BlueSky ATC Simulator Project
An Open Data and Open Source Approach

Hoekstra, Jacco; Ellerbroek, Joost

Publication date
2016

Document Version
Accepted author manuscript

Published in
7th International Conference on Research in Air Transportation

BlueSky ATC Simulator Project: an Open Data and Open Source Approach

Jacco M. Hoekstra
Control & Simulation section
Faculty of Aerospace Engineering
TU Delft, Delft, Netherlands

Joost Ellerbroek
Control & Simulation Section
Faculty of Aerospace Engineering
TU Delft, Delft, Netherlands

***Abstract*—To advance ATM research as a science, ATM research results should be made more comparable. A possible way to do this is to share tools and data. This paper presents a project that investigates the feasibility of a fully open-source and open-data approach to air traffic simulation. Here, the first of the two main challenges is to achieve a high fidelity, without using proprietary data, e.g. for aircraft performance. The second challenge is to increase the adoption by the community by keeping the program easy to use, easy to modify, multi-platform, downloadable for free and running stand-alone on relatively simple systems. The approach chosen by the project is to investigate this feasibility by trying to start the development. The paper describes the many hurdles to be overcome when using a fully open-data and open-source policy in this area.**

Keywords: air traffic simulation; modelling; agent-based simulation; open-data; open-source; repeatability;traceability;

I. INTRODUCTION

Something important is missing in the field of ATM research. New concepts and new tools are common topics of ATM research projects. The result is often a proposed change with today's concept: a new tool, a new procedure or a change in task allocation either between man and machine, or between air and ground-side.

Most studies validate their proposed changes using either fast-time or human-in-the-loop simulations. To test for potential benefits, a comparison is made with today's operations. The results of these simulations are then expressed using custom defined metrics. The result is almost invariably positive: the new proposed concept or tool has benefits.

Many of the proposed concepts exclude each other. Examples are: improving the human machine interface for a certain task versus automating the task, or moving a task to the cockpit crew versus providing the air traffic controller with a tool for this task. Hence, when an air navigation service provider, an authority or an industry needs to make investments for a proposed change, it is important to know whether this change is the best of all alternatives. What's missing is the ability to compare the effects of the different proposed changes.

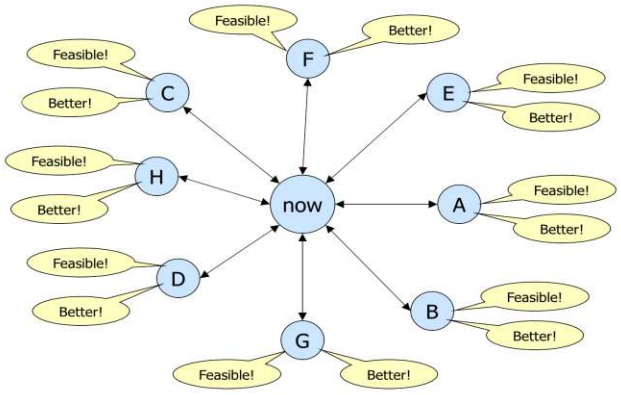

Figure 1 Illustration of the dilemma of a stakeholder when faced with incomparable ATM studies.

Comparability of results requires the following aspects to be common (or at least very similar) between studies:

- Common traffic scenarios
- Common metrics
- Common simulation tools

In this paper, a shareable simulation tool is proposed as a first step to achieve this, as this could provide a self-propelled, bottom-up approach to the other two aspects, the sharing of metrics and the sharing of scenarios. The aim is to provide an open-source and open-data multi-platform simulation tool. The proposed simulator, called BlueSky, is created using a high-level programming language with a simple structure, to facilitate expandability of the code. Scenarios are defined with a straightforward, plain-text based language, further lowering the threshold for first-time users.

Restriction to open-data and open-source tools, however, poses significant limitations. Performance data for aircraft simulation, for instance, has always relied on license-restricted sources. The main challenge will therefore be to achieve high fidelity without having to resort to licensed data and software. Similar to Wikipedia in its early days, the approach of the BlueSky project is to find out if this is feasible by trying it.

II. COMPARABLE EFFORTS

There are a number of past studies that have attempted to address the challenge of shareable simulation tools. This section discusses the three most well-known efforts.

