Journal of the Society of Korea Industrial and Systems Engineering
Vol, 26, No. 3, pp.74－78, September 2003.

Study on Developing a Flight Data Visualization

Dr. Jae-Sug Ki
Visual System Research Dep. Of KCEI
Seoul, Korea

비행 데이터의 가시화에 대한 연구

기 재 석

케이씨이아이 영상시스템 연구소

본 연구에서는 비행 데이터의 실시간 가시화 시스템 개발을 위한 소프트웨어 설계 및 2차원과 3차원간의 인터페이스 설계 안을 사례와 중심으로 제시한다. 제시하는 소프트웨어는 비행체의 운동성등을 관찰하고 분석하기 위한 시뮬레이션 소프트웨어로 실시간으로 사용자가 원하는 정보를 도시할 수 있도록 설계 된 것이 큰 특징이며, 손쉽게 사용자의 요구를 적용할 수 있도록 유연성을 높게 모듈화 하였다.

1. Introduction

Data analysis is one of the main processes in development and training of flight vehicles. Several flight data visualization tools are already developed [6, 7, 8]. However, target users of these tools make it extremely hard to meet all requirements. While flight school teachers and students want to be familiar with flying environment and the aircraft itself; engineers and scientists like to analyze and post-process flight data to better understand the behaviour of the aircraft.

There are many works that try to solve information sharing between applications [4, 5] : Building an internal data structure, providing necessary tools to interact with the information and managing all applications. While they aim to handle communications for independent information management systems, a similar approach may be implemented in flight visualization data systems :

Instead of focusing the functionality, focusing the application management and provide a framework which will manage all other tools (in our case plugins). The result will be a set of development libraries and sample applications which other users can develop their own applications. This development process is not limited with provided configuration tools, it uses all functionality with the programming extensions.

2. General Requirements

How to effectively manage large terrain models and data sets in a ·a Source

Most of the time data is provided in pre-recorded data files. Special hardware that's installed on the aircraft can record required data in specific formats. These files can be loaded in to flight data visualization tools and used as data sources. It is also possible to use networking to display real-time data. This feature extends the capabilities of the tool to use with flight simulations or in flight test environments where radars are used to collect real time flight data. In both cases data formats (for network packages or pre-recorded files) might be specific for applications or hardware which is used to collect the data.

2.2 Data Processing

Incoming data is named as raw data since there is no processing after the data collection phase. Even data

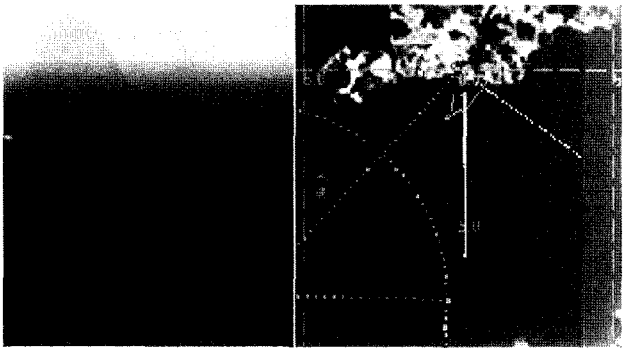
collection hardware may implement some processing, we will still use this term for all data which is just arrived to the flight data visualization system.

Most of the cases, users require simple processing such as unit and/or coordinate conversions. Visual display options may require higher frame rates than incoming data. This can be assured with data interpolation. For example if 3D display is available in the system it will require position data around 60Hz to have a smooth view. If incoming data is lower than this rate, interpolation should be used to provide extra data to the 3D display. Another processing might be calculation of new values which we will call as virtual values. Virtual values are not available in raw data but can be calculated using one or more variables.

Event generation or condition checks are also another problem to solve. Most of the time, users want audiovisual warnings according to conditions such as low speed, high engine temperature etc. A flight data visualization tool may need to have a mechanism to define the conditions and execute them during the mission time.

2.3 Visual Displays

3D display features are one of the essentials of all data visualization tools. For a large range of users this is a good way to understand how flight conditions change during the simulation. Moving map displays also help users to understand how aircraft interacts with the environment and how flight path changes during the flight. Figure 1 shows sample shots for both displays.

