BlueSky: Open Source, Open Data Approach to Air Traffic Simulation

Jacco Hoekstra
Outline

• Goal and context

• Open challenge:
  • Open data
  • Open source

• Future work and work in progress

• How you can participate/contribute:
  • Wiki
  • Easily make your own plug-in
Current status of ATM Science

now \rightarrow A

Feasible!
Better!
Feasible!

Better!

Feasible!

Better!

Feasible!

Better!

Feasible!

Better!
Feasible!
Better!

Feasible!
Better!

Feasible!
Better!

Feasible!
Better!

Feasible!
Better!

Feasible!
Better!
Research lines CNS/ATM group TU Delft

1. **Foundation of ATM research**
   1. Open source tools (ADS-B decoding, ATM simulation, conversion tools)
   2. Open Data (Performance data, scenarios, historic traffic data)
   3. Big Data & Metrics (Complexity, Traffic Flow dynamics)

2. **ADS-B surveillance and applications**
   1. Raw data analysis, surveillance quality
   2. Airborne Applications (ATSAW, CD&R algorithms)

3. **Drone CNS/ATM technology**
   1. Indoor & outdoor navigation and autonomy
   2. Swarming

4. **General Aviation CNS/ATM technology**
   1. Airborne radar & surveillance technology for safety

5. **Schiphol/Mainport related CNS/ATM issues**
Improve science & research in ATM

- **Comparable** research results, same metrics, tools and scenarios

- Research does not thrive under *standardisation*

- But every **researcher needs** aircraft performance models => success
  BADA 3

- **Open Source, free** (also for tools)

- **Open Data, free**
Previous work: BADA, “only” aircraft data

- Data files with performance and procedure data
- BADA 3 vs BADA 4
- Quality BADA 3 could be improved easily
- BADA 4 has more sensitive, proprietary data
- BADA 3 currently default standard for research
- Licensing regime tightens
- Not fully open
Previous work: Traffic Manager (TMX)

- NLR Traffic Manager: Fast-time simulation and real-time simulation
- Batch simulation
- Open architecture, but Visual studio Visual C++ with Fortran core
- Many features and data are proprietary
- Used by many partners of NLR such as NASA, TU Delft, industry and universities
BlueSky is based on these experiences

- For use in academic world (free) and based on sharing (open)

- 100 % fully open data:
  - Aircraft performance data
  - Navigation data

- 100 % open source:
  - Co-development shared via GitHub/ProfHoekstra/BlueSky
  - Should not require any commercial development tools or libraries
  - Quality control?

- Increase adoption:
  - Easy to use: GUI and data/scenarios in plain text files, run locally
  - Readable source code for scientists who are not computer experts
  - Philosophy: no need to understand what you do not need to know
BlueSky GNU General Public License v3

• **Required:**
  • Disclose Source
  • License and copyright notice
  • State Changes

• **Permitted:**
  • Commercial Use
  • Distribution
  • Modification
  • Patent Use, Private Use

• **Forbidden:**
  • Hold Liable

Challenges Open Data vs Quality?

**Navigation data:**
- Geographical information
- Navigational Aids
- Waypoints
- Airport data: taxiways, runways
- Sector lay-out/Airspaces

**Aircraft Performance data:**
- Drag polar
- Engine performance
- Operating weights
- Autopilot/Autothrottle settings, mode logic
- Procedural speeds

**Weather data**
- GSF data (tbd)
Open data: navaids

- Web-crawling programs, public data, converted to text data files

- Global coverage for navaids and waypoints

- No guarantee that it is always up-to-date

- Customizable, e.g. taxi routes
Open Data: Geographical information

- Just background for GUI

- Geo-website contains all required info:
  - Coastlines
  - Rivers
  - Borders

- Satellite image used from Google Earth
Open data: Airports

- 14,480 airports included
- Large airports also feature taxiway and runway layout
- Flight simulator and gaming community have people who collect this information as well
Open Data: FIRs, Sectors etc

- Still very limited
- Web-crawling possible?
- Now implemented manually based on scenarios
- Console is not ATCo HMI
- More help needed
Open Data: Aircraft Performance models

- Compatible Plug& Play with BADA v3.12 (nearly open):
  Copy BADA files to empty folder: `\data\performance\BADA`

