and in the final approach area to ensure a safe distance between all aircraft, give clearances for landing and takeoff, give directions to the pilots for any movement of the aircraft between the airport facilities (boarding are, hangar, maintenance area, etc...). The TC is also responsible for directing any other vehicle traffic within the airport area (such as boarding bridges, baggage tractors, baggage loaders, aircraft tractors, fuel trucks, etc.). Finally, the TC should keep pilots informed about changes in weather conditions such as wind shear (a sudden change in the velocity or direction of the wind that can cause the pilot to lose control of the aircraft), and other weather parameters such as the visibility, local atmospheric pressure, and clouds. The use cases of the TC are given in Figure 3.

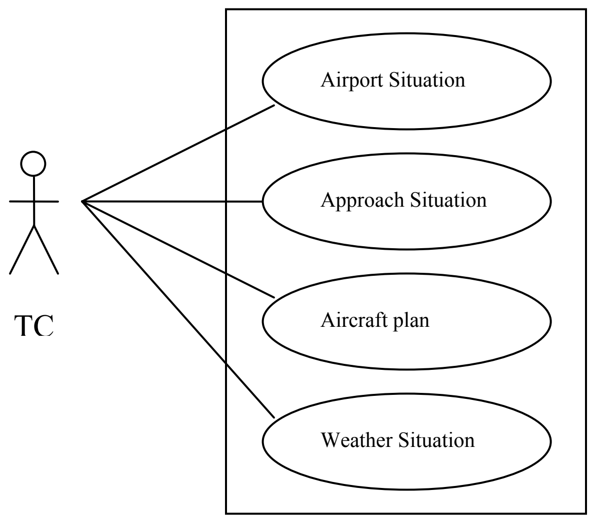


Figure 3. Tower Controller Use Cases

2. Approach Controller Use Cases

Aircraft are handed-off to the approach/TRACON controller either from the tower controller or from the en-route controller. To accomplish his or her tasks, the approach controller should have a graphical interface depicting a radar scope and different kind of communication systems. When it is appropriate, AC communicates with any aircraft ordering a change in aircraft parameter (altitude, speed, or heading). Figure 4 illustrates the use cases of the approach controller.

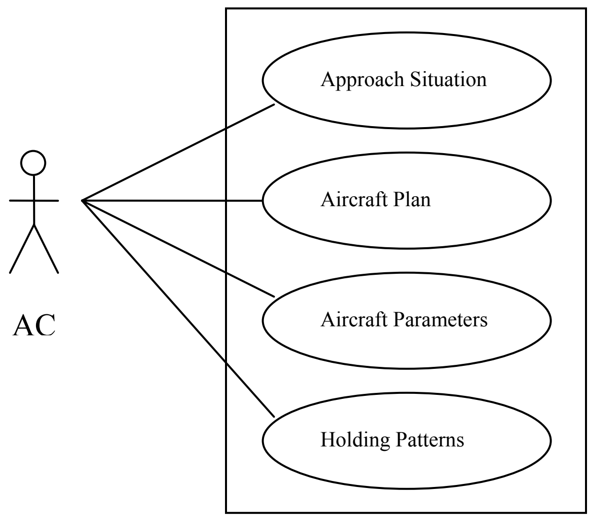


Figure 4. Approach Controller Use Cases

3. En-route Controller Use Cases

The en-route controller (EC) operates almost in the same conditions as the approach controller. In particular, the EC has a graphical user interface depicting one or more sectors of the air situation. Within this interface, the EC can visualize the route of each aircraft and can order each aircraft to change part or all of its flying parameters. In addition, the EC has of a set of navigation maps and can collaborate with other EC of other FIR in order to maintain a safe situation at all times. Figure 5 gives the use cases of the en-route controller.

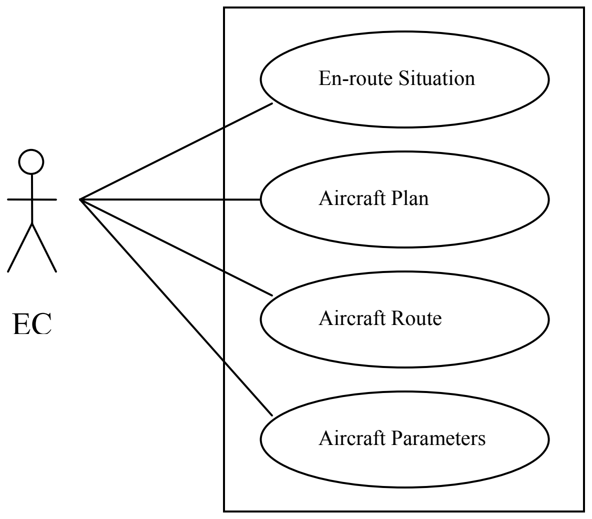


Figure 5. En-route Controller Use Cases

4. Pilot Use Cases

In most simulators, the pilot’s part is generally a pseudo-pilot or voice-recognition based pilot. In other terms, the pilot appears as a passive actor. In this project the pilot is an active part in the simulation, which make the overall simulation more realistic and the concept of team and teamwork more concrete.

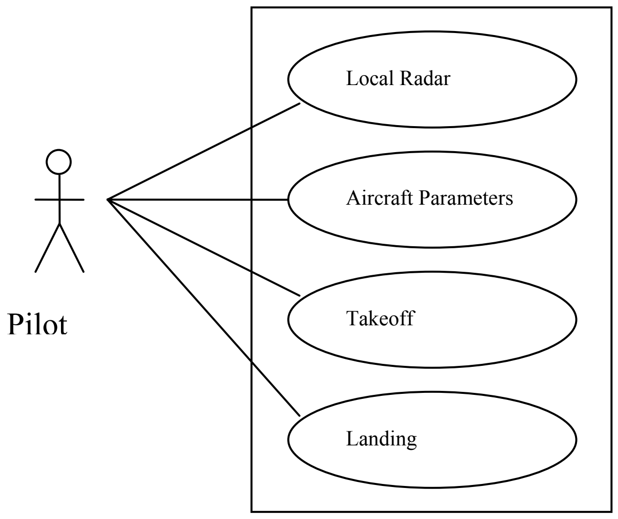


Figure 6. Pilot Use Cases

A modern aircraft cockpit has many instruments that aid the pilot in overall situational awareness of the aircraft and the airspace around it. However, since the scope of this thesis is to provide a synthetic environment that helps emphasis the teamwork concept, a completely realistic model of the cockpit is not be represented, instead there is a simplified model of the cockpit displays that best fits the requirement of the pilot's functionality and the teamwork capabilities. This interface will presents the aircraft's most important navigational instruments such as the altimeter, the speedometer, the heading indicator, and other flight parameters. In addition, the display will contain any necessary controls to execute any controller order. Figure 6 gives the use cases of the pilot in the simulator.

5. Instructor Use Cases

The instructor is the controller of the simulation exercise and oversees each student's progress. For these purposes he or she utilizes of a complete set of tools allowing him or her to follow all the simulation events. Figure 8 gives the use cases of the instructor.

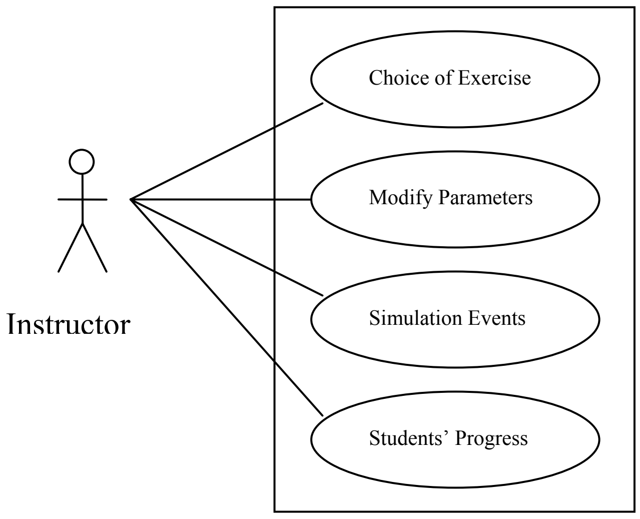


Figure 7. Instructor Use Cases

The main activities that the instructor requires from the system are shown in the use case diagram of Figure 7. Particularly, the instructor should be able to choose or modify any simulation exercise, have access to the simulation events as they are occurring, and to consult each student's progress database.

E. CONCLUSION

In this chapter, many concepts related to the air traffic domain have been illustrated. It was also shown that a complete air traffic control simulator contains five main categories of actors which are:

- tower control
- approach control
- en-route control
- pilot
- instructor

These concepts along with the definition and the use cases of each actor of air traffic control constitute the requirement of the simulation.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SYSTEM SPECIFICATION AND DESIGN

A. INTRODUCTION

In Chapter III, it was mentioned that a waterfall model (Figure 8) will be followed to develop the Teamwork Air Traffic Control Simulator (TATCS). The most important points of the requirement phase of this project have already been illustrated in the previous chapter. In this chapter we will continue defining and exploring the next step of the waterfall life cycle model which is called the specification phase. The specification phase constitutes a formal document that explicitly describes the functionality of the product, that is, precisely what the product is supposed to do along with any constraints that the product must satisfy. As can be inferred, the specification is an important phase in the sense that it constitutes a contract between the developer and the client or the intended user. This is why it should be accomplished with precision by exploring in depth what was mentioned in the requirement phase.

After completely specifying the different parts of the product, the next step will be the design phase. The design phase is a technical description of the specification. In other words, it describes how the product should be implemented to ensure the functionalities of the whole software. The design phase begins with the determination of the internal structure of the product which is a set of modules connected with each other. It is important in this phase to precisely specify the modules' interfaces, that is, the arguments passed to each module and the arguments returned by each one (Schach, 2002). This will help to gain insight about the whole project and help detect some design errors in the earliest steps of the software development life cycle. When an error is detected at this level it will not propagate to the other phases where any error correction becomes difficult and time consuming.

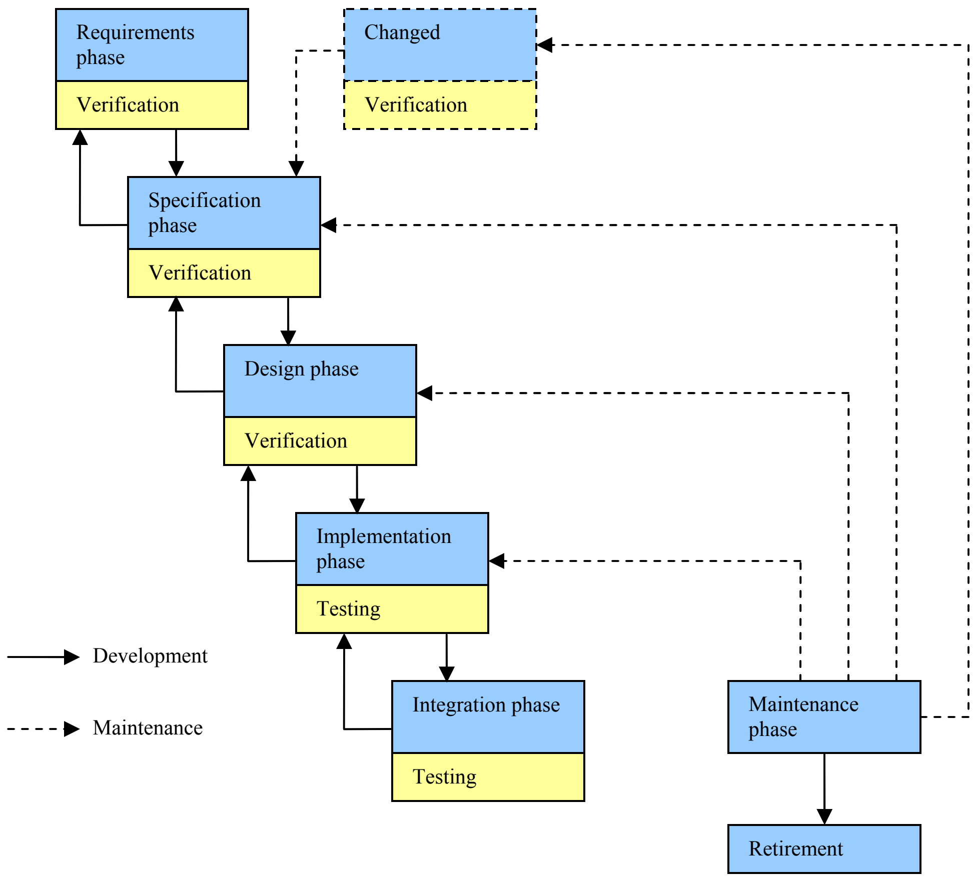


Figure 8. The Waterfall Life Cycle Model

B. SPECIFICATION

1. Specification for the Approach Part

Before going in depth of the approach specification, it is important to clarify two issues. First, the term approach control is used interchangeably with the term Terminal Radar Control (TRACON). This is because the controllers use local radar to control the terminal phase of the flight. Second, no distinction will be made between the simulator (as a software) used by the controller in the after takeoff, pre-landing phase, and in between the space in which the aircraft reaches its cruising speed and altitude. This is because the different controllers working in these phases usually use the same computer tools. Hence this thesis will specify only one simulation module for all these specialists.

The approach part used in the TATCS is a software module capable of visualizing simulated air traffic in the vicinity of the airport. The vicinity of the airport can be defined as a cylindrical volume around the central point of the airport. This volume may have different specification according to the airport location and constraints. However, for the purpose of this thesis it will be about 30 nautical mile (56 km) radius from the center of the airport and the height will be about 10,000 ft (about 3,050 m) from the surface of the airport.

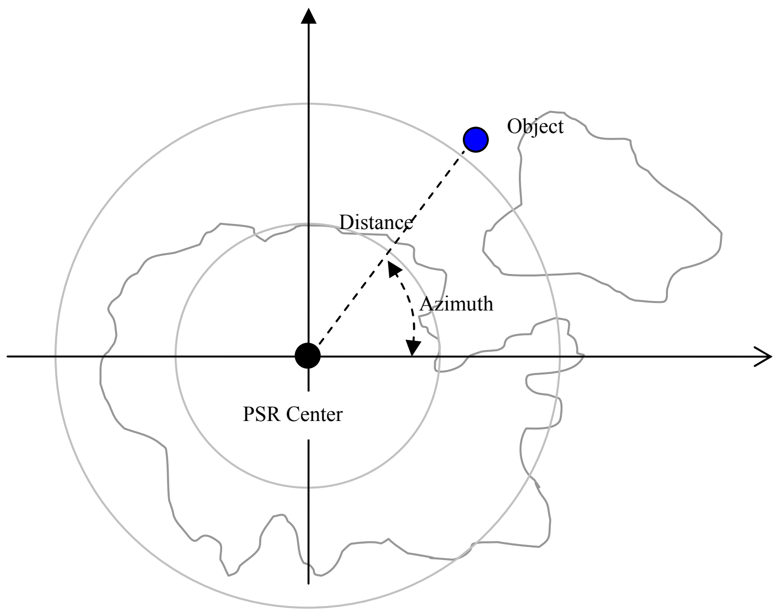


Figure 9.	Typical PSR output

To correctly visualize the different aircraft positions in this module, radar capabilities should be implemented in order to refresh the screen in the same manner as the real radar scope. In ATC, the main radar is called primary radar detection or Primary Surveillance Radar (PSR). Primary radar is a pulsed beam of ultrahigh frequency radio waves in a circle from a rotating aerial. If the radar beam 'illuminates' an object, some of the energy is reflected back. The amount of the reflected energy depends on many parameters such as the illuminated object's size, shape and material. The main output of the PSR is a spot representing the illuminated object which will be placed in the screen according to its distance and azimuth with respect to the center of the radar. Figure 9 shows a typical PSR output. The Primary radar is capable of detecting small or large aircraft, ULM, clouds, as well as passing birds and other non-desired objects. However, in this application, the PSR is supposed detecting only the aircraft of engaged in the simulation.

As can be seen, the PSR does not provide all the information needed by the ATC specialists in order to handle the air traffic situation such as the indicative, the altitude, the speed and other information regarding each aircraft. This can be done using the secondary radar or Secondary Surveillance Radar (SSR). The SSR is an antenna attached to the PSR that emits a special signal called interrogation signal to any illuminated object by the PSR. If the illuminated object is equipped by a special device called "Transponder", a special signal called "Response" will be sent from the transponder to the SSR. Once these responses are picked up by the secondary radar antennas they will be analyzed and processed electronically to be displayed on the screens of the air traffic controllers. The data displayed for the secondary surveillance radar includes the following information regarding the aircraft (see Figure 10):

- The flight ID
- The aircraft registration or the SSR code
- The aircraft flight level (FL)
- Aircraft's speed
- Aircraft's position

Another fundamental part in the approach module is the map in which the air situation is evolving. The map is important because it gives immediate visual cues about the position of all aircraft and the relative positions between aircrafts or between an aircraft and a landmark. There are a number of ways in which a map may be integrated into a TRACON application.