A. BADA aircraft data

An example of a comparable approach is the distribution of the BADA (Base of Aircraft Data) models and aircraft performance data by Eurocontrol [10]. BADA is not a complete simulation tool but merely a set of data files together with a description of how this data can be used. The user still needs to implement the models described in the manual. Version 3 of BADA is available for free, although it has a light licensing process. As a result, it has been used by many, also academic, studies. This broad adoption facilitates comparison of results, and the reference to the BADA models explains to every reader which models have been used. This makes BADA a very successful example of the sharing approach. Its successor, BADA 4, has chosen a different approach: it does rely heavily on commercial data to improve the quality compared with BADA 3. As a result the licensing rules are more strict and inhibiting on a per project basis.

B. CASSIOPEIA project

The CASSIOPEIA project [1] aims to provide an on-line simulation platform for research into traffic complexity. A user has to translate a use case into the Auto-lead Data Format, open XML-based mark-up language. This approach requires some knowledge of the simulation engine, which is unfortunately not fully open-source, as the idea is not to download the simulation but to run it remotely.

It also has a quite unique approach tailored to a specific problem. It uses a JADEX [2] simulation engine; a tool which extracts so-called 'facts' from JAVA classes. Through definition classes that describe the ATM behavior, it can be used to simulate the ATM system. Messages are exchanged between the agents using the standard of the Foundation of Intelligent Physics Agents [3].

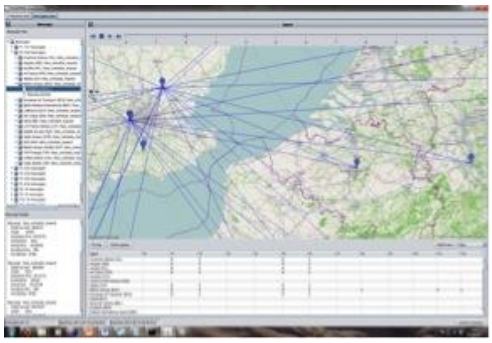

Figure 2 Cassiopeia

The difference between BlueSky and CASSIOPEIA is the open-source and ease-of-use approach. With the BlueSky project, the goal is to be able to download the complete simulator and run it on any computer, even without an internet connection. The CASSIOPEIA project asks you to provide an input file, which will run on the server with the simulation. CASSIOPEIA has a larger focus on, and hence requires more, computer science knowledge.

As this is a server-based closed-source project, extending the program with, e.g., custom conflict detection & resolution algorithms is not possible. Because of the use of the complex formats used for the input files, also for merely using the tool it already requires knowledge of the JAVEX simulation set-up, without having the actual source.

C. ELSA project

Closer to the philosophy of the BlueSky project, but more limited in scope, is the ELSA project: an agent-based open ATM simulation [4]. The 'agent-based' aspect is not unique to ELSA, as it is a characteristic of any ATM simulation, also of BlueSky.

Figure 3 ELSA Air Traffic Simulator

The unique aspect of ELSA is that the focus is on investigating traffic complexity metrics using scenarios which consist of flight plan inputs. Due to this scope there are also some limitations to the scenarios. ELSA does allow downloading the Python source code. This makes it easier to understand and to use than CASSIOPEIA. However, it still requires some knowledge of the simulation set-up as it currently still lacks a simple and intuitive user interface for input and output. ELSA is also still platform dependent as it uses compiled C-code next to the Python part. This makes it slightly harder to use the tool, but it is a definitely a promising approach with many similarities to BlueSky.

III. BLUESKY SPECIFIC APPROACH

The aim of BlueSky is to develop a fully portable, freely downloadable ATM simulator, with an easy to use graphical user interface. As the target users are ATM researchers, it should not require a lot of knowledge of computer science: files should be simple, editable text files.

None of the existing simulations have used the fully open-source, open-data approach to achieve a user-friendly, high-fidelity, versatile air traffic simulation program, which is comparable with commercial tools like AirTop [5], SIMMOD [6], NARSIM [7], etc. BlueSky aims to achieve a level of user-friendliness and fidelity that is comparable to these commercial off-the-shelf tools for ATM simulation.

To set up a simulation in BlueSky, users only need a basic knowledge of some commands to create and control traffic. These commands can be entered during run-time in the console, or they can be provided in a previously created scenario text file. The result can be viewed during run-time on a radar-like display. The use of a simple scenario language TrafScript (see section VIII) lowers the threshold for new users. The file format used is plain text. No further knowledge of simulation engines or file formats is required.

The main target community is academia, but the tool could be useful for any ATM researcher. In order to achieve adoption by the community, two factors are important. First, for unrestricted accessibility, the simulation tool should be license free: there should be no limitations of the use of the data, source code and results. Second, it should be easy to use: scenario definitions should be kept simple, and the source code should remain readable, so that users can make adaptations, without the need for ICT experts.