- Generic Open Data models fully based on Open sources (built-in)

- Big Data effort using ADS-B data to develop comprehensive set of aircraft performance files (built-in)

- Metrics also use generic measure Energy, independent of Fuel Flow
Open Source: choice for platform

• BlueSky simulator made entirely in Python

• Based on standard scientific libraries as defined in Python(x,y) (and Anaconda)

• Python 3.6, also 2.x

• Uses Numpy, Scipy

• For GUIs:
  - Qt, OpenGL (included)
  - and/or Pygame SDL

PyQt5  PyOpenGL
Open Source: choice for platform

- BlueSky simulator made entirely in Python
- Based on standard scientific libraries as defined in Python(x,y) (and Anaconda)
- Python 3.6, also 2.x
- Uses Numpy, Scipy
- For GUIs:
  - Qt, OpenGL (included)
  - and/or Pygame SDL
- No need to understand this!
Open Source: choice for platform

- BlueSky simulator made entirely in Python
- Based on standard scientific libraries as defined in Python(x,y) (and Anaconda)
- Python 3.6, also 2.x
- Uses Numpy, Scipy
- For GUIs:
  - Qt, OpenGL (included)
  - and/or Pygame SDL
- No need to understand this:

```python
from math import *

print ("Hello ICRAT")

a,b,c = input("Give a,b,c":)
D = b**2-4.0*a*c
top = -b/(2.0*a)
if D>=0.0 :
    x1 = (-b-sqrt(D))/(2.0*a)
    x2 = (-b+sqrt(D))/(2.0*a)
    print ("x = ",x1," or x =",x2)
else:
    print ("No solutions")
```
if wpok:
    # Overwrite existing origin
    if self.wpname[0] > 0 and self.wptype[0] == self.orig:
        self.wpname[0] = name.upper()
        self.wptype[0] = wptype
        self.wplat[0] = wplat
        self.wplon[0] = wplon
        self.wpalt[0] = alt
        self.wspsd[0] = spd
        self.wpflyby[0] = self.swflyby

    # Or add before the first waypoint in route
    else:
        self.wpname = [name.upper()] + self.wpname
        self.wptype = [wptype] + self.wptype
        self.wplat = [wplat] + self.wplat
        self.wplon = [wplon] + self.wplon
        self.wpalt = [alt] + self.wpalt
        self.wspsd = [spd] + self.wspsd
        self.wpflyby = [self.swflyby] + self.wpflyby

    self.nwp = self.nwp + 1
    if self.iactwp > 0:
        self.iactwp = self.iactwp + 1

    # DESTINATION: Wptype is destination?

elif wptype == self.dest:
    if not (name == traf.id[iac] + "DEST"):  # published identifier
        i = self.navdb.getapidx(name.upper().strip())
        if i >= 0:
            wpok = (i >= 0)
            if wpok:
                wplat = self.navdb.aplat[i]
                wplon = self.navdb.aplon[i]

    else:  # lat/lon type
        wplat = lat
        wplon = lon
        wpok = True

    # Overwrite existing destination
Modules inside BlueSky

Plug-ins

1. BlueSky Main program
2. Simulation Engine
   - User interface
   - Command Stack
     - Scenario File reading
     - Command Processing
   - Navigation Database
   - Datafeed interface
   - Network interface
   - Metrics
   - Screen Module
     - Keyboard & Mouse
     - Mouse cmd completion
   - Graphics Set-up
     - Radar Screen
     - Edit/Console Window
   - Traffic simulation
     - ADS-B Antenna feed
     - Traffic Performance
     - FMS Routes
     - Autopilot and AutoThr
     - Traffic Dynamics
     - Conflict Detection
     - Conflict Resolution
GUI: Classic (Pygame/SDL based)
GUI: Advanced (Qt/OpenGL based)
Download & set-up
GitHub>ProfHoekstra>BlueSky

- Easiest way to set up BlueSky:
  - Download Python 3.6, pip install: numpy, scipy, matplotlib, pypengl, pyqt5 and pip install pygame

- Now you have the Python environment, then download and install BlueSky, Choose Clone or Download, then Download as Zip:
BlueSky setting up

• Run for BlueSky.py

• The first time (when no settings.cfg exists), answer question: Try 1

No config file settings.cfg found in your BlueSky starting directory!