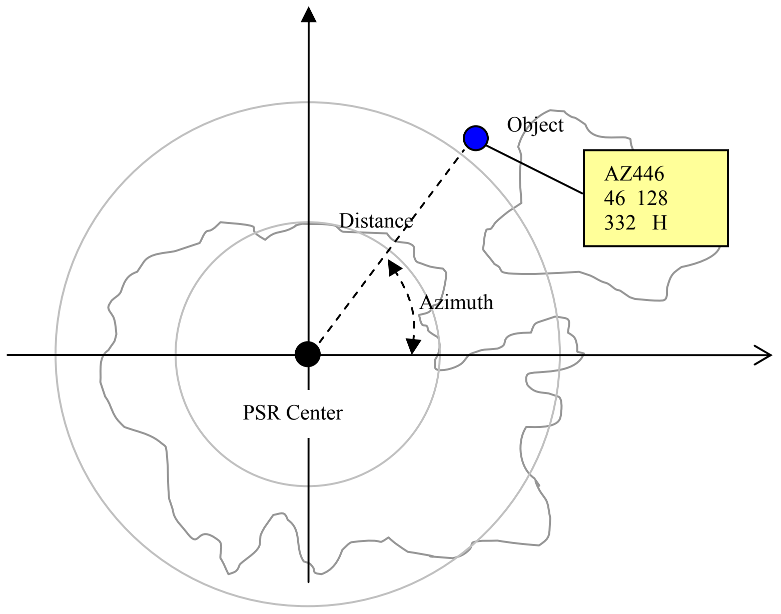


Figure 10. Typical SSR output

One obvious solution consists in using a bitmap map as a background image and scale all traffic to it. Even though this solution presents the advantage of being simple to implement, it is not useful for the purpose of this thesis. First of all, a bitmap image is a heavy component for an application that should frequently refresh an already complicated screen containing all air traffic symbols and the air traffic situation. Secondly, any scaling operation of the map (operation frequently used during a normal job of the ATCS) will produce a poor looking image in addition to the time consumption of the scaling algorithm that generally decreases the application performances. Finally, since the air Traffic GUI deals with human factor issues, it is generally difficult to change a map's appearance (such as the contours' colors and their intensity) to meet the controller's preferences. Further more a bitmap image will not allow the controller the possibility to select or hide the different map symbols that may create some visual confusion with the real air traffic.

A better solution to all the above problems consists in using a digitalized version of the bitmap image. This will allow the separation of all the map symbols in different independent categories. Each category, or layer, can be represented independently with its own appearance and can be stored within a data structure. In this way the user can add or change the properties of a layer whenever he or she needs without compromising the entire application performance. In addition, any scaling operation will be easy to implement without a sophisticated, heavy algorithm; it simply requires a multiplication with a scaling factor and a simple translation whenever the origin of the map is changed.

These layers can be easily represented using an XML (Extensible Markup Language) representation. XML is an industry-standard, system-independent way of representing data. The most important capability in XML resides in its portability, which allows data to be shared between different applications with little effort. All that is needed is an XML parser on the client side to correctly read the desired data file. Hence the map layers can be reused without much complication in any other application.

In conclusion, the specification phase of the approach module is the realization of a user-friendly interface that allows the controller the best view of the air traffic during the approach or takeoff phases. The following are the most important capabilities and characteristics of the approach module:

- A layered air traffic map of a desired approach area is depicted allowing the controller to see all sets of operations that are normally executed in any electronic map. These operations include, but are not limited to, the following commands:
    - scaling the map (zoom in, zoom out)
    - navigation
    - distance measurement
    - hiding or visualizing any map layer
    - personalized layers (by changing its appearance)
- Visualizing the air traffic data of the primary radar.
- Visualizing the air traffic data of the secondary radar.
- Changing the position of each aircraft SSR data in order to get a better visual situation of the air traffic.

- Refreshing the screen according to the radar revolution.
- Rapid access to the weather data in the approach zone.

2. Specification for the En-route Part

En-route ATC is a facility established to provide air traffic control service to aircraft operating on IFR flight plans within controlled airspace. This includes all traffic of class A (flying altitude is above FL 18 and FL 600) as well as Jet flight level (from FL 18 to FL 450). The en-route control and certain advisory or assistance services can be extended to VFR flights when equipment capabilities and controller workload permit.

An aircraft flying from point A to point B generally follows a well pre-established route defined in its flight plan. Each route can be viewed as a set of flight segments. The en-route control begins just after the airplane reaches the end of its takeoff phase and starts following its first route segment. The TRACON controller notifies the en-route controller of this event, who will then take care of that particular flight.

En-route controllers work in teams of up to three members. Depending on how heavy traffic is, each team is responsible for a section of the center's airspace. One team, for example, might be responsible for all planes that are between 30 and 100 miles north of an airport and flying at an altitude between 6,000 and 18,000 feet. These teams should collaborate with each other in order to guaranty a safe flight and good flying conditions for all aircraft under their control. In order to accomplish their task, en-route controllers utilize a highly sophisticated computerized radar system allowing them a complete view of the air situation. In addition, they maintain a two-way radio communication with aircraft in their controlling sector. These are the main devices used by the en-route controllers to ensure an adequate air separation between aircrafts at all times. Air separation for en-route controller is shown in Figure 11 below, in which the lateral separation is always 5 NM, and the vertical separation depends on the altitude of the aircraft. For aircraft below the FL 29, the vertical separation is 1,000 ft, and it is 2,000 ft for those flying above the FL 29 (Air Traffic Management System, 2005).

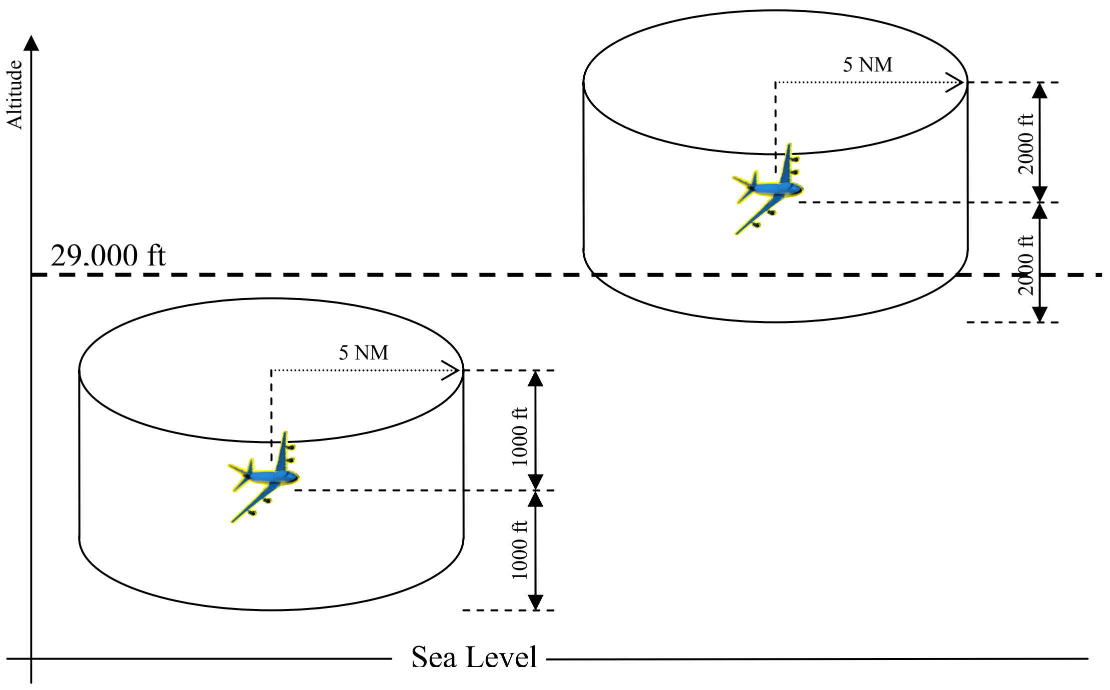


Figure 11. Aircraft separation in en-route control

The controllers can accomplish this separation by issuing instructions to the pilots of the aircraft involved. Altitude assignments, speed adjustments, and radar vectors are examples of instructions that might be issued to aircraft.

In conclusion, the en-route specification is another version of the TRACON specification with some additional capabilities. These capabilities include the possibility for the controller to obtain a large view of the air space with the possibility to query in real time the planned route for each airplane. In addition, the controller will have the possibility to visualize some other information on the map such as the Area of Minimum Altitude (AMA), the Flight Information Region (FIR) and all established routes in it with their respective entry point.

3. Tower Control Specification

The tower control is the part of the entire application dedicated to the ATCSs responsible for the ground and tower control. Even though there are many air traffic control specialists operating as airport controllers, no distinction between them will be made in realizing the tower control software. This is because the application can be used by any controller in the tower regardless his or her specific task. In addition, the main objective of this thesis is not to create applications for any specific air traffic controller, but to create a global application environment that allows all ATCSs to deal with the same air situation in order to increase the concept of teamwork among them.

In the application, the tower control part is a software module that can be integrated with the rest of the ATC modules, and can be operated by the tower control specialists. This module has the following characteristics and features:

- Visualization of the airport and its vicinity in 2d.
- Possibility to navigate easily in the airport space.
- Visualization of all aircraft paths in the airport.
- Real time rendering of aircraft and other vehicles' movement in the airport.
- Possibility of switching between day and night environments.

In addition to these characteristics, this module integrates some other capabilities such as the labeling of each moving object in the airport, consulting the weather database in order to inform the pilot about the airport conditions, and other traditional operation such as scaling the airport and personalizing the working environment.

4. Pilot Specification

The pilot part is an independent software application that can be integrated with the other parts of the entire application by some means of communication. As mentioned earlier, some other ATC simulators use a synthetic pilot, that is, a special software module, based on voice recognition technology that plays the role of the pilot in the whole ATC training loop. This works well for individual training purposes, or to accelerate the training period and help reducing procedural errors and expedite controllers' qualification, as mentioned in an article about modern training systems on the USAF Traffic Controllers (National Defense, 2006). However it is not suitable for an application in which the emphasis is on the fundamental concept of teamwork in which a human is in the loop in every part of the simulation. It can be argued that in real air traffic schools, it is hard to find real pilots to play the pilot part of the ATC loop. This can be true in pure air traffic control school. However, in many cases ATC training programs are part of a general aviation school, and in this case it is not hard to find some pilot students to play the role of the pilot as part of their training program. In addition, even in pure ATC schools, where there are no pilots on staff, it is important that the pilot part be played by an individual (some times called pseudo-pilot) instead of a software module. In such ways we can at least insure that the communications errors between controller and pilot are some kind of "human errors", and this is what we want to reduce by this work.

Returning to the fundamental specification of the pilot part, the pilot application consists of a GUI from which the pilot or the pseudo-pilot can easily visualize the most important data regarding any aircraft participating in the air traffic exercise. The pilot has a list containing some assigned aircraft (identified by their flight ID) from the exercise. By clicking on a particular flight ID, the pilot's work station automatically switches to the pseudo-cockpit of the related aircraft. It is called pseudo-cockpit because it is not as complicated as the cockpit of a real aircraft. However it contains the most important information that a pilot needs to have awareness about that particular flight, and to correctly execute the orders of the controller. This information includes all data about that particular flight such as aircraft type, heading, speed, SSR code, etc. In addition, there are the most important navigation instruments of the aircraft: the altimeter, the speedometer, and the cap indicator. These graphical instruments help give the pilot a quick and accurate vision about the aircraft situation. The GUI also contains some other buttons that can be manipulated to create an emergency event in the aircraft (communication failure, fire, and other anomalies) according to the instructor directives.

5. Instructor Specification

The instructor application is another independent module allowing the instructor to gain full control of a particular air traffic control exercise. The instructor application is loaded into a particular workstation, called "the instructor workstation". From this workstation, the instructor has access to all possible settings for any air traffic control exercise. These settings include the following operations:

- The choice of the ATC exercise that will be applied to all application's workstations.
- Enabling the time period of the exercise (day or night).
- Changing some flight parameters of an exercise in real time. These changes will be updates automatically in all workstations.
- Keeping track of all controller progress in a particular exercise. This is done by implementing a database containing records for each trainee. Each record contains, other than the trainee identification, the starting and the end of his or her session, the start and end of each air warning (AW), and the start/end of each air conflict period.
- Keeping track of all trainees' progress during their entire training program. In such way, the instructor can consult at any time the evaluation history for each student.

In addition to these possibilities, the instructor can collaborate with the pilot or the pseudo-pilot to insert any emergency that may be encountered during a normal flight. These emergencies include communication failure, a fire aboard the aircraft, a hijack situation of the aircraft, or any other situation in which the pilot needs particular care. Any of these setting will be reflected in all workstations instantly and all controllers will see a particular SSR code that reflects that emergency. For example if the pilot is losing control of his or her aircraft, an SSR code of 7700 denoting 'Mayday' will be displayed in all controllers' workstations. Table 3 gives a summary of all aircraft emergencies and their relative SSR codes (ICAO, 2006) used in this application.

SSR code	**Use**
0000	Available as a general purpose code for local use by any State.
1000	Reserved for use as a conspicuity code for Mode S.
2000	Used by flight crew in the absence of any ATC instructions or regional agreements unless the conditions for the use of codes: 7000, 7500, 7600 and 7700 apply.
7000	Used by flight crew not receiving ATS service in order to improve detection of suitably equipped aircraft in areas specified by States, unless otherwise instructed by ATS.
7500	Reserved for use in the event of unlawful interference.
7600	Reserved for use in the event of radio communications failure.
7700	Reserved for use in the event of emergencies and interception.
7776	Reserved for SSR ground transponder monitoring.
7777	Reserved for SSR ground transponder monitoring.

Table 3. Special purpose SSR codes and their uses

C. APPLICATION DESIGN

1. Overall Application Design

As mentioned in the requirement and specification phases, the overall application consists of five modules:

- The Approach Module
- The En-route Module
- The Pilot Module

- The Tower Module
- The Instructor Module

These modules are connected via a communication layer to ensure that all information is synchronized among them. In addition, a server application keeps track of all air traffic exercises and the different students' outputs. The overall design of the application is shown in Figure 12. It can be seen that the different modules interact with each other by the means of the communication layer. Note that the controller can change any the local workstation settings, but will not have any privileges to change any exercise data such as aircraft status. All the controllers can do is sending a voice order to the pilot, who will execute the order. Although the vocal infrastructure is not implemented in this thesis, it is part of the normal equipment that is normally used by the controller to communicate with the pilots.

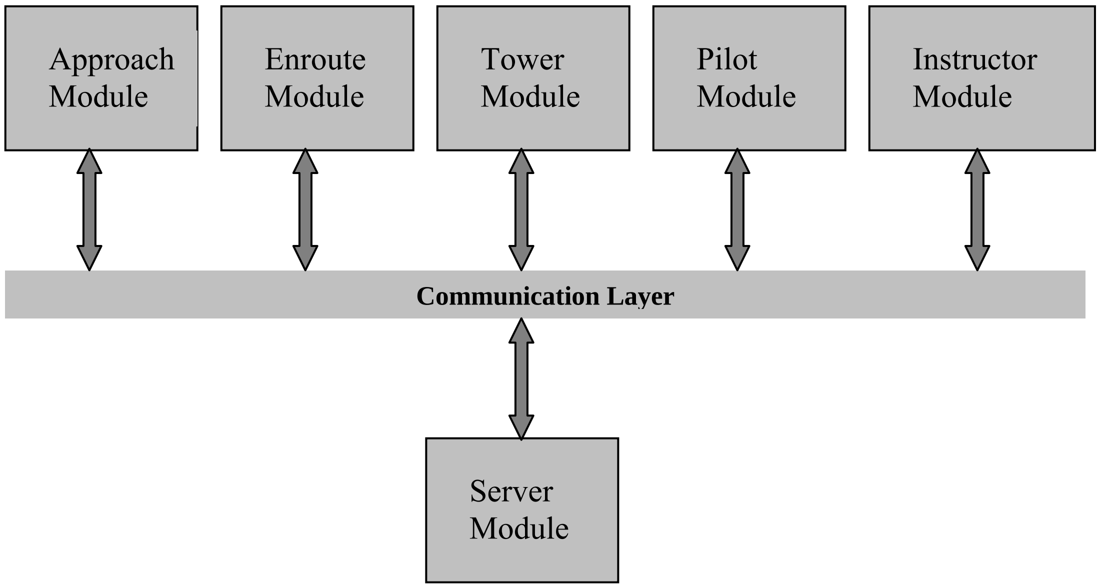


Figure 12. The overall application design

The design of the overall application in Figure 12 shows that the different modules are independent entities that can be easily implemented using the Object

Oriented paradigm. The only means of interaction between these modules is the communication layer which is also implemented using the object oriented techniques. Before examining in depth the design of each module, it is convenient to take a look at the communication module since it is the basic one.

2. The Communication Module

As mentioned above, the communication module (CM) insures the interaction between the different modules of the application. This module is designed to support the particular constraints of the entire application and its different modules. The communication module acts as middleware between the different modules. In fact the data representation can be heterogeneous between the different modules, and the CM should be the place in which each piece of information is "packed" according to the needs of the destination module. Secondly, all the modules of the application are remotely located to each other and can use a wide range of network protocols as a physical means of communication, and the CM should be architected to support this network's diversity. Thirdly, there is no guaranty that the future modules will operate on the same platform using the same operating system. Finally, different implementations using different programming languages can be envisaged and the CM should be able to work correctly with those implementations. All these constraints can be solved at least in part using a Common Object Request Broker Architecture (CORBA).

CORBA is the Object Management Group's answer to the need for interoperability among the rapidly proliferating number of hardware and software products available today. CORBA allows applications to communicate with one another no matter where they are located or who has designed them. CORBA is not a language, but a specification for how objects will interact. Thus, it is not limited to a single language. CORBA services and clients can be written in a variety of languages such as Java, C++, or ADA. CORBA is made up of a collection of objects and software modules that cooperate together in a networked environment. The main component in the CORBA architecture is the Object Request Broker (ORB).

The ORB provides a mechanism for transparently communicating client requests to target object implementations. The ORB simplifies distributed programming by decoupling the client from the details of the method invocations. This makes client requests appear to be local procedure calls. When a client invokes an operation, the ORB is responsible for finding the object implementation, transparently activating it if necessary, delivering the request to the object, and returning any response to the caller (CORBA & RMI, 2006). In other words, the ORB acts as a middleware handling all requests from the client's side about an object's services located at the server side, as well as sending the client's responses to some queries from the server. Figure 13 provides an illustration in how client and server interact with each other using the ORB.