Summarizing this with an example: a foreign, stand-alone PhD student with a simple computer and only aeronautical knowledge should be able to use it within the first hour after downloading it for scenarios which are relevant to him/her.

IV. OPEN-DATA

The challenge of using open-data exclusively is to achieve a sufficiently high quality without using commercial or proprietary data. Especially aircraft performance data are commercially sensitive. An initial belief in the success of this approach was obtained by the promising results of recreational flight simulator communities, who are in many respects facing a similar challenge. Both commercial as well as open-source PC flight simulator communities like flight gear, X-plane, and DCS have shown that what was initially meant as recreation often achieved a quality sufficient to be used in professional training [8] [9].

The main data as required for air traffic simulation can be divided in two main categories:

- *Navigation data*
  - Geographical Information
  - Navigational Aids
  - Waypoints
  - Airports
  - Sector boundaries
- *Aircraft Performance data*
  - Drag polar
  - Engine performance (Thrust, Fuel Consumption)
  - Operating Weights
  - Autopilot/Autothrottle settings (Bank angle, Mode logic settings)
  - Procedure speeds

A. *Navigation data*

Most navigation information is available on the internet, albeit scattered. As required by ICAO, it is published as part of the Aeronautical Information Service (AIS) by the authorities for use by the public.

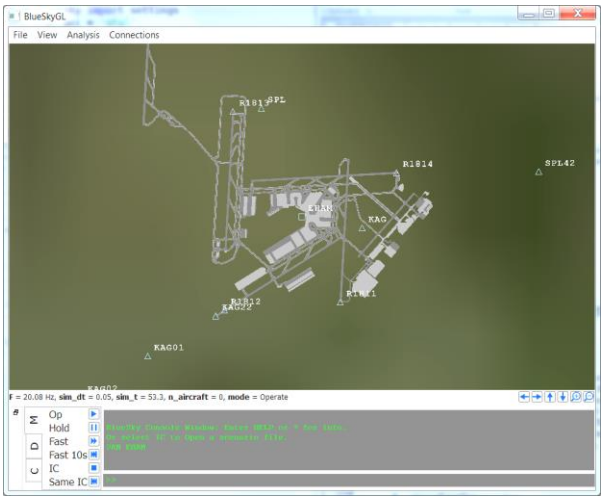

Figure 4 Global airport data is provided by BlueSky

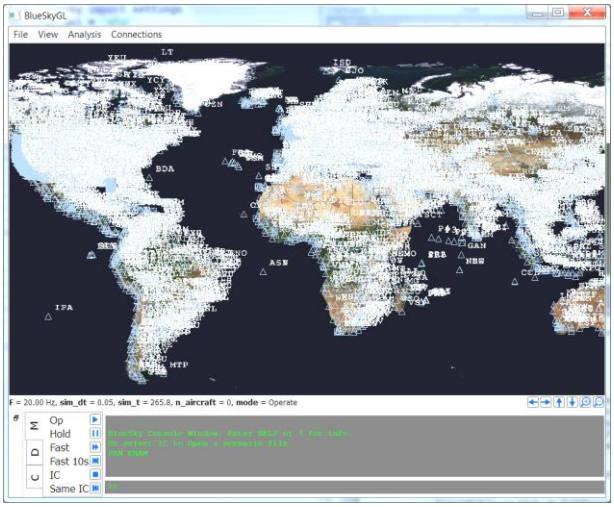

Figure 5 Global coverage of navigation data illustrated by zooming out in the user window (white areas are labels of included navaids)

Commercial packages derive their value form maintaining, updating and collecting this information. Using modern web crawling tools, this process can be replicated. For BlueSky, the

browsing of these websites was automated using a Python Script. The result is stored in text files, which are provided with BlueSky. The BlueSky simulation program reads these files during start-up. Geographical data such as terrain information, coastlines, and borders are collected and used in a similar way.

B. *Aircraft performance data*

BlueSky is compatible with BADA 3. When the BADA 3 files are saved in the coefficients data folder, BlueSky recognizes and uses these files for aircraft performance. Eurocontrol provides licenses to most researchers for using BADA 3 aircraft performance models. BADA 3 has some limitations, for example in modelling thrust in relation to speed and in estimating the fuel flow [11]. BADA 4 has a higher fidelity, at the cost of computation speed, as well as strict license requirements.

In addition, two parallel approaches are employed in the BlueSky project to develop fully open-source and open-data aircraft performance models:

- Conceptual Design Methods: building models using methods used in the conceptual design phase of new aircraft types with data publicly available on the internet on aircraft and engine specifications.
- Machine learning: performance models are built from large amounts of historical ADS-B data, received world-wide, using machine learning methods.