This config file contains several default settings related to the simulation loop and the graphics. A default version will be generated, which you can change if necessary before the next time you run BlueSky.

BlueSky has several user interfaces to choose from. Please select which one to start by default. You can always change this behavior by changing the settings.cfg file.

1. QtGL: This is the most current interface of BlueSky, but requires a graphics card that supports at least OpenGL 3.3.
2. Pygame: Use this version if your pc doesn't support OpenGL 3.3.
3. Console: Run a console-only version of BlueSky. This is useful if you want to do batch simulations on a remote server.

Default UI version:
Choose yourself which user interface you want

• Delete settings.cfg and run BlueSky.py again

• Or double click on (create a shortcut to)
Scenario file

- Scenario file are basically time stamped input by a user
Scenario file

- Scenario file are basically time stamped input by a user
Scenario file: demo.scn
Run BlueSky:

- In QtGl version: Enter “IC” to select a file or “IC demo” to start demo file (“IC de”+Tab uses file completion) as Initial Condition

- In pygame version: select demo.scn
Language in console and files: TrafScript

- CRE?
  CRE acid,type,lat,lon,hdg,alt,spd
- CRE KL204,B747,52,4,090,FL100,250
Language in console and files: **TrafScript**

- CRE ?
  CRE acid,type,lat,lon,hdg,alt,spd
- CRE KL204,B747,52,4,090,FL100,250

*Click on an aircraft (once to select, double to see status)*

- KL204 HDG 270 *(call sign and command can be reversed)*
- ALT KL204, FL250

- PAN KJFK and zoom in with zoom buttons or gestures on touchscreen
- QUIT stops program
- IC allows selecting a new scenario file

- ADDWPT ?
  KL204 DEST EHAM
  KL204 ORIG KJFK
  ADDWPT acid, (wpname,lat,lon),[alt],[spd],[afterwp]
  • KL204 addwpt SPY,FL240,350

*Flight plan entered like an FMS*

*Also see commands LNAV, VNAV, DIRECT, etc.*
TrafScript: ease of use, compatibility

- **Easy** to use in **console** or in **time-stamped scenario file**

- Scenario files are **text files** can be edited in Notepad or Word

- Scenarios **compatible** with tools like NLR Traffic Manager TMX or NASA LaRC ATOL lab

- **Converters** for Eurocontrol DDR2, ADS-B and FlightRadar24 are in beta phase, other converter can easily be added due to simplicity TrafScript language

- **Scenario generation tools** run external

- Requires only **basic computer skills** from user
Applications so far

- UAVs sense & Avoid algorithms
- **Complexity metrics** in existing traffic demand scenarios
- Simulate and investigate effects of ATM procedures like **Upstream delay absorption**
- **AMAN and XMAN** interference of scheduling with pop-up traffic
- **Visualize ADS-B** data from receiver
- **Conflict Detection and Resolution** studies
- **ASAS/Swarming** algorithms
- Effect of procedures and **airspace structure** on capacity and safety
BlueSky already contains many features

- FMS and autopilot logic
- ASAS/ATC simulation for conflict detection and resolution (different modules)
- Extensive scenario control
- External connections
- Batch simulation on multiple CPUs and multiple PCs, very fast, thousands of aircraft (QtGL version only)
- Metrics & data logging
- Visualisation: traffic samples, ADS-B plug-in
Plug-ins

• Plug-ins can access all data inside BlueSky

• Plug-in can send commands to all modules inside BlueSky (using TrafScript commands)

• Can call and contain any modules thanks to open structure of Python

• Can contain/call built-in, customizable datalogger

• Simply define init, update etc.

• Examples provided with BlueSky
Your help is needed

- Fully open source, so help us:
  - **Missing** data in GUI/model: *sectors, airspaces, weather/wind data*
  - **Debugging**: send reports
  - Extension: *request us to add functionality* you need
  - Extension: add your own *functionality in a fork*
  - Wikipedia philosophy: open source and forever beta
  - Documentation: See wiki on GitHub

[https://github.com/ProfHoekstra/bluesky/wiki](https://github.com/ProfHoekstra/bluesky/wiki)
Plug-ins

- Plug-ins can access all data inside BlueSky
- Plug-in can send commands to all modules inside BlueSky (using TrafScript commands)
- Can call and contain any modules thanks to open structure of Python
- Can contain/call built-in, customizable datalogger
- Simply define init, update etc.
- Examples provided with BlueSky
The End Goal of BlueSky

- A fully open source, open data, with extensive features but still easy to use and develop by everyone without any restrictions or licensing

- Exchange metrics (common measures)

- Exchange scenarios (reference set will be developed in AHMED project)

- Stand on the shoulders of your fellow researchers (giants?)