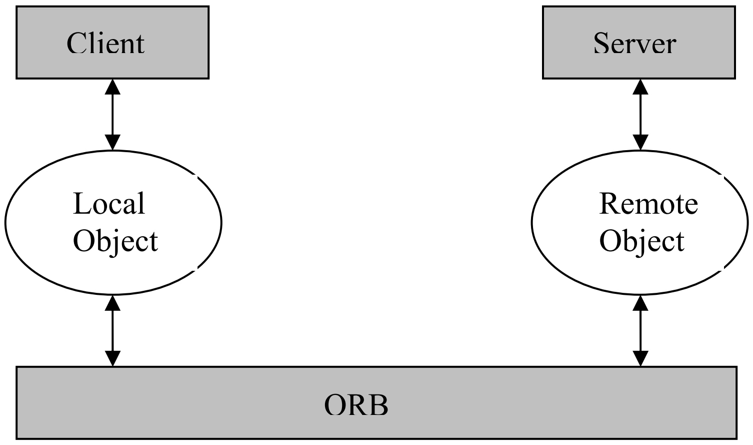


Figure 13. Client/Server interaction in CORBA

Even though remote objects are treated as local objects using a proxy (by invoking their methods as if they are in the client site), there is communication between the two objects' representations. This communication is supervised by the ORB via the Internet Inter-ORB Protocol (IIOP) as shown in Figure 14 (Reilly & Reilly, 2002).

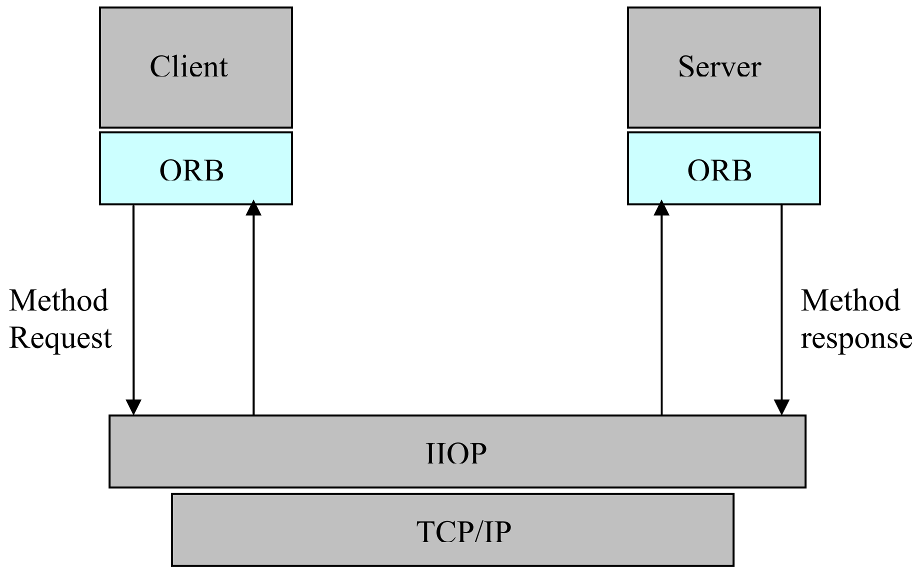


Figure 14. CORBA communication architecture

Within the CORBA architecture, software services are described by a schema and implemented by a servant, a special piece of software that registers itself with a lookup service so that other CORBA software can locate and access its services. Typically, a CORBA server will create a CORBA servant, and is then responsible for creating an ORB for the servant and registering the service for access by other clients (Reilly & Reilly, 2002). All registered services in CORBA architecture will be kept in a special registry known as CORBA Name Service.

As previously mentioned, a CORBA-compliant application can be implemented using a wide variety of programming languages. However, since all modules of this application will be implemented in Java, it is convenient to look at a Java implementation of the distributed objects architecture, a Java package called Remote Method Invocation (RMI). RMI enables the programmer to create distributed Java technology-based to Java technology-based applications, in which the methods of remote Java objects can be invoked from other Java virtual machines (JVM), possibly on different hosts. A Java object make a call on a remote object once it obtains a reference to the remote object, either by looking up the remote object in the bootstrap naming service provided by RMI or by receiving the reference as an argument or a return value. A client can call a remote object in a server, and that server can also be a client of other remote objects. RMI uses object serialization to marshal and unmarshal parameters and does not truncate types, supporting true object-oriented polymorphism (CORBA & RMI, 2006).

In the RMI architecture, the developer designs servers with some specified services available for clients. These services should be named (using common strings), and registered in a special independent process called “rmiregistry”. The client needs these names to lookup at their related services in the registry. In RMI the server does not have to be located on any specific machine. It can be activated from any computer in the network in a complete transparent way to the client which does not have to change any parameter in his or her application. Other essential components in the RMI architecture are the stub and the skeleton for each service offered by the server (Figure 15). Both components are obtained by a special compilation, called RMI compiler, which accepts as arguments classes that have been compiled successfully with the JVM compiler.

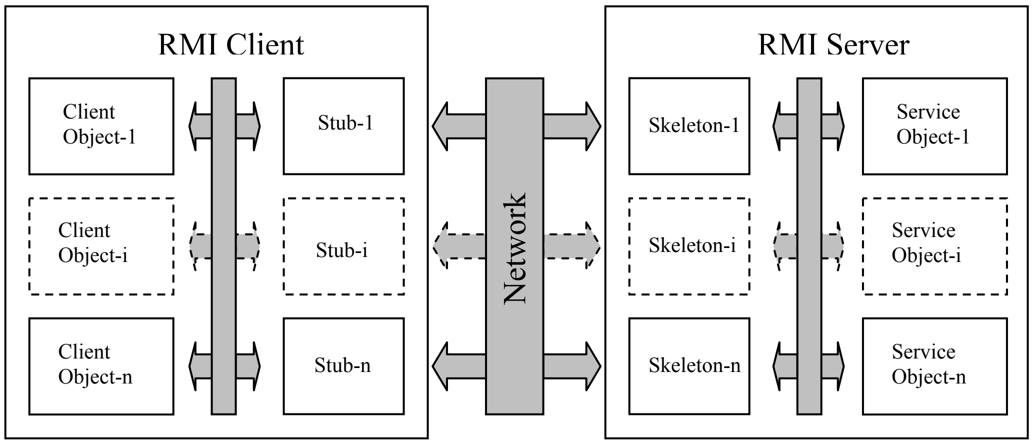


Figure 15. RMI Client/Server architecture

The stub objects each implement a remote interface and are responsible for marshalling and unmarshalling the data and managing the network connection to the server in a way totally transparent to the client. So stubs act like proxies to the client, and an instance of them is required on each client. Whenever a client object invokes a method on the remote object, the call will be directed to the stub, and a service request will be sent on the network to the corresponding skeleton object. The skeleton object is located on the server side and acts as a listener for incoming service requests.

In this work, two main RMI services will be provided to any module of the application. The first service, the most important, is related to the air situation. At any moment, any client module can ask about the number of aircraft actually engaged in the simulation exercise, as well as other related information such as the speed, the heading, or the altitude of any desired aircraft. In addition, this service includes all the commands that can be given to an aircraft such as changing altitude, or turning left or right to face a given heading. Operations such as the timing of the insertion of an aircraft in the exercise or eliminating it from the exercise are reserved for the instructor who acts from the server side. Therefore there is no need to implement them as services; they can be directly invoked by the instructor as local methods.

The second service offered by the RMI server is the user's service. This service is responsible for the identification of any user of the system, and keeps track of the different user's activities. In this application, each user has a name and an ID string. When the user first connects to the server, he or she will be asked to enter his or her name and ID string. The local application will automatically add to this information the URL identification of the current workstation. After that, an identification request will be sent to the remote server via the user's stub object, and the user can then continue using his or her module if the answer was positive. The other important operation offered by this service is the ability to register all users' activities. This is done by the local application and is transparent to the user. Every time the user performs any operation via the network, the local module will send an event registration request to the user's service, and that activity will be saved in the server database for future consultation and analysis. This service therefore acts mainly as an event listener for the clients' application, and event delivery on the server side as shown in Figure 16.

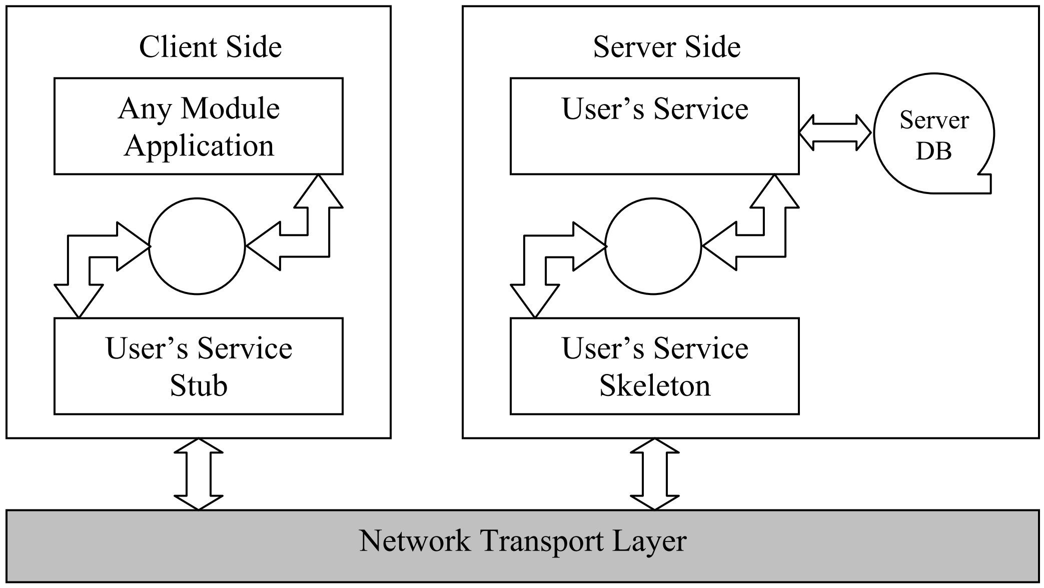


Figure 16. The ATC user service

3. The Instructor Module

The instructor is the local user of the simulator server. He or she has access to all resources of the exercise and can also see in real time the different trainees' activities. To facilitate his tasks, the instructor module has an intuitive GUI that is easy to use and contains all the necessary activators. These activators are implemented as menu items and as action buttons on a specific toolbar.

The instructor module is divided into three graphical interfaces. The first one contains all the basic operations for the server, such as starting or stopping the different services, and other operations related to the kind of exercise that will be RMI-broadcasted in the network. The second module is a list of all aircraft participating in the exercise. From this interface, the instructor will have access to the "global aircraft commands". These commands include starting or stopping any aircraft, and removing aircraft from the exercise. This will help to implement many different kinds of exercises based on the initial loaded exercise file. This interface allows the instructor to customize the exercises in a manner that best fit the level of the trainees. Finally, the third interface will allow the instructor to control the different students' activities. These activities are the output of the event handler described in the previous paragraph. An example of what the instructor might see in this interface is depicted in Table 4 below.

Date	03/26/2006
Exercise	Fp123
Instructor	John Smith
Starting	10:05:00
Stopping	11:30:22

Time	D	Qualif.	URL	Event
10:05:55	s01	Pilot	130.121.99.50	AZ340 TL 90
10:10:40	a04	En-route	130.121.99.51	AW AZ340 TU787
10:10:55	s01	Pilot	130.121.99.50	AZ340 L 280
10:50:05	c05	Tower	130.121.99.30	AS234 LD 11
11:03:28	a02	Approach	130.121.99.40	AC TU423 AF200

Table 4. A typical instructor's event display GUI

In Table 4, the different columns are all self-explanatory except the "Event" column. Each event is described using a pseudo-language. For example, the event "AZ340 TL 90" means that the aircraft with flight ID AZ340 received an order to turn left (TL) and faces the heading 90 degree (see Figure 17 for the heading interpretation adopted in Air Traffic). The event "AW AZ340 TU787" means that there is an air warning between AZ340 and TU787. The field of such event is colored in yellow to alert the instructor. An air conflict (AC) event is a more dangerous situation and is colored in red as shown in the last row of Table 4. Table 5 gives a summary of all symbols of this pseudo-language. Note that an air warning or an air conflict is raised by the controller to the pilots if a group of aircraft is going to violate separation minima considering their current or future situation. These separation minima are generally subjected to changes by the ICAO. In the application's design, these parameters will be read from a data file and the instructor can change them at any desired time.

Symbol	Interpretation
AW x-1 x-2... x-n	Air Warning among the aircraft x-1 x-2... x-n
AC x-1 x-2... x-n	Air Conflict among the aircraft x-1 x-2... x-n
TL x	Turn Left to face x degree
TR x	Turn Right to face x degree
L x	Change to Level x
LD x	Landed in runway x
TK	Takeoff from runway x
S x	Change Speed to x Knots/Hour
SSR x	Change in SSR code to x

Table 5. Interpretation of the pseudo-code used in the Instructor GUI

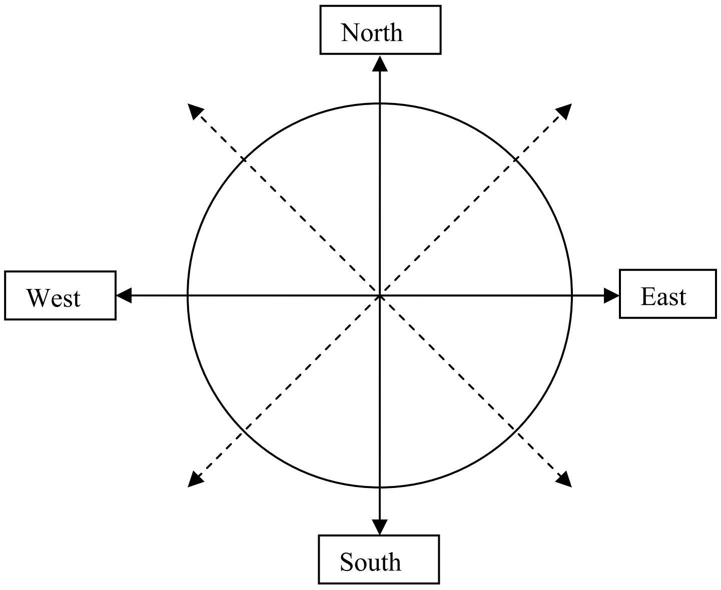


Figure 17. Headings in ATC

4. The Approach Module

The approach module has a GUI that helps the controller gain good situational awareness of the air traffic that he or she is controlling. Since the controller spends most of his or her time manipulating this GUI, it is crucial that the controller can personalize this GUI in any desired manner. For this reason, as mentioned in the specification part of this module, all data regarding the GUI is stored in XML format within a particular directory of this module (the data directory).

The reason for the use of XML files for all the data of the simulator is not only in the portability and interoperability of XML, but also in the possibility to “extract” real Objects from an XML file. That is, it is possible to add some accessory methods to these objects such as changing their color, or their position. As a result the user can manipulate all aspects of this object with the minimum computational cost. In addition, since the instructor can theoretically run a simulation in any geographic region, the software includes a tool to extract XML information from a given map that can be easily used in the ATC simulator.

The overall design of the Approach application is shown in figure 18. The first step that the application does is load the map, navigation data, and any other symbols used in ATC from a package containing the related XML files. Since the meta-data enclosed in these files aren’t the same, there is a different XML parser for each category of data. The next step consists in passing the parsed data to the “Updater Thread”. This class is the core object of the approach application. It first constructs a collection of objects based on this data, and then provides an update of the “Approach Frame” which is a simple GUI to display and manipulate graphical symbols. Among the graphical symbols that are displayed are the aircraft representations. However these reside in the server part and the only way to access them is via the server’s aircraft services. For this reason, the updater uses one stub to access this service, and another stub to access the user service for user identification and event handling purposes, as explained in the previous paragraph.

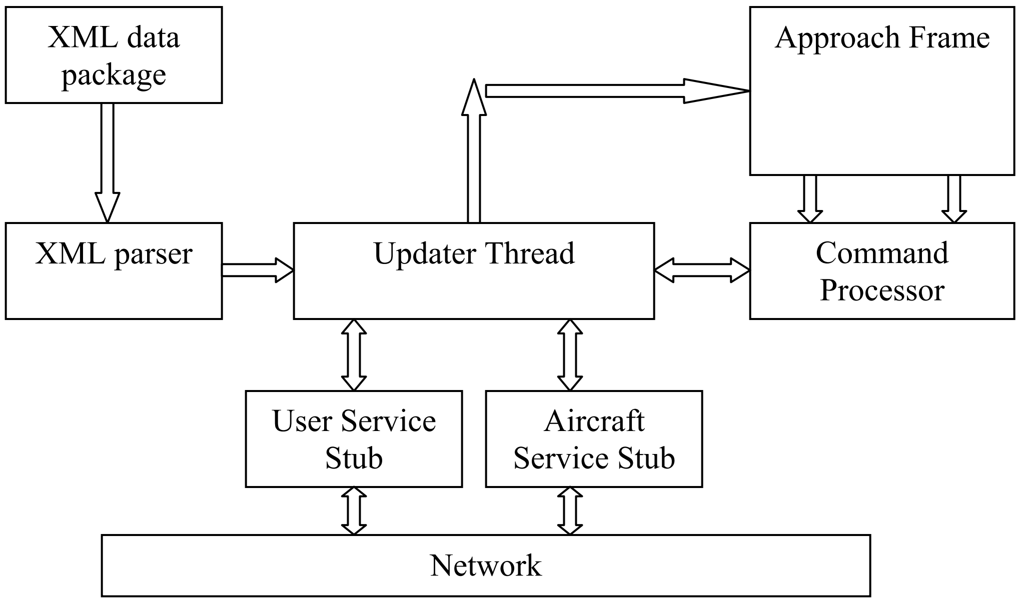


Figure 18. Overall approach module design

The approach frame contains all the necessary tools that allow the user to interact with it. When the user issues a command, it passes to the "Command processor" which plays the role of command interpreter for the updater thread. At this point, the updater thread will prepare another screen containing all previous data and the results of the user commands, and will finally provide an update of the approach frame. In this way the graphics- heavy interface of the approach frame is made easy to draw and update. In fact it has only to draw the same image size, regardless of the quantity of information on it, at every refresh rate (depending on the radar revolution). Furthermore, the approach frame will not spend any additional computation once it is launched, since all image preparations are made off-screen by the updater thread.