In both approaches the knowledge of the physics of flight will be used to structure the models. The fact that ADS-B does not provide sufficient information to estimate the complete state of the model can be compensated by the large amount of data: the characteristics of the variations caused by winds and different weights, together with the aircraft types will be used to estimate the parameters using big data statistical methods.

Both the input and output of this process are trajectories, as long as they are correct and within the real flight envelope, the most important goal of modelling realistic trajectories has been achieved. One of the costs of ATM is the extra time and fuel due to changes in the route. The fuel consumption is one of the hardest figures to derive using only open data. The order of magnitude can be estimated, but the specifics of individual engine and aircraft types can cause a variation in fidelity. As BlueSky also calculates the amount of energy used, the effect of ATM concepts and tools can alternatively be analyzed which takes away the dependency on the fidelity of individual engine models or aircraft types.

V. OPEN-SOURCE

A benefit of the open-source approach is that it allows the community to contribute to the development, and adapt the program to their specific needs by extending it with the required features. It also has several risks. For example, maintaining an overview of the quality of the program is often a challenge.

A. *Open-source Programming Language: Python*

Python 2 was chosen as main programming language for BlueSky. Python is a very popular, free, and open-source language. Python programs can easily be used on multiple platforms (Windows, Mac, Linux), which rarely requires any modification to the source code.

Python is currently nr. 4 on the list of the most popular high-level languages, directly after JAVA, C and C++ (ref 2015 TIOBE/IEEE top list). Python has the largest academic user community, which shares their modules for free.

Python is both an interactive as well as a scripting language. Using an interpreter in runtime, it does not require any compilation, which makes it completely platform independent. As a scripting language, it lacks the optimization that normally takes place during the compilation of code. For basic operations, which still do require a lot of computing power, such as graphics rendering or large scale calculations, libraries are used which are distributed as Python modules and can be called directly from Python.

Third party libraries like Numpy and Scipy use compiled, optimized, open code and have distributions of precompiled versions for all of the major platforms. Scientific python bundles that include these libraries, such as Python(x,y) [12] for Windows/Linux and Anaconda for Mac, are therefore taken as reference point for BlueSky. Both are acknowledged as standard bundles by the academic community.

Next to Python 2 there is newer version: Python 3. However, Python 3 is the first version, which has broken the downward compatibility. Therefore a vast majority of the Python community is still using Python 2 (>80% as of 2015 [14] BlueSky follows the choice of the Python(x,y) bundle in this respect, following the academic community. Should the majority eventually move to Python 3, only a relatively minor effort will be required to move the BlueSky simulation to Python 3.

B. *License and distribution*

The BlueSky Simulation is distributed under the GNU General Public License Version 3 [15].

The most important features of this license are:

- Required
  - Disclose Source, License and copyright notice, State Changes
- Permitted
  - Commercial Use, Distribution, Modification, Patent Use, Private Use
- Forbidden
  - Hold Liable

The GitHub online repository is used to distribute BlueSky. Searching for the keywords "BlueSky" and "ProfHoekstra" will lead to the required links. One of the benefits of GitHub is the branching option, which allows users to easily develop their own version of the program 'branch') alongside the main BlueSky version, which in a later stage can be merged with the main version ('trunk').

C. *Ease of use and Compatability*

To make the program user-friendly, data files will be using the plain text format as much as possible and should be self-explanatory. One of the unique features of the python syntax is that it uses a minimum of special characters, resulting in source code that reads almost like English.

To increase the adoption by the community, the scenario files will be compatible with the traffic manager program used by e.g. NLR and NASA as part of ASTOR simulation [16]. This program uses plain text scenario files with time stamped commands, that can also be entered during runtime by the user, in the console of the simulation program. This simple syntax allows even non-experienced users to generate scenarios. As sharing scenarios is one of the goals of the BlueSky project, a compatible scenario system is also used in BlueSky. For other sources of traffic data, like Eurocontrol's Demand Data Repository (DDR2, or DDR Service), conversion tools to the BlueSky scenario format with the TrafScript language are available.

VI. Modular Structure of BlueSky

Figure 6 shows which modules can be distinguished at the highest level of BlueSky. The modular set-up of the BlueSky allows easy extension of the functionality without the need to fully understand the remainder of the source. The two modules that control the execution of the program are the *Simulation Control* and the *Command Stack* module. The actual traffic simulation is contained in the *Traffic* module. The TrafScript language as used by the user is also used to communicate events or settings between the different modules, further simplifying the adaptation of the program.