- Join the BlueSky community

- Experimental.....
BlueSky
Open Source, Open Data
Air Traffic Simulation

- Real-time & Fast-time Air Traffic Simulation program
- Runs on Windows, Mac OS or Linux PCs
- Free Downloadable Complete Program (with all sources)
- Documentation, online Wiki and Help functions provided
- Runs stand-alone on standard laptop, on multiple processors or on a PC cluster with parallel simulations
- Global coverage (open navigation data included)
- Contains open source performance models
- Compatible with BADA 3 (just put downloaded files in the empty data folder)
- Easy to use graphical user interface, both text input and point & click
- Contains library of algorithms for conflict detection and resolution
- Simple scripting language in console or text files (using time stamps)
- Script/console controls all functions including flight plans, FMS, autopilot, environment
- Logging functions can be fully customized
- Plug-ins: Template and examples plug-in for easily adding your own functionality. Plug-ins can access all internal data and control the program also by generating console commands internally, e.g. make your own ATC system, CDMR system or autopilot functions

# Example scenario file: (c console commands with time stamp in a text file)
00:00:00.00 < CRE KL204 9744, N52.20/10, E004.29/10, 270, FL070, 250
00:00:00.00 < KL204 HG 280
00:00:00.00 < KL204 DEST LGP
00:00:00.00 < KL204 ADD WP TFK, FL300
00:00:00.00 < KL204 DEST JKFK
00:00:10.00 < KL204 UNAV ON
00:00:10.00 < KL204 VNAV ON
00:00:10.00 < KL204 DIRECT FK
00:00:12.00 < CRE MRD, A821, S170, 2000, 2500

Tech Specs:
- Downloadable from GitHub.
  Use the following link:
  http://github.com/ProfJMHoekstra/BlueSky
  or Google: “BlueSky open source air traffic”
- Check out online wiki:
  http://github.com/ProfJMHoekstra/BlueSky/wiki
- Install a scientific bundle for Python 3.x e.g.
  from Enthought (free for academia) or Anaconda. BlueSky uses NumPy, SciPy, Matplotlib, Qt, OpenGl, and optionally pyqtgraph. All can be installed manually as well
  e.g. by using “pip install module”, when you do not use a bundle. Python 2.7 or higher is recommended. But BlueSky will also run on Python 2.7
- Occasionally, a compiled version (without the sources) are also released, see “releases”, but directly running the actual sources is recommended as they are updated continuously

Thursday Dec 14th
BlueSky Meeting & Workshop/Hackathon

Thursday 14th December 2017 at TU Delft, Kluiverweg 1, Delft

What: Open Source, Open Data Air Traffic Simulation Workshop

When: Thursday December 14th 2017

Where: TU Delft Faculty of Aerospace engineering
Kluiverweg 1, 2629 HS Delft, Netherlands

How: Register by sending an e-mail with your name and affiliation to:
J.M.Hoekstra at-symbol tudelft.nl

For whom: Users, potential users, developers and people interested in open source ATM simulation

09:00 Walk-in with coffee
09:30 Opening

10:00 Meeting BlueSky User & Interest group
- To exchange ideas for current developments and future developments or fixes.
- Potential users are also welcome to attend.

11:30 General Overview BlueSky status
- Overview of current functionality
- Update on recent developments

12:00 Demo’s and Installation help

13:00 Lunch

14:00 Interactive Workshop in three parts
a) TrafScript: How to control your simulation and scenario
  - Routes, guidance
  - Advanced options like multi-processors, distributed parallel scenarios
  - Automatic generation, etc.

b) Making a plug-in:
  - Start your own plug-in (based on plans of users)

c) Data logging in BlueSky:
  - Existing logging facilities
  - Customize logging to suit your needs: create your own data logger

17:00 Drinks