The design of the approach module is versatile and can be applied as it is or with small modification for the en-route module. In fact the main difference between the two modules resides in the visualization and not in the functionality. Such differences will be directly detailed in the implementation phase.

5. The Tower Module

The tower module is a 2D environment representing an airport and its vicinity which we denote with “tower working zone”. Within this zone the tower specialist has the task to control the activities of all moving vehicles in order to insure safety and reasonable aircraft flow from and to the airport. All the information regarding the tower working zone will be parsed from an XML file. Thus, there is complete independence between data representation and information processing. For the purpose of this thesis, only two tower working zone will be presented; the international airport of Tunis-Carthage (DTTA), and the international airport of Monastir (DTMB).

In this design, the tower working zone is a collection of objects including the runways, the taxiways, and the stationing area presented in an adequate GUI. This GUI offers to the user (the tower controller) the following capabilities:

- Easy navigation.
- Showing or hiding airport data.
- Scaling the airport.
- Centering the viewpoint.

However, this module cannot interact directly with an aircraft (for example to change its position or heading). Interaction with an aircraft will only be allowed via a voice communication to the pilot module, examined in the next section.

6. The Pilot Module

The pilot module is an independent module of the ATCS offering to the pilot all the necessary tools to control the aircraft situation during flight conditions or whenever the aircraft is in the ground. Since the operation conditions of the aircraft in flight or ground are not the same, two sub-modules are included within this module. The first sub-module mainly controls the aircraft during flight conditions. It contains a pseudo-cockpit depicting the main cockpit instruments which are the speedometer, the altimeter and the heading indicator. In addition to these instruments, and in order to make the pilot gain situation awareness about the surrounding traffic, a radar screen is provided. Finally, a panel of buttons is also added to this module. This panel will be used by the pilot under the order of the instructor to cause an abnormal event in the aircraft as specified in the specification part.

The second sub-module exclusively controls the aircraft in the ground (once the aircraft is in the airport). The operations that are allowed in this module are the following:

- Parking the aircraft in a given position.
- Taxiing the aircraft to a point specified by the tower controller.
- Positioning the aircraft in the runway.
- Make the aircraft takeoff.
- Make the aircraft landing.

The overall design of the pilot module is shown in Figure 19 which shows the complete independence between the two sub-modules.

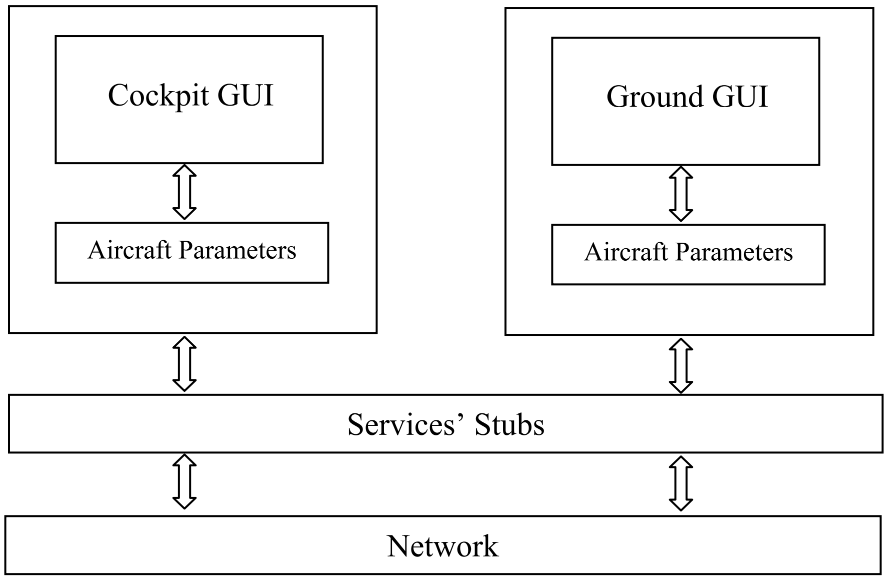


Figure 19. The Pilot Module Design

D. CONCLUSION

In this chapter a detailed specification for each of the five modules constituting the air traffic control simulator are given. During the specification phase, some concepts related to air traffic control, such as the separation volume, SSR code, and heading convention, are explored and made clear. Also, since the communication module is an important link between all modules of the application, an ample space is dedicated to it. Particularly, many concepts of RMI architecture is given.

As specified by the waterfall model, these specifications are the input of the design phase with is the argument of the second part of this chapter. In this phase, each module is detailed in depth and, for each module, an architectural design is given. This helps to ease the implementation phase which is the topic of the next chapter.

V. IMPLEMENTATION OF THE ATC SIMULATOR

A. MAP REPRESENTATION

All the graphics representations in this application are vector images extracted from real maps or real spatial photos. The module named “MapEditor” allows the user to extract the data from an aeronautical map. The user has only to click the mouse on contours of that map to create a vector of points corresponding to those contours. These points are then saved to an XML file. Figure 20 illustrates the module “MapEditor” and Figure 21 gives the XML schema of the saved data.

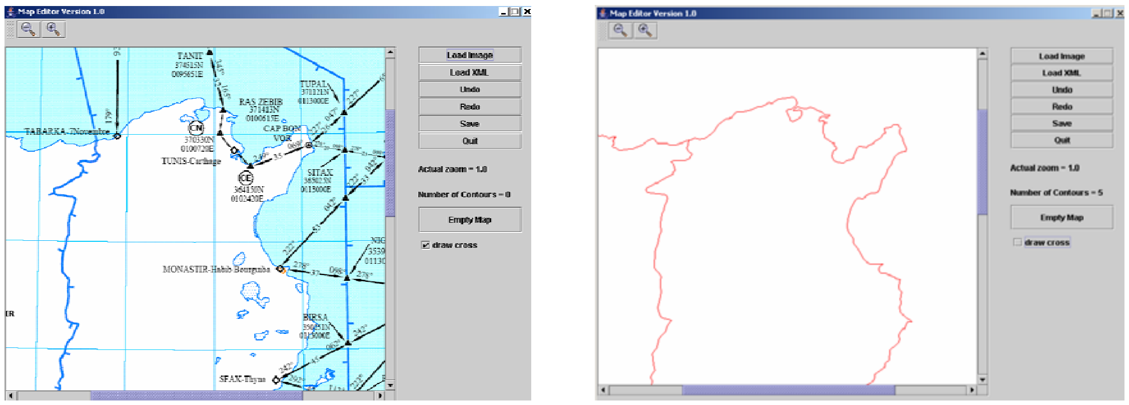


Figure 20. Bitmap and XML representation of the map

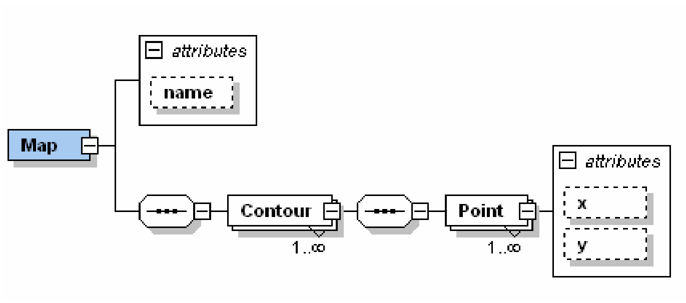


Figure 21. The XML schema of a map

Once the data is prepared, it can be used by the approach and en-route modules. Note that each xml file containing map data is viewed as a map layer which will be stored in an appropriate data structure within the latter modules. These layers can be easily accessed by the user either to show or hide them or personalize their appearances. Figure 22 shows the approach GUI with a special control panel allowing the user to select the layers that he or she wants to visualize. The use of the control panel is very intuitive. When a layer is selected, the corresponding button will be highlighted; otherwise it is in a dark color.

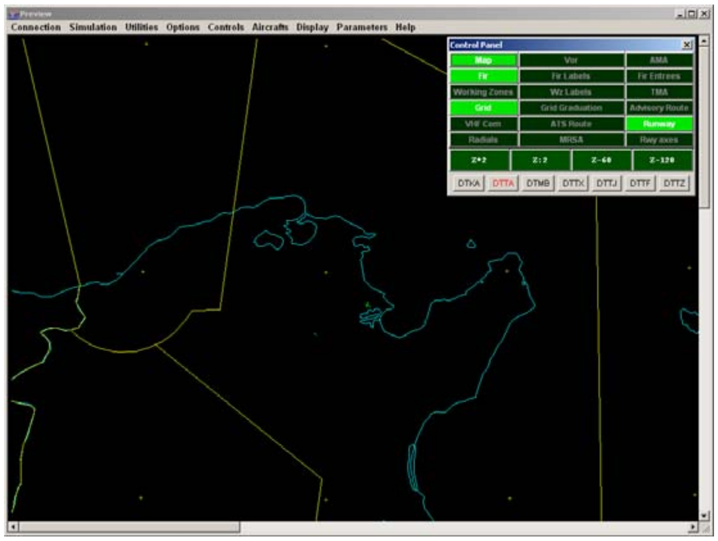


Figure 22. Visualization of the XML Data in the Approach and En-route Applications

B. THE SERVER/INSTRUCTOR MODULE

As stated in the previous chapter, the RMI server module and the instructor module constitute a single package that can be accessed and manipulated only by the instructor. The main reason behind this is to simplify the implementation of these two modules. The overall UML diagram of this module is shown in Figure 23. The main remote service offered by this module is the aircraft service (see Appendix B for a complete services' listing). This service is represented by the class "AcVector" which is the implementation of the interface "IAcVector". Particularly, the latter interface offers a set of methods to get or set any parameter of the aircraft such as the speed, the altitude, the cap, and etc. All services in this module are made available to any other module via the class "RegistryStarter". Since a reference of this class exists in the main class "ATCServer", the instructor will be able to start or shutdown any service at any moment if required.

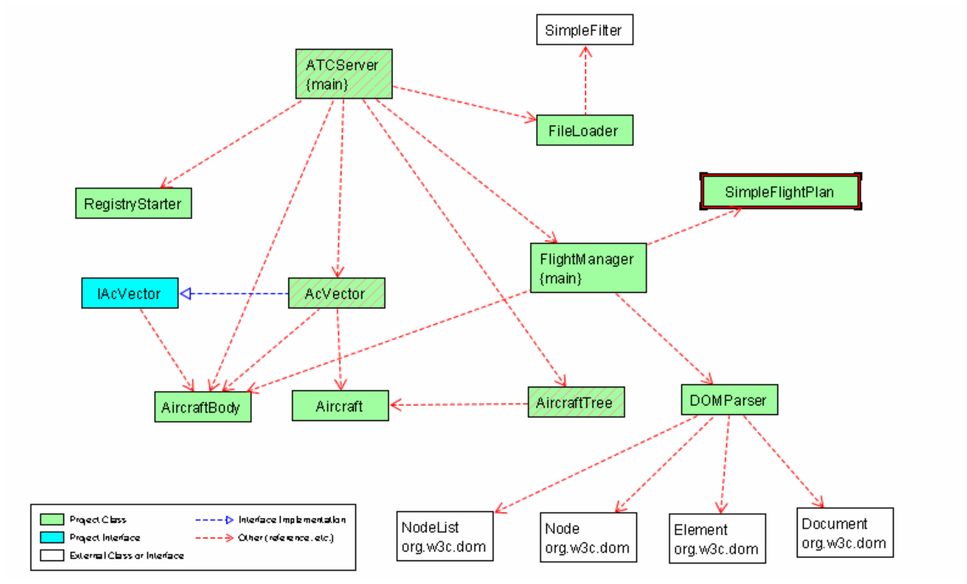


Figure 23. UML diagram of the server module

All the flight plans are stored in a specific directory within this module package (data directory). A Document Object Model (DOM) parser named "DOMParser" is used to parse the flight plans' files. Each of these files is a valid XML document according to the XML schema of Figure 24. Once they are parsed, the "DOMParser" object constructs an array of instances of "SimpleFlightPlan" class which is passed to the main class in order to build up the remote service class (AcVector).

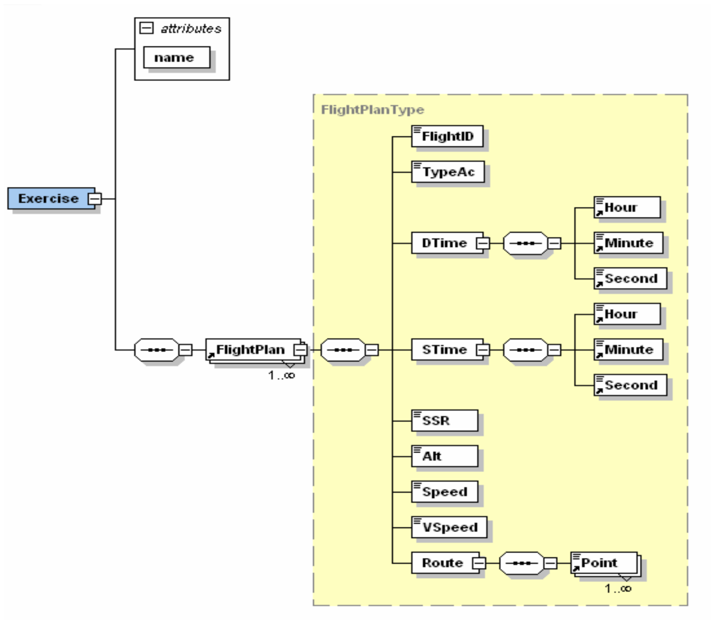
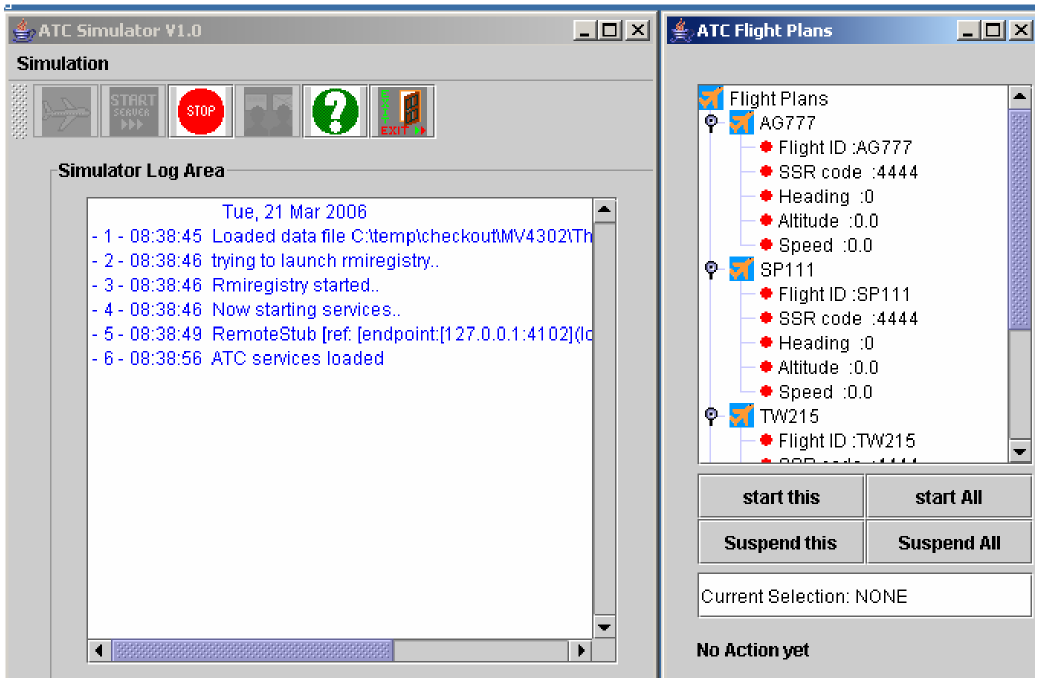


Figure 24. XML schema of the flight plan

The server module presents a user-friendly interface (see Figure 25 below). The instructor has only to click few buttons to make the services available to the other module. The instructor first selects the exercise to play and then clicks on the "Start Services" button. He or she can then start all aircraft of the exercise at once or only select a group of them to be part of the current simulation.



```
Tue, 21 Mar 2006
- 1 - 08:38:45  Loaded data file C:\temp\checkout\MV4302\Th
- 2 - 08:38:46  trying to launch rmiregistry..
- 3 - 08:38:46  Rmiregistry started..
- 4 - 08:38:46  Now starting services..
- 5 - 08:38:49  RemoteStub [ref: [endpoint:[127.0.0.1:4102](lo
- 6 - 08:38:56  ATC services loaded
```

```
Flight Plans
  AG777
    Flight ID :AG777
    SSR code  :4444
    Heading  :0
    Altitude  :0.0
    Speed  :0.0
  SP111
    Flight ID :SP111
    SSR code  :4444
    Heading  :0
    Altitude  :0.0
    Speed  :0.0
  TW215
    Flight ID :TW215
```

Current Selection: NONE

Figure 25. The server module GUI

C. THE PILOT MODULE

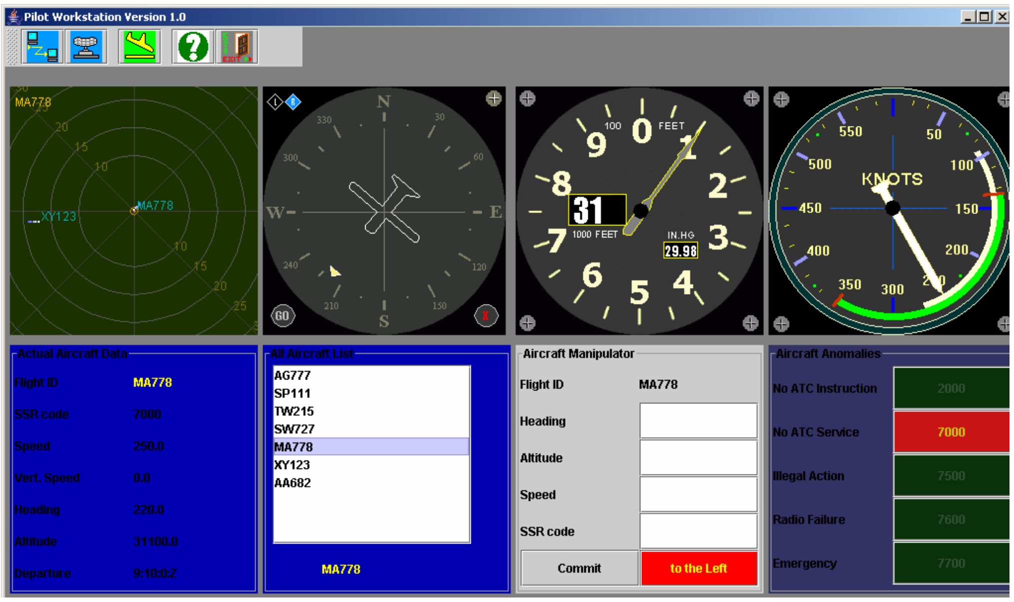
The pilot package contains two distinct modules. The first one represents the aircraft cockpit, and the second one is the ground module. The cockpit module in normally used to control the flying aircraft, and the ground module is only used to manipulate the aircraft within the airport. The following paragraphs describe in depth each of these modules.