The next paragraphs briefly describe some modules to help the understanding of the functionality inside BlueSky.

A. *Simulation Control (sim)*

The simulation control module controls the scheduling of the main simulation loop and the simulation mode. It initializes the other modules in one or more threads. There are two versions of this module: a single threaded and a multi-threaded version. Depending on the choice of the user interface (See sections VI D and VI F) , the appropriate version is called from the main program. the user can choose the first time interactively, or by editing or deleting the configuration file (called *setting.cfg*) whether to use the classic or advanced user interface.

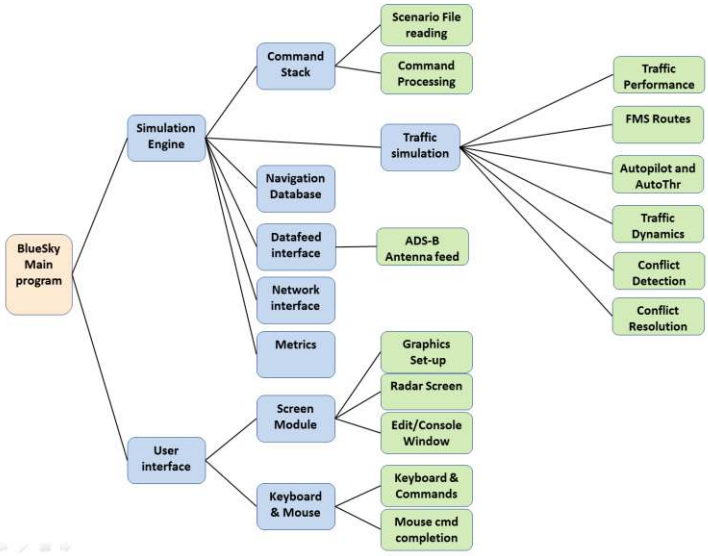

Figure 6 Overview of main BlueSky modules

B. *Command Stack (stack)*

The command stack is used to parse the simulation commands defined in the TrafScript language. It can be seen as the central processing unit of the BlueSky program. The stack receives the command lines entered by the user in the console window of the graphical user interface (GUI) or generated by the mouse clicks in the GUI block. But also other modules can send commands to the command stack via calls to the stack method. Even external programs running on different computers can send any command to the command stack using a network connection. The commands received are processed first-in–first-out during the stack update cycle. As the syntax allows full freedom to control every aspect of the simulation, this provides a high versatility.

Commands are simple text strings. The versatility of text strings means that the communication between different platforms and programs is not hampered by interface definitions and data structure definitions, as these often require an intensive version control. Downward compatibility is also easily ensured. The commands use the ATM scenario control language called TrafScript. This language, also used by NLR, DLR, NASA and several others, is described in detail in section VIII.

C. *Traffic module(traf)*

This object contains data related to the actual simulated traffic: the state of all aircraft, the performance database as well as the navigation data base. There are also systems simulations for the flight management systems (route module),

the autopilot, as well as an ASAS module included in this part of the simulation. Methods to create and control traffic are provided. Many of the calculations and logic dealing with individual aircraft have been vectorized for speed.

D. User interface (UI)

The user interface consists of the drawing of the screens. There are several screens such as the map with the traffic, the console window, in which commands are edited, the menus etc. This module also processes keyboard inputs. Mouse inputs for aircraft selection, lat/lon, heading or menu selection are also handled here. BlueSky comes with two user interfaces (see Figures 7 and 8): the classic GUI and the advanced GUI.

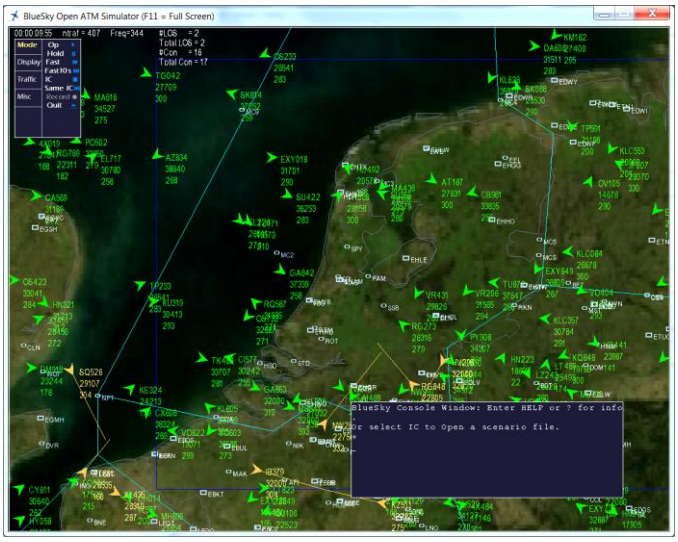

Figure 7 Classic User Interface (high compatibility)

The classic version is based on a wrapper of the SimpleDirectMedia Layer (SDL) called pygame. A second version of the UI uses OpenGL in a Qt windows environment. This frees the CPU of many graphical tasks.