1. The Cockpit Module

The cockpit module, shown in Figure 26, allows the pilot to control the aircraft once it is flying. This module is a GUI containing all the display and control devices that the pilot needs to gain situational awareness about the current flight and the overall air situation. Particularly, the first row of this GUI presents the most important devices that help the pilot obtain a quick look assessment of the situation and the flight parameters of the aircraft. The second row contains more parameters about the flight, and allows the pilot to interact with the aircraft.

The pilot can switch between one aircraft and another by simply clicking on the desired aircraft flight ID, and instantly the GUI will change to that aircraft cockpit. When the pilot received an order from any controller, he or she can execute it by typing the new flight parameters in the appropriate fields of the aircraft manipulator panel. In addition, when directed by the instructor, the pilot can simulate the most common anomalies in an aircraft by clicking in the appropriate button of the anomalies panel situated in the right lower corner of the cockpit GUI.

This module, as all other client modules of the simulator, has a stub for each remote service. These stubs are used by a thread named “ControllerThread” to update the aircraft situations. At every loop, this thread checks the server for an update in the aircraft list and for an update for each of the aircraft situation. The new list with the new aircraft situation is then passed to the main class which provides an update about the different devices and panels of the GUI. The complete UML diagram of the cockpit module is shown in Figure 27. In this diagram, it is easy to see that each panel of this module has its own independent class, and that the ControllerThread has a reference to each of these classes.



Actual Aircraft Data

Flight ID	MA778
SSR code	7000
Speed	250.0
Vert. Speed	0.0
Heading	220.0
Altitude	31100.0
Departure	9:10:0:Z

All Aircraft List

AG777
SP111
TW215
SW727
MA778
XY123
AA682

MA778

Aircraft Manipulator

Flight ID	MA778
Heading	
Altitude	
Speed	
SSR code	
Commit	to the Left

Aircraft Anomalies

No ATC Instruction	2000
No ATC Service	7000
Illegal Action	7500
Radio Failure	7600
Emergency	7700

Figure 26. Cockpit GUI of the pilot package

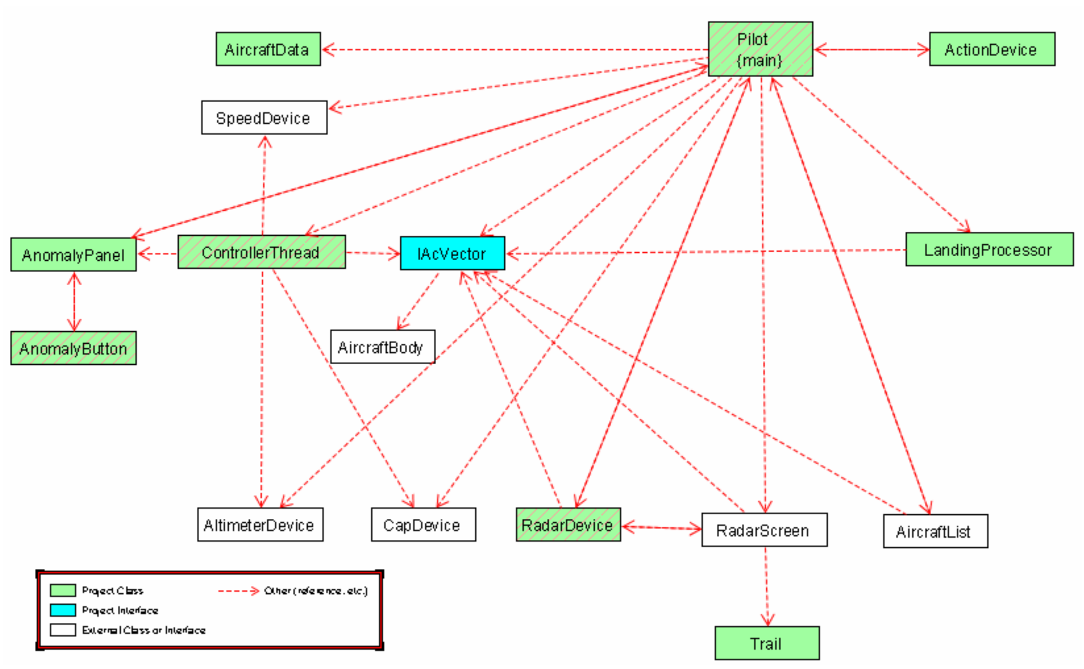
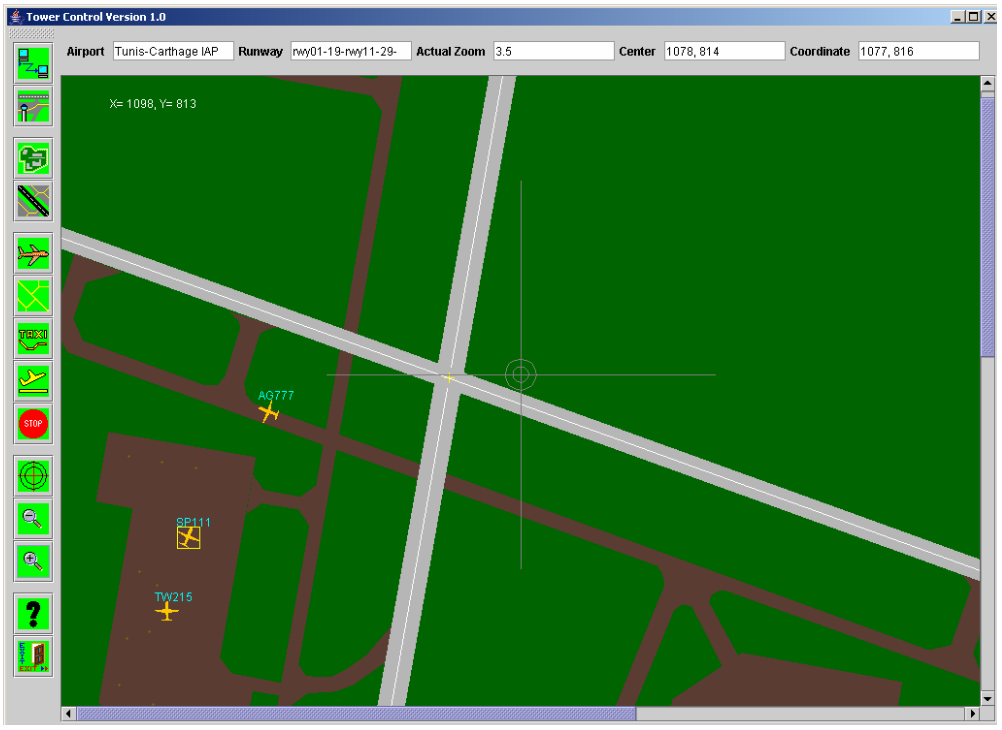


Figure 27. The UML diagram of the cockpit module

2. The Ground Module

The second module of the pilot package is the ground module. This module allows the pilot to manipulate the aircraft on the ground. Such manipulations include the positioning of the aircraft in a given parking position, the taxiing, and the takeoff of the aircraft. The use of this GUI is made easy by a set of control buttons in four groups. The first group is used to connect to the server and to choose the airport in service. The second group allows the pilot to personalize the airport appearance, by allowing the pilot to hide or show the most common graphical component in the airport. The third group allows the pilot to show or hide the aircraft or their current path within the airport. The fourth is the most important group since it directly manipulates the aircraft position and parameters. This group is normally inactive until the pilot selects an aircraft (by double-clicking on it), at which time this group will be highlighted, indicating the possible actions that can be undertaken at that time. These actions are either stopping the aircraft from moving, taxiing the aircraft in the pre-established path or initiating the takeoff. The last group of buttons is reserved for manipulating the view by changing the center or the scale. All active parameters of the airport are shown in a panel situated at the top of the ground module GUI. A complete ground GUI is shown in the Figure 28 below.


Tower Control Version 1.0

Airport: Tunis-Carthage IAP
Runway: rwy01-19-rwy11-29-
Actual Zoom: 3.5
Center: 1078, 814
Coordinate: 1077, 816

Figure 28. The ground GUI of the pilot package

The UML diagram of the ground module is shown in Figure 29. As it will be seen later in the Tower section, the ground module of the pilot part is simply an extended version of the tower module, which explains why the main class of the UML diagram is named “Tower”. The UML diagram shows that all the graphical objects (those using a graphical context to depict some kind of shape or symbols) inherit one root object which is the “GraphicalShape” class. This class is an abstract class containing the most common methods shared by all its children classes. Some of the methods of this class are rendered abstract allowing the children to implement more personalized ones. Among the most important methods that are implemented by the children is the draw method of the super class. The UML diagram shows also the presence of two classes specific to this module: the Mover class, and the “TakeoffProcessor” class. The following sections give some details about these two classes.

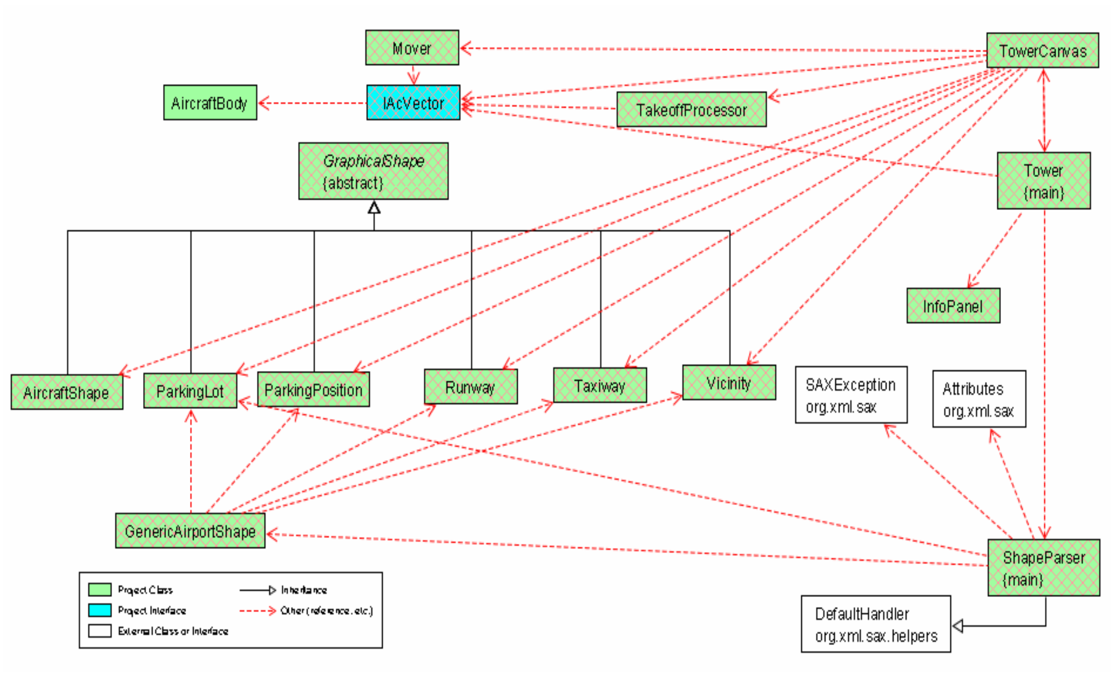


Figure 29. The UML diagram of the ground part

a. Aircraft Mover

Within the ground module the movement of aircraft is controlled by a thread named “Mover”. This thread looks at the selected aircraft and verifies that it has a path assigned to it. If so, a linear movement process is applied to that aircraft's position. The mover moves the aircraft at a constant speed (v) according to the following equations:

$x(t) = x(t_0) + v_x(t-t_0)$

$y(t) = y(t_0) + v_y(t-t_0)$

where $v_x = v.\cos(\theta)$, $v_y = v.\sin(\theta)$, and θ is the arctangent of the current segment of the path assigned to the aircraft. These equations are converted in numerical form in the following way:

$x(n + 1) = x(n) + v_x.\Delta t$

$x(n + 1) = x(n) + v_x.\Delta t$

and if $\Delta t = 1$, we simply obtain $x = x + v_x$, and $y = y + v_y$.

The change from one segment to another is made possible using the intersection of the current position of the aircraft and a rectangular area centered in the endpoint of the current segment. This rectangular area has a width, respectively a high, equal to $2v_x$, respectively $2.v_y$. In such way it is guaranteed that the intersection will occur.

b. The Takeoff Processor

The takeoff processor is a thread responsible of the takeoff phase of the aircraft. It guides the aircraft with the correct orientation along the runway. At every step the thread increases the power of the aircraft gradually until its maximum is reached. The increase of power causes an increase in speed, and once the aircraft reaches its takeoff speed (for simplification, all aircraft have the same takeoff speed), the thread increases its altitude until it reaches 1,500 feet (as mentioned in chapter II, this is the height limit of the tower control). After that point the thread dies and the aircraft is left alone to follow its pre-established route or be subject to further pilot actions.

Before takeoff however, the "TowerCanvas", which is the running process of the main class, executes a series of verifications to ensure that the aircraft is in fact ready for takeoff. First it gets the current aircraft position and orientation (the selected aircraft is marked with a yellow diamond shape). Then it asks the runway in service whether or not the selected aircraft is within its limit and has an acceptable orientation (the class runway has a specific method to calculate the orientation tolerance). If all these conditions are met, the class modifies the route of the aircraft in the following way. The current aircraft's position becomes the first waypoint, and the end of the runway becomes its second waypoint. The rest of waypoints remain unchanged. The final step is to launch the TakeoffProcessor class, and the aircraft will behave as explained earlier in the first paragraph of this section. Figure 30 shows a flowchart that summarizes the major steps of the takeoff process.

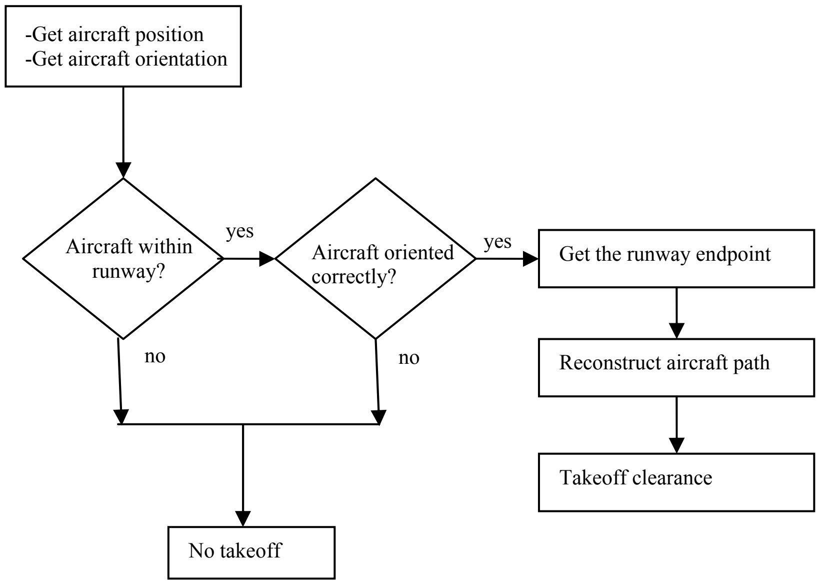


Figure 30. The takeoff verification process

D. THE TOWER MODULE

The tower module is very similar to the ground module of the pilot package. The main difference between the two modules is that the tower module does not contain any means to manipulate the aircraft. In addition this module contains a placing algorithm to park the aircraft in the airport. Figure 31 shows the UML diagram of this module.

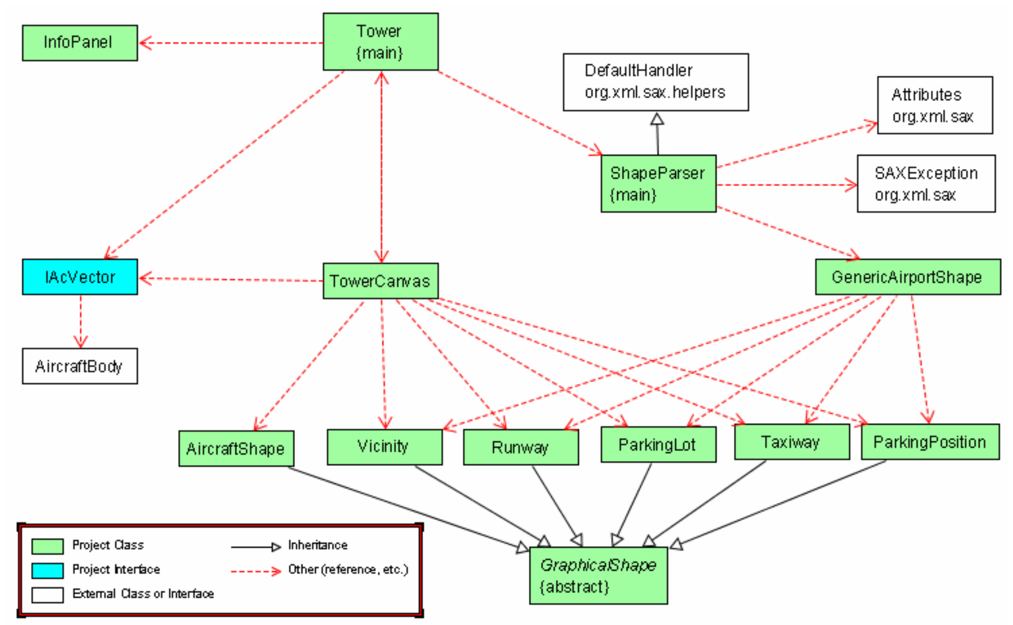


Figure 31. The UML diagram of the tower module

1. Placing Algorithm

One important task of the tower module is determining the initial position of every ground aircraft at the beginning of the simulation exercise. In fact, all the ground aircraft at the beginning of the exercise have as initial location the center of the airport. Without any intervention all the ground aircraft would occupy the same position within the airport. To solve this problem a placing algorithm is applied to the list off all aircraft on the ground. The placing algorithm, depicted in Figure 32, simply chooses a parking position in the airport and assigns it to the first non-positioned aircraft in the filtered list. This process continues until all aircraft are positioned. Then the algorithm updates the remote service with the new position of the aircraft. This algorithm operates only once at the beginning of every simulation. If the aircraft received an order to taxi, its parking position is made free so the parking algorithm can assign it to another aircraft. For the arrival aircraft, the pilot will receive instruction from the tower controller about where to place the aircraft.

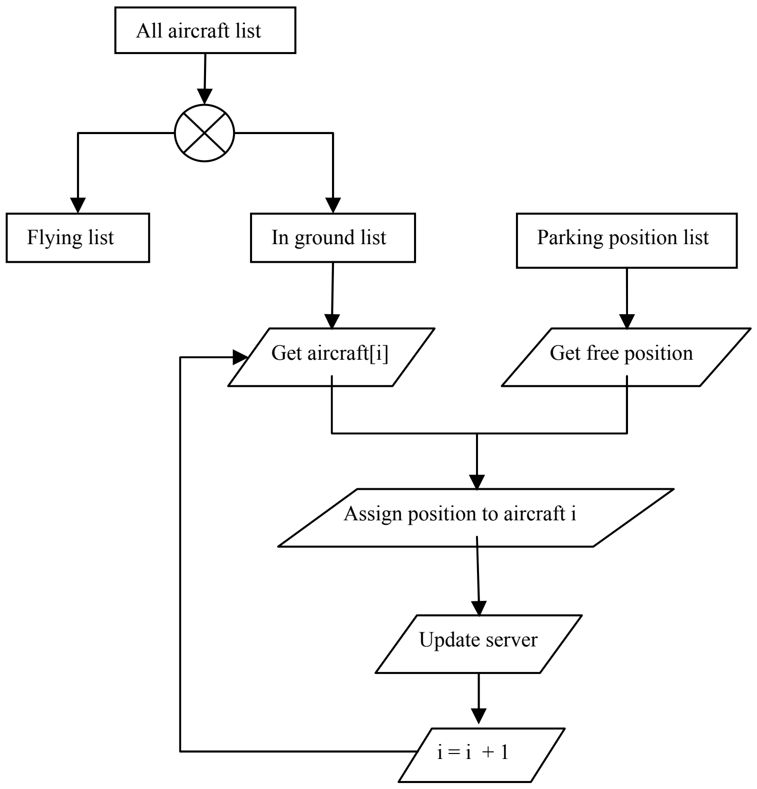


Figure 32. The placing algorithm

E. APPROACH MODULE

The approach module is used by the ATCSs either during the approach phase or after the takeoff and before the aircraft reaches its cruising speed and altitude. The GUI of this module is similar to many real approach screens. The approach module is articulated according to the UML diagram of Figure 33. The diagram shows two main classes that are the center of gravity of the approach application.

The first class, “ControllerWS”, represents the controller GUI. It contains all the necessary menus and option panels to allow the controller manipulating it to gain the best awareness of the air situation. The use of this class is similar to all the other GUIs of the simulator. The controller has only to connect to the available server and customize the interface.

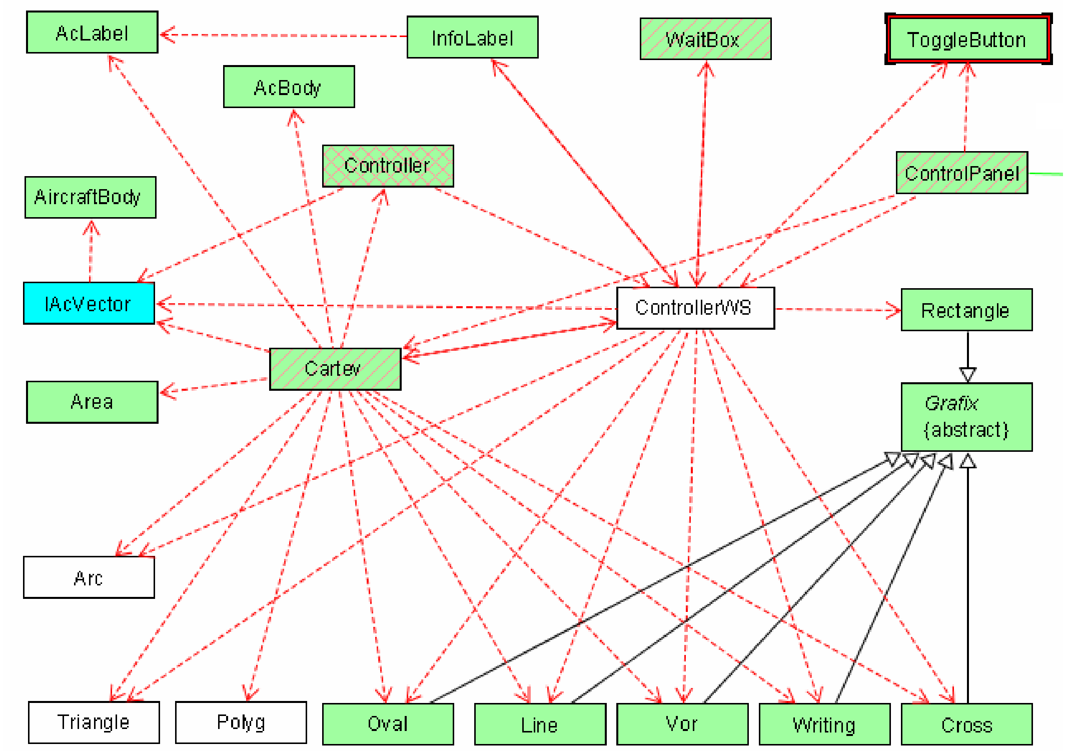


Figure 33. The UML diagram of the approach module

The second most important class in this module is the "Cartev" class. This class represents the running process of the class "ControllerWS". At every loop, Cartev checks the controller settings and draws all the required components. It also updates the aircraft situation and represents them in the correct position. In order to make it easy to represent the many graphical components, this class uses a double buffering technique. While an image is presented to the user, another one is prepared for the next loop. In this way, the user will not observe any noticeable discontinuity in the GUI. Figure 34 shows the overall approach workstation GUI. Figure 34 also shows the control panel used by the controller to quickly manipulate the GUI. This control panel is an instance of the class "ControlPanel" shown in Figure 33.

Even though these two classes are the most important classes in this module, there are many others that play a role for the controller and the overall simulation. The class "Controller" is one example. This class is responsible for detecting any situation that may lead to an air conflict. As explained in chapter III, there are two air conflict situations, both occurring when two or more aircraft volumes intersect with each other. The difference between the two types of air conflict resides only in the extent of the aircraft volumes. To keep track of the air conflict situation, this class has two bi-dimensional boolean arrays. If the element [i][j] of a particular array is true, this means that the aircraft [i] is in some air conflict form with the aircraft [j]. All the parameters of the air conflict are exposed in this class and can be changed by the instructor when required.

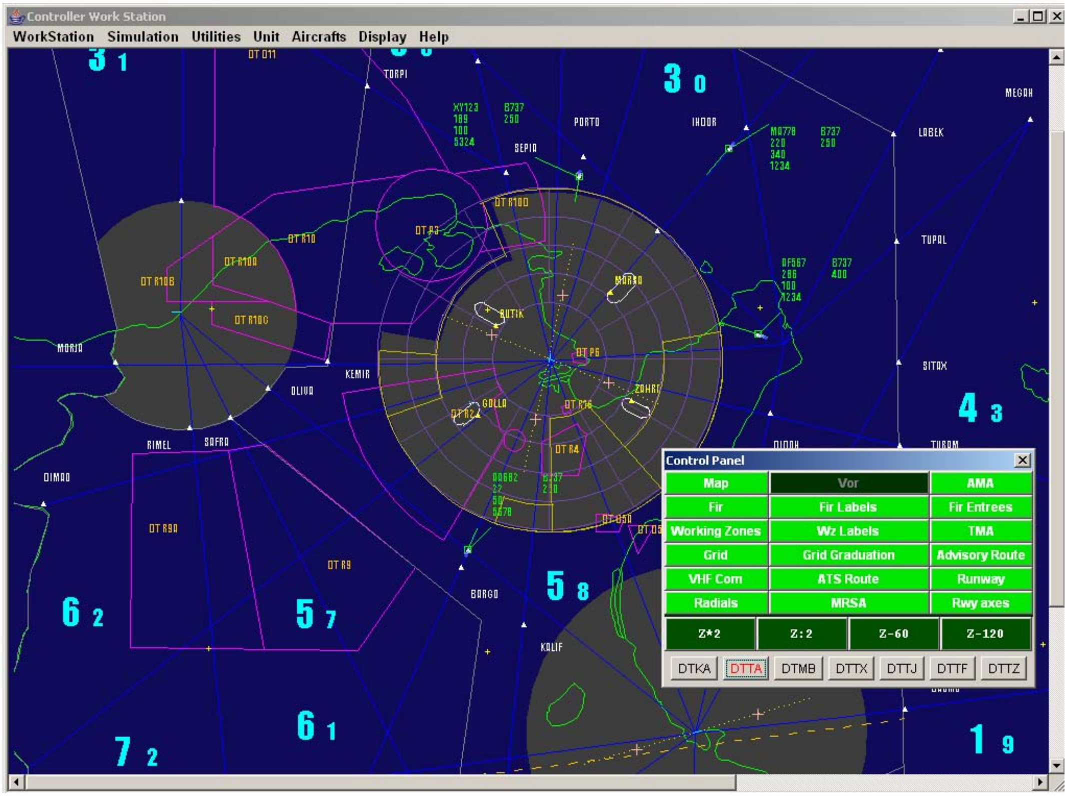


Figure 34. The approach workstation GUI

The approach module is the largest module of the ATCS. Its description alone is quite lengthy, so only an overview will be given here. Many variables and method of this module are either self-explanatory or well commented and can be reviewed in the module documentation part. The following sections will provide extra clarifications about some mathematical concepts used in this module. These concepts are related to the coordinate conversion and runways representation.

1. Coordinate Conversion

In a real aeronautical map the intersections between the longitudes and latitudes form a quasi rectangular shape delimiting a geographic area. During the process of preparing the data for the application, the upper left corner (in terms of longitude and latitude) of each of these areas are acquired. These data along with the pixel coordinates of the four corners are used as arguments to construct the object "Area" (Figure 35).

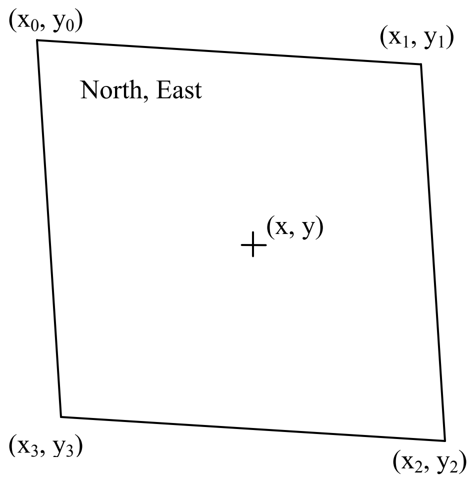


Figure 35. A simple geographic area representation

These area objects are used to determine the approximate x-y coordinate of a given point expressed in universal geographic coordinate. This is done in two steps. First, the data structure containing all Area objects is checked to identify that particular area having the same longitude and latitude as the given point. Second, once the area is identified, a simple interpolation formula is used to calculate the corresponding x-y coordinates of the given point. The portion of code doing such operation is the following:

```
if(area.getNorth()==nord && area.getEast()==est){
        x1 =(double)(xpoints[0]+xpoints[3])/2;
        x2 = (double)(xpoints[1]+xpoints[2])/2;
        y1 = (double)(ypoints[0]+ypoints[1])/2;
        y2 = (double)(ypoints[3] + ypoints[2])/2;
        x = x1+ (((x2-x1)*(me*6000.0+se*100.0+ce))/360000.0);
        y= y2 - (((y2- y1)*(mn*6000.0+sn*100.0+cn))/360000.0);
        break;
}
```

2. Runway Representation

The runways are represented differently within each module of ATC simulator for two reasons. The first reason is that not all modules require the same details and precision. The second reason is that in the normal aeronautical maps, runways are not represented due to their small dimensions. To solve this problem, two approaches were adopted. For the TRACON and en-route modules, each runway is represented as a line segment. The endpoints of this segment are directly obtained from the geographical coordinates of the runway. The tower module utilizes the XML representation since the spatial photo of the airport is sufficiently clear to acquire all data required for the airport (mainly the runways, the taxiways, and the parking areas). The following paragraphs explain how the end points of each runway are obtained, and how other important components are computed, such as the runways' extensions which are important especially in the landing phase of the aircraft.

By using the method described in the previous section, it is easy to convert the geographical coordinates of the runways into the corresponding x-y coordinates of the simulator. For example the geographical coordinates of the runway 01 of Tunis-Carthage airport (code DTTA) are 36° 50' 20.87"N and 010° 13' 25.52"E which give X= 1079.47316805, Y= 809.49381944. This method was also useful in determining the center of the airport which is very important in the entire application. In fact to obtain a perfect conservation of the coordinates in all modules, it is mandatory that the center is the same in all modules. This is because the center of the airport plays the role of the origin of a local coordinate system, and obviously every object is designed around the origin of this coordinate system.

The same process is used to calculate the runway extensions which are virtual lines extending each endpoint of the runway. These extension lines are important since the help a lot the controller to approximately align the aircraft with the runway in service during the approach phase. They are depicted as dashed lines in the Approach GUI. To determine the low and high point of each runway extension (see Figure 36), we first used the endpoints of the runway to compute its slope. The low and high points are, respectively, the intersection of the runway and the circle with radius r_l , and radius r_h, centered on one of the runway extremity.

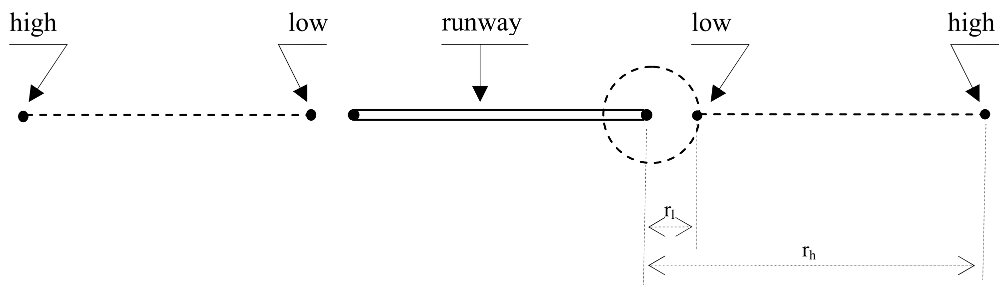


Figure 36. Runway extensions

In mathematical terms, each point is the solution of the following equations:

$(x - x_0)^2 + (y - y_0)^2 = r^2$ (equation 1)

$(y - y_0) = m(x - x_0)$ (equation 2)

In the above equations, (x_0, y_0) is the extremity of the runway, r is either of the radius r_l and r_h, and m is the slope of the runway. The solution of these equations is:

$x = x_0 \pm r / \sqrt{1 + m^2}$ (equation 3)

To decide which x is needed, the center of the runway is used as a reference point. If the x-coordinate of the runway extremity is greater than the x-coordinate of the runway center, then the correct x will be obtained with the plus sign in equation 3, otherwise the minus sign will be used.