E. Tools module

The tools module contains functions for aerodynamics and navigation calculations, as well as functions for data logging or the conversion of ADS-B feeds from ADS-B receivers to commands to be processed via the stack to visualize traffic using the inputs from an ADS-B antenna.

F. Versions: Classic (Pygame) and Advanced (Qt-GL)

The current version of the BlueSky Open Air Traffic simulation requires Python(x,y) [12] and the Pygame graphics library [13] to be installed beforehand. Selecting "Download as Zip' at the repository is the easiest to install the BlueSky program. This download package contains both versions of the user interface and simulation control. The version is chosen by selecting the main script to run:

- The (default) pygame version (started with Bluesky-pygame.py) will run on any machine but is single-threaded and not as fast as the more advanced version

- The Qt-OpenGL version hands off more task to the graphics card and is multi-threaded, hence faster, but will only run on relatively new machines.

As this only affects the simulation engine and some user interface modules, the majority of the source is shared by both versions. It is generally recommended to run the Qt-OpenGL when the hardware, as the performance allows much larger scale simulations to run fast-time or real-time.

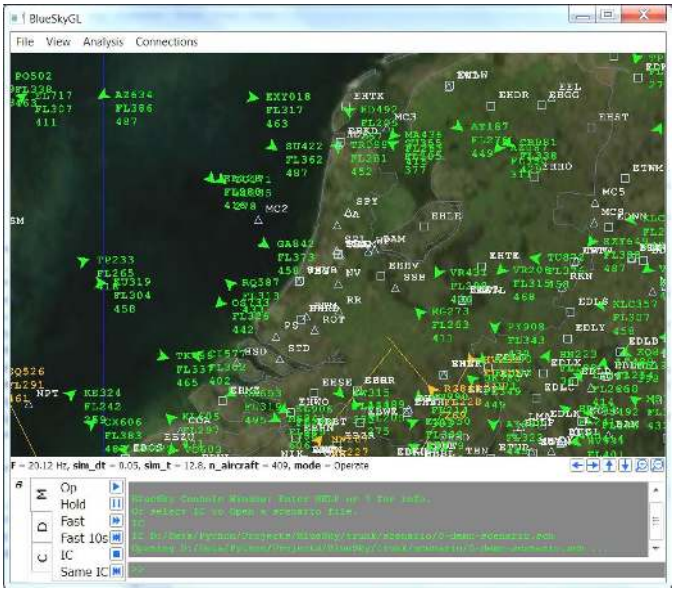

Figure 8 New QT-OpenGL User interface (faster/modern)

VII. DATA STRUCTURE AND VECTORISATION

When we look at the traffic state data as used in an air traffic simulation program as a table:

Table 1: Two directions to bundle traffic data, most programs choose the horizontal (blue) one, while only the vertical (yellow) allows vectorizing

Aircraft	Lat	Lon	Heading	Altitude	Speed
KL204	52.34	-4.84	151	2000	180
MP205	43.12	6.51	125	10000	250
HV307	50.42	9.88	45	24000	350
BA240	52.00	-1.64	220	31000	415
LF229	51.76	0.75	12	7000	250

The data can be organized and viewed in two ways: by first grouping for a row (aircraft) or by first grouping for a column (type of value). The first example, organizing by row or individual aircraft, is most trivial. However, this will not allow

vectorizing the code, while the second does. Therefore column-grouped approach is chosen in BlueSky. The consequences of this choice are explained below. In code, this looks different. Imagine we want to look at the latitude of aircraft *i*.

- Row grouped: looks like traf[i].lat
- Column grouped: traf.lat[i]

As can be seen, this minimal difference in syntax still allows an aircraft (row) based view. In the column based option, an aircraft based look is still possible, and this allows access to all latitudes at once and therefore allows vectorizing the code. This a feature provided by the library Numpy and requires the data to be stored in a Numpy array per column.