F. THE EN-ROUTE MODULE

One of the most important advantages in using the Object-Oriented Programming (OOP) is the usability of the produced classes and code. This concept was fully applied during the implementation of the tower module. In fact, as mentioned, the tower module was only a modified version of the ground sub-module of the pilot module. The same concept is used to implement the en-route part. The en-route module is simply a modified version of the approach module.

Unlike the approach controller who should dispose only of the air situation around particular airport, the en-route controller should have access to all controlled airspace (precisely, the airspace within the Flight Instruction Region (IFR)). This was done in the implementation by centering the map and all the other objects around any point chosen by the en-route controller (see Figure 37 below).

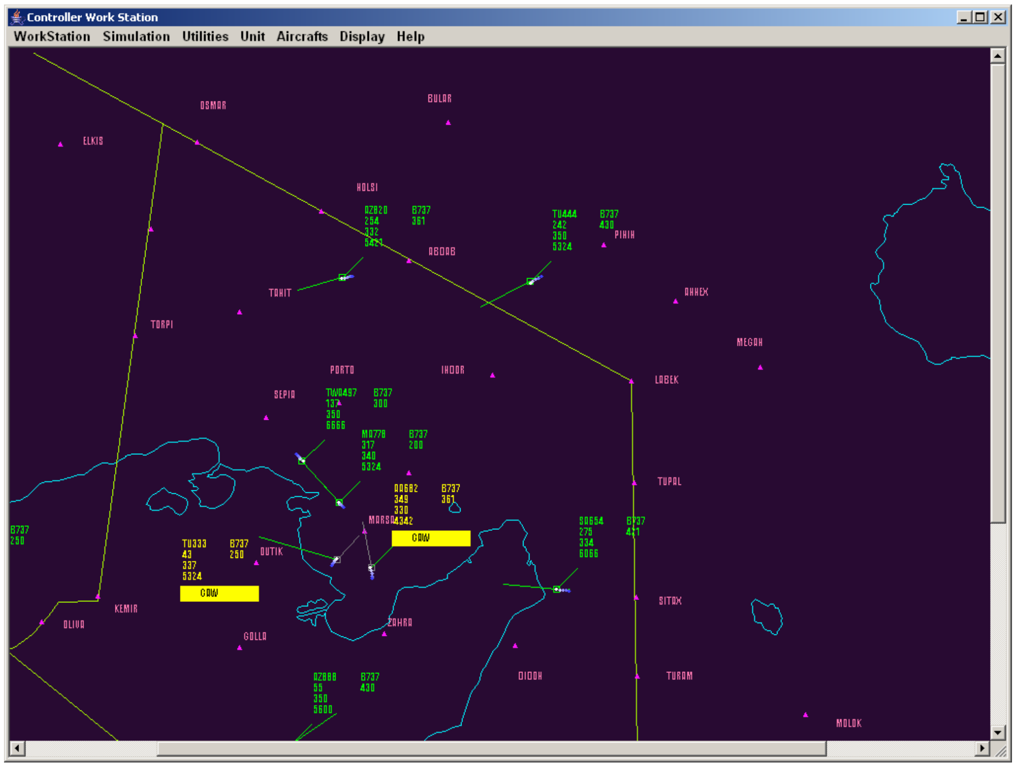


Figure 37. The en-route GUI

VI. CONCLUSIONS AND FURTHER WORK

A. CONCLUSION

Air Traffic Control (ATC) is a network of facilities whose purpose is to organize aircraft movements in airspaces in order to guarantee safe and efficient air traffic. ATC contains many kind of specialists (ATCS) commonly grouped in three main categories; tower, approach and en-route. To meet the required skills necessary to manage an incessant evolving air traffic density, these specialists should be trained continuously. However, often the training method consists in using separated modules for each air traffic control category. The most important deficiency of such training method is the lack of the concept of teamwork.

The work in this thesis consists of conceiving and implementing a teamwork air traffic control simulator that does not suffer the previously mentioned drawbacks, and that could be easily used in ATC schools. The goal is to allow the different ATCSs to work together, in the same conditions as in the job environment, in order to manage in a collaborative way an air traffic situation. A simulator like this should allow ATCSs to acquire further coordination skills, and hence, reduce the amount of air traffic faults due to a lack of teamwork in the actual ATC training system.

The product of this thesis was developed using a waterfall software development life cycle model. This model was preferred to other approaches (prototype, spiral, extreme programming, etc) because of its simplicity as well as ease of understanding and implementation. In addition, the waterfall model forces a verification activity in between each two adjacent steps which guaranties a big degree of fidelity and robustness to the realized software.

The resulted simulator is a flexible training tool for both trainees and instructors. A simulator like this will help in obtaining a low cost solution that can substitute buying simulators for each category of ATCSs. In addition, the main training capabilities of this simulator could be summarized as:

1. Decreasing the OJT time.
2. Help to increase teamwork performances, and hence, reduce air traffic faults.
3. Give some insights in teamwork requirements for Air Traffic Control Simulators.

B. FUTURE RESEARCH

This study's goal was to design, develop, implement, test, and evaluate a novel air traffic control simulator that offer a low-cost solution to enhance the concept of teamwork in ATC design, but it raised many interesting questions than are beyond its scope.

Possible future research includes questions about design improvement and the influence of assumptions made in the model. Other questions may include: Do the modules of this simulator meet their requirements and specifications? How much can the model be simplified? This software can also be the subject or the tool for many specialized studies related to the ergonomic and human factor issues such as the usability study, or can be used as a tool for assessing and evaluating training methods. The following sections briefly discuss some of these studies.

1. Usability Study

Modernization of Air Traffic Control (ATC) display systems includes increased use of color to code information. While colors can enhance display designs, human factors issues like legibility and salience manipulation are still problematic. In implementing this simulator, no particular considerations about the choice of the different used colors were made. Some researchers argue that color palettes that are not specifically designed for layered data and a large number of objects can create legibility and salience problems (Ahlstrom & Arend, 2005). Even though all modules of this application (except the instructor module) contain tools allowing the user some degree of freedom to customize the GUI, it is recommended to apply a usability study for all modules in order to detect:

- Human factor limitations about the GUI layout.
- Influences of the color palette in the overall controller's activities and awareness.
- Influence of brightness and contrast in the controller awareness.
- Legibility, salience manipulation (clutter avoidance), and color recognition.

2. Design of Experiments

As stated previously, this software is a versatile and can be used for a multitude of purposes such as:

- General ATC training.
- Rehearsal is some ATC issues.
- Emphasize team and teamwork skills in ATC.
- Improvement of the Time Of Transfer (TOT) rate.

However, none of these capabilities were tested or analyzed using this software. Hence, it is recommended to design experiments about one or more of the above aspects in order to demonstrate potential training improvements. This can be done using two groups of trainees: a control group, whose member are trained using the traditional tools, and a second group that uses this software as the main training tool. A statistical analysis at the end of the training period could identify the differences, if any, between the two approaches.

3. Rebuilding the Tower Module

The tower module was designed using a 2D projection. Even though this does not compromise the objective of this thesis, it would be a good idea to implement it by a 3D model. Many 3D authoring tools allow an easy development of a 3D model provided that the user possesses all the data of the intended model. These tools include X3D, Visx3D, Wings3D, SwirlX3D, etc. The author has already developed a 3D model of the Tunis-Carthage International Airport (see Figure 37) however this model is not integrated in the overall application.

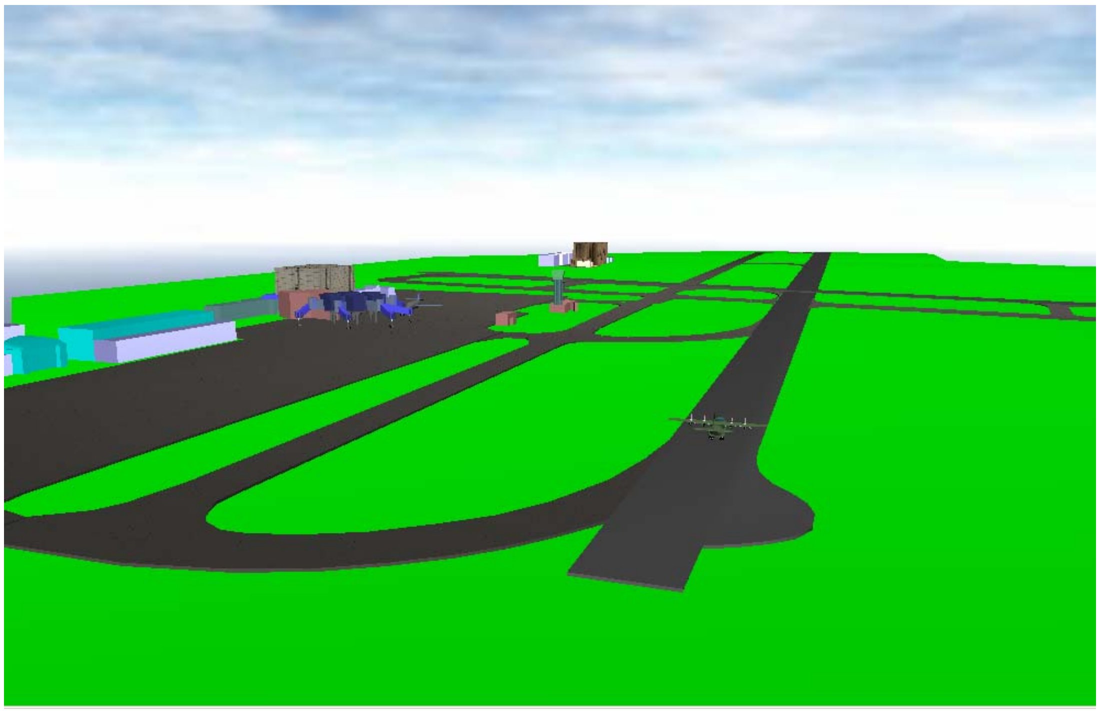


Figure 38. 3D model of the Tower module

Once the 3D model is completed, the tower module can be implemented using any Java package that supports 3D scene, such as Java3D or the Scene Access Interface (SAI). The reader can refer to Appendix C for a short description of SAI.

LIST OF REFERENCES

Ahlstrom, U., Arend, L. (2005). Color Usability on Air Traffic Control Displays: Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting-2005. Federal Aviation Administration, Atlantic City International Airport, NJ, USA, NASA Ames Research Center, Moffett Field, CA, USA.

Air Traffic Management System. *Safe Separation Standards*. Foud at http://virtualskies.arc.nasa.gov/ATM/tutorial/tutorial6.html, Jan 2005.

Aviation Sri Lanka. Air Traffic Control History. Found in http://atcsl.tripod.com/air_traffic_control_history.htm, Dec 2005.

Bailey, L.L., Broach, D.M., Thompson, R.C. & Enos, R.J. (1999). *Controller teamwork evaluation and assessment methodology: A scenario calibration study* (DOT/FAA/AM-99/24). Washington, DC: Federal Aviation Administration.

Bennett, S., Skelton, J. & Lunn K. (2001). *UML*. McGraw Hill.

Brutzman, P. D. (2003). *Web-based 3D Graphics Rendering of Dynamic Deformation Structures in Large-scale Distributed Simulation*. Naval Postgraduate School, Monterey, CA.

CORBA & RMI. Found at http://www.dalmatian.com/corba.htm. Jan 2006.

D'Arcy, J.F., Della Rocco, S. P. (2001). *Air Traffic Control Specialist Decision Making and Strategic Planning – A Field Survey*. (DOT/FAA/CT-TN01/05). U.S. Department of Transportation. Federal Aviation Administration. Found in http://www.hf.faa.gov/docs/508/docs/wjhtc/tn0105.pdf. Jan 2006.

EUROCONTROL, European Organization for the Safety of Air Navigation. (1998). *Proceedings of the Second EUROCONTROL Human Factors Workshop Teamwork in Air Traffic Services*. (HUM.ET1.ST13.000-REP-02). EUROCONTROL.

Federal Aviation Administration. (2002). *Research was sponsored by the Federal Aviation Administration's Office of the Chief Scientific and Technical Advisor for Human Factors*. http://www.hf.faa.gov/docs/508/docs/volpe/volpe_9817.pdf. Dec 2005.

ICAO. *SSR Code Allocation List for the EUR Region*. 2003. Found at http://www.paris.icao.int/documents/pdf/CAL_Ed2_Amd3_PartA.pdf. Jan 2006.

Li, J., Yang, J. 2006. Xj3D, *Architecture of Xj3D Browser*. Found at http://wiki.cs.uiuc.edu/cs427/Xj3D#sai. Mar 2006.

Lintner, T. & Buckles, J. (1992). Why can't we talk to each other? In *Proceeding of the 37th Annual Air Traffic Control association Conference*, Atlantic City, NJ.

McCleam M. *Clear skies ahead*. (2003). Canadian Business. Found in http://www.adacel.com/press/innews/5_26_2003.pdf, Dec 2005.

National Research Council. (1998). *The Future of Air Traffic Control, Human Operators and Automation*. National Academy Press.

National Air Traffic Service. *The History of Air Traffic Control*. Found in http://www.nats.co.uk/library/history1.html, Dec 2005.

National Aerospace Laboratory of the Netherlands (NLR). *TRS: the NLR Tower Research Simulator*. Found in http://www.nlr.nl/documents/flyers/f172-06.pdf, Jan 2006.

National Defense. Modern Trainers on the Way for USAF Traffic Controllers. Found at http://www.nationaldefensemagazine.org/issues/2002/Nov/Modern_Trainers.htm, Jan 2006.

O'Hare D. & Roscoe S. N. (1990). *Flightdeck Performance: The Human Factor*. Iowa State University Press.

Plaettner-Hochwarth, K. J., Zhao K. Y. & Robinson, E. J. (2000). *A Modularized Approach for Comprehensive Air Traffic System Simulation*. American Institute of Aeronautics and Astronautics.

Ruiz L. E. (2004). *Perception of Communication Training Among Collegiate Aviation Flight Educators*. Aviation Institute, University of Nebraska at Omaha.

Schach, R. S. *Object Oriented and Classical Software Engineering*. (2002). Mc Graw Hill.

Reilly D. Reilly M. (2002). Java *Network Programming and Distributed Computing*. Adisson Wesley.

Smolensky, M. W. & Stein, E. S. *Human Factors in Air Traffic Control*. (1998). Academic Press.

Sood, N. & Wieland, F. (2003). *Total Airport and Airspace Model (TAAM) Parallelization Combining Sequential and Parallel Algorithms for Performance Enhancement.* Center for Advanced Aviation System Development. Found in http://www.informs-sim.org/wsc03papers/210.pdf, Jan 2006.

U.S Department of Transportation. (2002). *Aviation Delay*. Found in http://www.dot.gov/PerfPlan2004/mobility_delay.html, Dec 2005.

Wikipedia. *Terminal Control Center*. Found in http://en.wikipedia.org/wiki/TRACON, Dec 2005.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. GLOSSARY AND ABBREVIATIONS

ACC: Area Control Centre.

ACC is an ATC unit that provides ATC service to aircraft operating within a flight information region (FIR). See FIR and ATC for complete information.

AMA: Area Minimum Altitude.

The lowest altitude to be used under instrument meteorological conditions (IMC) that will provide a minimum vertical clearance of 1000 ft or, in designated mountainous terrain, 2000 ft above all obstacles located in the area specified, rounded up to the nearest 100-ft increment.

ATC: Air Traffic Control.

ATCS: Air Traffic Control Specialist.

ATS: Air Traffic Service.

ATS is a global term used to denote a set of services that includes ATC services, flight services and alerting service.

CORBA: Common Object Request Broker Architecture.

Specification for how different objects of some application will interact. CORBA has been designed from the beginning to support a wide range of network, operating system, and programming languages. In our days CORBA is viewed as the "elegant" solution to ensure interoperability and communication between heterogeneous objects.

CTA: Control Area.

The CTA is a controlled airspace extending upwards vertically from a specified height above the surface of the earth.

CTR: Control Zone.

A controlled airspace of defined dimensions extending upwards from the surface of the earth up to and including 3000 ft AAE unless otherwise specified. The shape of that zone is a cylinder with a radius of about 50 km. The height of this zone amounts 3000 feet (1000m).

FAA: Federal Aviation Administration

The FAA is federal authority in the United States of America responsible for civil aviation.

FIS: Flight Information Service

The FIS is a controlled part of an airspace in which the following services are offered:

- the dissemination of aviation weather information and aeronautical information for departure, destination and alternate aerodromes along a proposed route of flight.
- the dissemination of aviation weather information and aeronautical information to aircraft in flight.
- the acceptance, processing and activation of flight plans (FP) and flight itineraries, amendments to FPs and flight itineraries, and cancellations of FPs and flight itineraries.
- the exchange of FP information with domestic or foreign governments or agencies or foreign ATS units.
- the provision of known information concerning ground and air traffic.

FIR: Flight Information Region

An airspace of defined dimensions extending upwards from the surface of the earth within which flight information service (FIS) and alerting service are provided.

FL: Flight Level.

The altitude expressed in hundreds of feet indicated on an altimeter set to 29.92 in. of mercury or 1013.2 mb. For example an FL of 300 indicates a flying altitude of 30,000 feet.

MRSA: Mandatory Radar Service Area.

PSR: Primary Surveillance Radar.

Radar equipment used to determine the position of an aircraft in range (distance between the antenna and the illuminated object) and azimuth (angle between the horizontal antenna plane and the horizontal plane containing the aircraft).

SSR: Secondary Surveillance Radar.