Vectorizing replaces loops by expressions which act on complete lists at once. Loops are taken care of at a lower, faster, level. Instead of logical conditions, Boolean logic is used in the expression to take care of for example different autopilot modes per aircraft. In the example below, the autopilot heading is set to different values for all aircraft depending on their mode in one line. In lnav mode there needs to be a flight plan with a number of waypoints larger than zero.

```
aphdg = ((mode==hdgsel)+(nwp<=0))*selhdg +
        (mode==lnav)*(nwp>0)*dirtowp
```

A simple example of vectorizing mode logic of an autopilot heading selection for all aircraft

As can be seen in the example, an important consequence, is that in this case, independent of the current autopilot mode, all variables used in the expression must have a value. So even when an aircraft does not have a flight plan, there should be a default value so that the expression can be evaluated. This is one of the differences with this style of programming. Another one is that it can shorten and thus simplify understanding the code enormously.

The vectorizing approach provides a higher execution speed while it does still allow selecting aircraft depending on conditions like being in a certain area or distance to each other. In complex logical situations, it is simply a different style of programming. The drawback of vectorizing the traffic state data is the creation and deletion requires more lines of code. This is outweighed by the fact that all other operations become simpler using the vectorizing approach. None of this is relevant for the novice user. But it is an important feature to recognize when extending the traffic modelling code with new modules as a contributor to the core program.

VIII. TRAFSCRIPT LANGUAGE

TrafScript is the language which is the Air Traffic Management Simulation scenario language used in BlueSky. It is based on the syntax as used and developed for TMX [17] [18] This language, called TrafScript, is described briefly in the following paragraphs.

Table 2 Explanation of the TrafScript commands and Simulation Control commands

Command	Meaning
TrafScript	
CRE acid,type,lat,lon,hdg,alt,spd	Create an aircraft
HDG acid,hdg	Given an aircraft a heading command (use heading in degrees)
LEFT acid,hdg	Given an aircraft a 'turn left' command (use target heading in degrees)
RIGHT acid,hdg	Given an aircraft a 'turn right' command (use target heading in degrees)
ALT acid,alt	Altitude command for autopilot, accepts feet ('30000') or flgith levels ('FL300')
SPD acid, spd	Speed command, accepts Mach numbers (speeds <1 optionally prefix by an M) and CAS in kts
DEST acid, airport/lat,lon	Add a destination as the end waypoint of the route
ORIG acid, airport/lat,lon	Add an origin as the begin of the route
ADDWPT acid,wpid,[alt],[spd],[afterwp],[rta]	Add waypoint to route in Flight Management System (FMS), optionally
DELWPT acid, wpid	Delete a waypoint from the FMS route
LNAV acid/*,ON/OFF	Switch lateral navigation mode of FMS on or off
VNAV acid/*,ON/OFF	Switch vertical navigation mode of FMS on or off
DIST lat,lon,lat,lon	Calculate the distance bewteen two positions (e.g. with mouse)
MOVE acid,lat,lon,[alt]	Reposition aircraft
DEL acid	Delete an aircraft
MCRE n,type/*,alt/*	Create n aircraft in area visible on display, optionally for specific type or altitude
MDEL lat,lon,lat,lon	Delete all aircraft within given area, specified by two corners
Simulation Control commands	
IC scenfile	Reset simulation and optionally load a new scenario file
SAVEIC scenfil	Save current situation as the start of a new simulation in scenario file
HOLD	Pause simulation
OP	Goto to OPeration mode: resumes simulation
FF [dt]	Fastforward mode optional time duration in simulated time
When an aircraft id is used as argument, it is allowed to swith it with the command,	
So: ALT KL204,FL070 is synonymous with KL204 ALT FL070	
Many arguments such as positions, aircraft, headings, waypoints etc can be clicked into the edit window	
When an argument is left blank, you can use commas to add later arguments	

A sample of a console interaction with BlueSky is given below and continued on the next page.

```
> CRE ?
CRE acid,type,lat,lon,hdg,alt,spd
> CRE KL204,B744,52,4,  90  FL120  250
> KL204 HDG 270
> ALT KL204 FL070
```

```
> KL204 ADDWPT SPY
> PAN EHAM
```

As can be seen each command line has the following structure of command + optional arguments:

Command *[separator]* **argument1** *[separator]* **argument2**

The separator can be a space or a comma. Multiple space always count as one separator, while multiple commas can be used to skip optional arguments. So twice a comma means an empty string is parsed as the argument in this location on the command line. The TrafScript language is not case sensitive: upper and lower case can be mixed freely without any effect.

The command syntax is explained by some examples in Table 2. The command module also reads the scenario file. This is a simple text file with time stamped commands which are executed when the simulation time matches the time stamp in that line of the file. An example of a part of a scenario file is given below.

```
00:00:00.00>PAN EHAM

00:00:00.00>CRE KL204,B738,N52'18'58,E4'46"47,180,FL120,300
00:00:00.00>KL204 ORIG EHAM
00:00:00.00>KL204 DEST LEMD
00:00:00.00>ADDWPT KL204,LEKKO,10000,250
00:00:00.00>ADDWPT KL204,WOODY,,0.55
00:00:00.00>ADDWPT KL204,CIV,,0.84
00:00:10.00>CRE MP205,B744,52.0123,4.000 , 270,10000, 250

# Martinair plane
00:00:20.00>MP205 HDG 300
00:00:20.00>MP205 SPD 280;MP205 ALT FL70
00:00:30.00>HDG MP205,270
```

Example of a scenario file (plain text file)

IX. Conclusion

Based on the results so far, the chosen approach is promising. BlueSky is now a full-blown, user friendly air traffic simulator which can run at very high update rates for a high number of aircraft. On a standard PC running a simulation with 600-800 aircraft flying simultaneously is not a problem. Using the multi-threaded Qt-OpenGL version a larger number of aircraft can be simulated simultaneously.

The program has been by students and researchers for research in several topics: air traffic complexity metrics, upstream delay absorption, path planning and conflict detection & resolution algorithms.

Future work will be aimed at: functionality, improving data quality, optimization of execution speed, providing standard scenarios and metrics. Also a community around the program will be built to jointly further develop BlueSky. The website [19] and the Github repository [20] are pivotal in this.

References

[1] Eurocontrol Demand Data Repository, http://www.eurocontrol.int/ddr

[2] A Pokahr, Alexander, Braubach, Lars, Lamersdorf, Winfried; JADEX: A BDI Reasoning Engine, book chapter, Multi-Agent Programming, P 149-174, Springer Verlag, 2005

[3] Foundation of Intelligent Physics Agents website, http://fipa.org/

[4] Bongiorno, C. et al, "An Agent Based Model of Air Traffic Management", Third SESAR Innovation Days, 26th – 28th November 2013

[5] Airtopsoft S.A. company website (2007): http://www.airtopsoft.com/ Belgium, 2007

[6] SIMMOD product page: http://www.atac.com/ITL/simmod-pro.html by ATAC Corporation, US, 2010

[7] Ten Have, J. M. "The development of the NLR ATC Research Simulator (Narsim): Design philosophy and potential for ATM research." Simulation Practice and Theory 1.1 (1993): 31-39.

[8] Craighead, Jeff, et al. "A survey of commercial & open-source unmanned vehicle simulators." Robotics and Automation, 2007 IEEE International Conference on. IEEE, 2007

[9] Koonce, Jefferson M., and William J. Bramble Jr. "Personal computer-based flight training devices." The international journal of aviation psychology 8.3 (1998): 277-292

[10] Nuic, Angela, Damir Poles, and Vincent Mouillet. "BADA: An advanced aircraft performance model for present and future ATM systems." International Journal of Adaptive Control and Signal Processing 24.10 (2010): 850-866.

[11] User manual for Base of Aircraft Data (BADA) Revision 3.12, Eurocontrol Technical Report 14/04/24-44, Eurocontrol, August 2014

[12] Python(x,y) website http://python-xy.github.io/

[13] Pygame website: http://pygame.org

[14] Thomas Robitaille, "Python 3 in Science: the great migration has begun!", http://astrofrog.github.io/blog/2015/05/09/2015-survey-results/ 2015,

[15] Quick Guide to GNU GPL v 3, Published on GNU/Linux Website, http://www.gnu.org/licenses/quick-guide-gplv3.en.html

[16] Liu, Ding-Jen, and William Chung. "ASTOR- An avionics concept test bed in a distributed networked synthetic airspace environment." AIAA Modeling and Simulation Technologies Conference and Exhibit. 2004

[17] Bussink, Frank, Jacco Hoekstra, and Bart Heesbeen. "Traffic manager: a flexible desktop simulation tool enabling future ATM research." Digital Avionics Systems Conference, 2005. DASC 2005. The 24th. Vol. 1. IEEE, 2005

[18] Wing, David J. "Development of a prototype airborne conflict detection and resolution simulation capability." (2002)

[19] BlueSky website: http://homepage.tudelft.nl/7p97s/BlueSky/download.html

[20] BlueSky repository: https://github.com/ProfHoekstra/bluesky