The SSR is an antenna (interrogator) attached to the primary radar emitting special signal called interrogation signal and wait for the "answer" from a particular antenna mounted in the aircraft of other vehicle generally known as transponder. Once receives the interrogation signal, the transponder generates a coded reply signal that includes a lot of information about the aircraft status. This information includes the aircraft identification, speed, cap, and any other anomaly such as communication failure, fire, and aircraft hijacked.

SSR code

A four-digit octal number received from the aircraft transponder when it is interrogated by secondary surveillance radar (SSR). This code could be changed by the pilot or automatically when some abnormal conditions happen in the aircraft such as communication failure, fire, and aircraft hijacked. In such way the SSR code is a valuable piece of information for the controller since it gives additional information about the aircraft status.

TCA: Terminal Control Area.

A controlled airspace of defined dimensions that is normally established in the vicinity of one or more major aerodromes and within which ATC service is provided based on the airspace classification.

TMA: Terminal Control Area.

The TMA is the ICAO abbreviation for TCA (see TCA). The TMA starts between 1500 ft and 2500ft, that depends on the height of the highest obstacles in the surroundings of the airport and goes on in the CTA sector till 10500 ft.

APPENDIX B. THE SERVICES' INTERFACES

The code of the Air Traffic Control Simulator (ATCS) consists of approximately 25,000 lines of code. The reader can refer to the download site[5] for a complete code listing and documentation. However, in order to have an idea about the services offered by the server module, the following are the listings of the aircraft service and the student service explained in Chapters IV and V.

```
package rmiServer;

/************************************************************************
 * class IAcVector.java is an interface for the remote aircraft service
 * of the air traffic control simulator.
 * @version $Id: IAcVector.java,v 1.3 2006/03/30 17:23:53 msidhom Exp $.
 * @author Mounir Sidhom
 ************************************************************************/

   import java.rmi.*;
   import java.awt.Point;
   import java.awt.geom.*;
   import java.util.Vector;

    public interface IAcVector extends Remote {

   /**
    * this method echoes the connected client via the RMI service.
    * It is only used for testing purposes.
    * @return a string representing the connected client.
    */
       public String getConnectedClient() throws RemoteException;

   /**
    * add an aircraft to the list of aircraft.
    * @param ac an instance of AircraftBody.
    */
       public void addObjectAircraft(AircraftBody ac) throws RemoteException;

   /**
    * add an aircraft to the list of aircraft.
    * @param indicatif the aircraft's flight ID.
    * @param codeSSR the aircraft's SSR code.
    * @param speed the aircraft's speed.
    * @param cap the aircraft's heading.
    * @param svertical the aircraft's altitude.
    * @param ptsDest an array of waypoints coordinates.
    * @param pointsLbls an array of waypoints labels.
    * @param depTime a string representing the departure time.
    */
    public void addAircraft(String indicatif,String codeSSR,
                          float speed,float cap, float alt,float svertical,
                          Point [] ptsDest,String [] pointsLbls,
```

5 http://diana.cs.nps.navy.mil/~msidhom/

```
                        String depTime) throws RemoteException;

    /**
     * remove the given aircraft from the active aircraft list.
     * @param indicatif the aircraft's flight ID.
     */
        public void removeAircraft(String indicatif) throws RemoteException;

    /**
     * get the last removed aircraft.
     * @return the flight ID of the last removed aircraft.
     */
        public String getRemovedFltID() throws RemoteException;

    /**
     * remove all aircraft from the aircraft list.
     */
        public void removeAllAircrafts() throws RemoteException;

    /**
     * start the given aircraft.
     * @param indicatif the aircraft's flight ID.
     */
        public void startAircraft(String indicatif) throws RemoteException;

    /**
     * start all aircraft in the aircraft list.
     */
        public void startAllAircrafts() throws RemoteException;

    /**
     * freeze the given aircraft. This will cause the thread moving
     * the aircraft to momentarily stop running.
     * @param indicatif the aircraft's flight ID.
     */
        public void freezeAircraft(String indicatif) throws RemoteException;

    /**
     * freeze all aircraft of the exercise.
     */
        public void freezeAllAircrafts() throws RemoteException;

    /**
     * unfreeze the given aircraft. This will cause the thread associated
     * to the aircraft to continue running.
     * @param indicatif the aircraft's flight ID.
     */
        public void unFreezeAircraft(String indicatif) throws RemoteException;

    /**
     * unfreeze all aircraft.
     */
        public void unFreezeAllAircrafts() throws RemoteException;

    /**
     * get the size of the actual aircraft list.
     * @return the size of the aircraft list.
     */
        public int getAcVectorSize() throws RemoteException;

    /**
```

```
 * get the aircraft's flight ID.
 * @param index the index of the aircraft in the list of aircraft.
 * @return the aircraft's flight ID.
 */
    public String getAcID(int index) throws RemoteException;

/**
 * get all aircraft's flight ID.
 * @return a array of flight ID.
 */
    public String [] getAcsID() throws RemoteException;

/**
 * get the aircraft location.
 * @param indicatif the aircraft's flight ID.
 * @return a point representing the aircraft location.
 */
    public Point getAcXY(String indicatif) throws RemoteException;

/**
 * get the aircraft location.
 * @param indicatif the aircraft's flight ID.
 * @return the aircraft location as an array of double values.
 */
    public double[] getLocation(String indicatif) throws RemoteException;

/**
 * set the aircraft location.
 * @param indicatif the aircraft's flight ID.
 * @param x the x-coordinate of the aircraft location.
 * @param y the y-coordinate of the aircraft location.
 */
    public void setLocation(String indicatif,
                           double x, double y) throws RemoteException;

/**
 * set the aircraft parking name.
 * @param indicatif the aircraft's flight ID.
 * @param parking the aircraft parking name.
 */
    public void setParkingName(String indicatif,
                               String parking)throws RemoteException;

/**
 * get the aircraft parking name.
 * @param indicatif the aircraft's flight ID.
 * @return the parking name of the given aircraft.
 */
    public String getParkingName(String indicatif)throws RemoteException;

/**
 * set the aircraft route waypoints.
 * @param indicatif the aircraft's flight ID.
 * @param ptDest an array of waypoints.
 */
    public void setRoutePoints(String indicatif,
                 Point[] ptDest)throws RemoteException;


/**
 * get all aircraft positions.
 * @return an array of positions.
 */
    public float [][] getAcsXY() throws RemoteException;

/**
 * set the heading of the aircraft.
 * @param indicatif the aircraft's flight ID.
 * @param cap the new heading.
 */
    public void setCap(String indicatif, double cap) throws RemoteException;

/**
 * make the aircraft reach a given point.
 * @param indicatif the aircraft's flight ID.
 * @param x the x-coordinate of the point to be reached.
 * @param y the Y-coordinate of the point to be reached.
 */
    public void reachPoint(String indicatif, int x, int y)
    throws RemoteException;

    public void setSegment(String indicatif, int segment)
    throws RemoteException;

    public Point getAcSpeedXY(String indicatif) throws RemoteException;

    public float [][] getAcsSpeedXY() throws RemoteException;

/**
 * get the aircraft speed.
 * @param indicatif the aircraft's flight ID.
 * @return the aircraft speed.
 */
    public float getAcSpeed(String indicatif) throws RemoteException;

/**
 * get all aircraft speeds.
 * @return an array containing all aircraft speeds.
 */
    public float [] getAcsSpeed() throws RemoteException;

/**
 * get the aircraft heading.
 * @param indicatif the aircraft's flight ID.
 * @return the aircraft heading.
 */
    public int getAcCap(String indicatif) throws RemoteException;

/**
 * get all aircraft's headings.
 * @return an array containing all aircraft headings.
 */
    public int [] getAcsCap() throws RemoteException;

/**
 * get the aircraft altitude.
 * @param indicatif the aircraft's flight ID.
 * @return the aircraft altitude.
 */
    public float getAcLevel(String indicatif) throws RemoteException;

/**
 * get all aircraft's altitudes.
 * @return an array containing all aircraft altitudes.
 */
    public float [] getAcsLevel() throws RemoteException;

/**
 * increase the altitude of a given aircraft.
 * @param indicatif the aircraft's flight ID.
 * @param alt the new altitude to reach.
 * @param rate the climbing rate.
 */
    public void upAc(String indicatif,float alt,float rate)
    throws RemoteException;

/**
 * decrease the altitude of a given aircraft.
 * @param indicatif the aircraft's flight ID.
 * @param alt the new altitude to reach.
 * @param rate the climbing rate.
 */
    public void downAc(String indicatif,float alt, float rate)
    throws RemoteException;

/**
 * set the speed of the given aircraft.
 * @param indicatif the aircraft's flight ID.
 * @param newSpeed the aircraft's new speed.
 */
    public void setAcSpeed(String indicatif, float newSpeed)
    throws RemoteException;

/**
 * change the aircraft's heading.
 * @param indicatif the aircraft's flight ID.
 * @param sens the direction of the turn (left = false, right = true).
 */
    public void acTurn(String indicatif,boolean sens, short cap,float rate)
    throws RemoteException;

/**
 * get all SSR code of the aircraft list.
 * @return an array of all SSR codes
 */
    public String [] getAcsCodeSSR()throws RemoteException;

/**
 * get the SSR code of the given aircraft.
 * @param indicatif the aircraft's flight ID.
 * @return a string representing the SSR code of the aircraft.
 */
    public String getAcCodeSSR(String indicatif)throws RemoteException;

/**
 * set the SSR code of the given aircraft.
 * @param indicatif the aircraft's flight ID.
 * @param newCodeSSR the new SSR code.
 */
    public void setAcCodeSSR(String indicatif, String newCodeSSR)
    throws RemoteException;


/**
 * get the category code of the given aircraft.
 * @param indicatif the aircraft's flight ID.
```

```
     * @return a string representing the category of the aircraft.
     */
        public String getAcCodeSpec(String indicatif) throws RemoteException;

    /**
     * get the category code of all aircraft.
     * @return a array of strings representing the category of the aircraft.
     */
        public String [] getAcsCodeSpec() throws RemoteException;

    /**
     * set the category code of the given aircraft.
     * @param indicatif the aircraft's flight ID.
     * @param newCodeSpec the new aircaft's category code.
     */
        public void setAcCodeSpec(String indicatif, String newCodeSpec)
        throws RemoteException;

    /**
     * get all aircraft status.
     */
        public boolean [] getAcsStatus() throws RemoteException;

    /**
     * get the aircraft's waypoints labels.
     * @param indicatif the aircraft's flight ID.
     * @return an array of the aircraft's waypoints labels.
     */
        public String [] getAcRouteLabels(String indicatif)
        throws RemoteException;

    /**
     * get the aircraft's waypoints.
     * @param indicatif the aircraft's flight ID.
     * @return an array of the aircraft route's waypoints.
     */
        public Point [] getAcRoutePoints(String indicatif)
        throws RemoteException;

    /**
     * get the current route segment of the given aircraft.
     * @param indicatif the aircraft's flight ID.
     * @return an int representing the current segment.
     */
        public int getCurrentAcSegment(String indicatif) throws RemoteException;

    /**
     * get departure times of all aircarft.
     * @return an array of depature times.
     */
        public int [] getAcsDepTime() throws RemoteException;

    /**
     * get the departure time of the given aircraft.
     * @param indicatif the aircraft's flight ID.
     * @return a string representing the deaprture time of the aircraft.
     */
        public String getAcTime(String indicatif) throws RemoteException;

}
```

```
package rmiserver

/***************************************************************************
 * class StudentServiceInterface is a remote service interface to keep track
 * of all activities in the Air Traffic Control Simulator.
 * @version $Id: StudentServiceInterface.java,v 1.1 2006/02/03 17:16:57$
 * @author Mounir Sidhom
 ***************************************************************************/
    import java.rmi.*;

    public interface StudentServiceInterface extends Remote {

   /**
    * get the student's identification string.
    * @param name the student's name.
    * @return a string representing the student ID.
    */
      public String getStudentID(String name) throws RemoteException;

   /**
    * get the student's nam.
    * @param studentID the student ID.
    * @return a string representing the student name.
    */
      public String getStudentName(String studentID) throws RemoteException;

   /**
    * get the time stamp of the student's connection.
    * @param studentID the student ID.
    * @return a string representing the time stamp of the connection.
    */
      public String getTimeStamp(String studentID) throws RemoteException;

   /**
    * insert the given event in the studens' database.
    * @param studentID the student ID.
    * @param event the given event.
    */
      public void insertEvent(String studentID, String event)
      throws RemoteException;
   }
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. SCENE AUTHORING INTERFACE (SAI)

The X3D Scene Authoring Interface (SAI) is an application programming interface (API) for the Extensible 3D Graphics (X3D) scene graph. The SAI combines the old external authoring interface and scripting interface (EAI) from the VRML97 specification and provides a single programming interface for either internal or external programming. SAI scripts will work for Script nodes inside the scene, external applets outside the scene in a Web page, Java and EcmaScript (i.e. JavaScript), and for language-independent scripting via XML's Document Object Model (DOM) (Li & Yang, 2006).

By using SAI, applications are able to create instances of browsers and manage the contents of the browser. External interactions form a separate code path from their close relative the scripting interactions. The principle difference is that an external interaction is not a direct part of the scene graph, where the scripting is. This component contains all of the implementation work needed to provide external access to the scene graph.

This thesis was first projected to use the SAI package to implement a 3D model of the tower module. The process was eased by the use of Visx3D[6] as a framework application. First, a 3D model of Tunis-Carthage International Airport was developed. Second, an early version of an aircraft 3D model[7] developed by the author was inserted in the airport (see Figure 38). However, this idea was discarded fro the following reasons:

- The SAI package is still in development phase, and changes in its implementation are frequent which make its users uncertain about their software especially if it is a large one like the air traffic control simulator.
- It is difficult to dynamically insert new 3D objects in an already running X3D scene.

6 Web site is at www.vizx3d.com

7 Available at http://web.nps.navy.mil/~brutzman/Savage/AircraftFixedWing/C130-Hercules-Tunisia

- The environment setting to use SAI is complicated and not yet standardized. It requires many modifications to the application's class path.
- The event delivery in SAI isn't deterministic. Don Brutzman wrote "the precise timing of event delivery (i.e. the event model) within X3D SAI scripts is no longer deterministic within the time bounds of a single event cascade." (Brutzman, 2003).

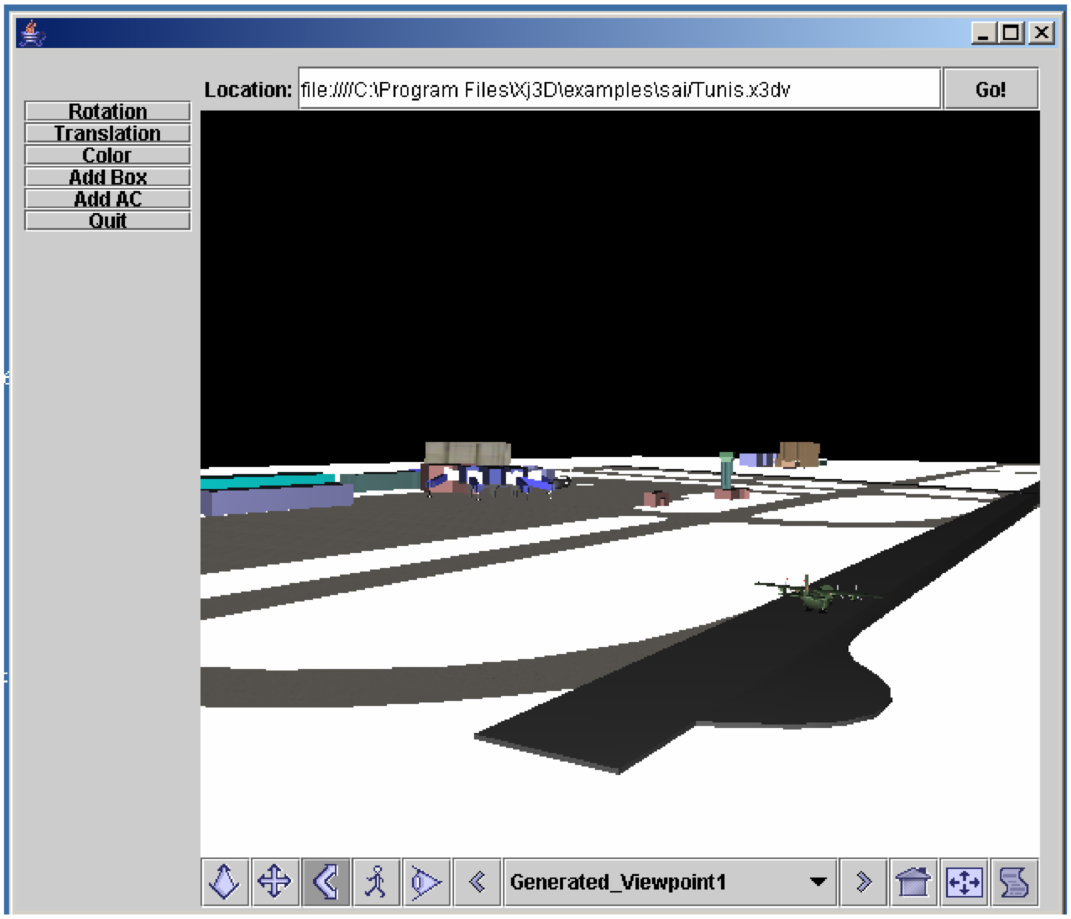


Figure 39. 3D model of Tunis airport rendered by SAI

When SAI reaches maturity, it will be a valuable asset to implement 3D models, such as the tower control module for the air traffic control simulator.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Tunisian Minister of Defense
   Tunisian Ministry of Defense
   Tunis, Tunisia

4. Arnie Buss
   MOVES Institute
   Monterey, California

5. Don McGregor
   MOVES Institute
   Monterey, California

6. Rudy Darken
   MOVES Institute
   Monterey, California