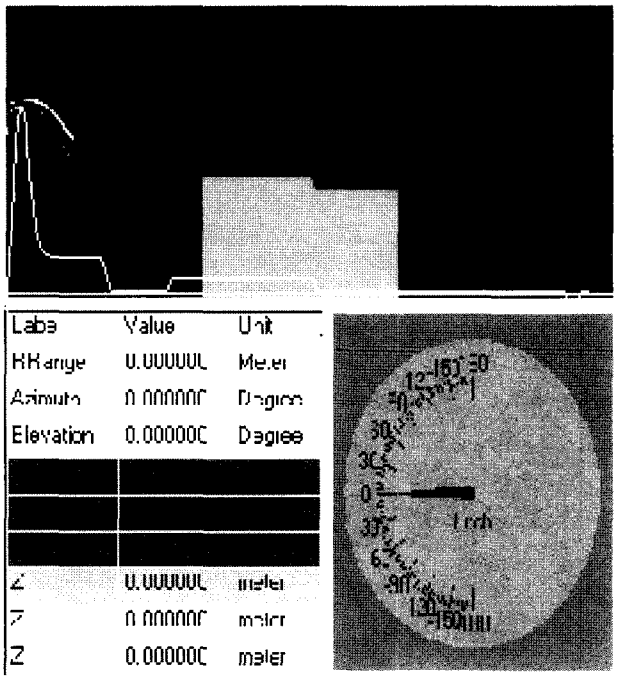


<Figure 1> 3D and Moving Map view of Missile Visualization System

For simple usages, both displays have just visual requirements and not used for scientific measurements. But still users may require additional features to display more accurate data such as Head Up Display (HUD) on 3D display. Common approach is to provide generic displays for all kind of vehicles. This might be enough for some applications but not for all of them.

2.4 Scientific Displays

After all data visualization is for data analysis. Therefore all visualization tools should support scientific displays somehow. Common displays are plots, watch variables and gauges. Figure 2 shows sample images for these displays.



<Figure 2> Plot, watch and gauge display from Missile Visualization System

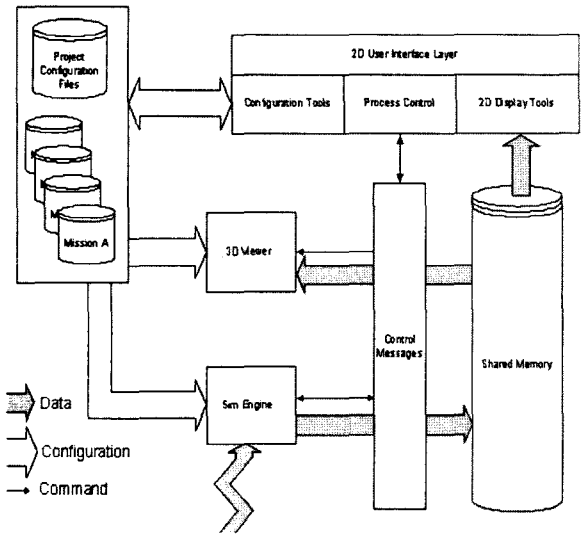
For real time data visualization it is better to use simple format display which allow users to easily understand the data changes in short time. But for further analysis users may need more complex displays to compare different data values on the same plot using different symbols or markings on the display area. Even just for the plot displays it is not easy to match the wide range requirements within a single application.

Scientific displays should be able to display data in real-time performance and they should have extra functionality for after flight data analyzing sections.

3. A Previous Implementation

Missile Visualization System is a set of applications developed by KCEI[1] to use in missile testing. System receives real-time data from network and perform data processing, provides 2D and 3D displays and data recording mechanism. Figure 3 shows the structural layout of the system.

There are three applications working together to meet the system requirements : 2D User Interface is responsible for configuration management and simulation control (start/stop/playback) as well as 2D data displays. Simulation Engine is responsible to receive data packages from network and convert it to internal data structure. And last, 3D viewer manages the 3D display according the configuration.



<Figure 3> Structural layout of missile visualization system.

Those three are standalone applications and work on the same computer. Data communication is based on shared memory. Besides each application sends and receives command messages to maintain the synchronization between processes.

An API is developed for the system which provides necessary functionality to interact with configuration files, shared memory and internal data structures. It also allows users to develop their own applications which can interact with the existing ones to perform extra functionality such as complicated map displays or plots. It is also possible to develop logging or post-analysis applications using the API.

However current system has limitations such as shared memory usage or messaging structure. Currently all applications should work on the same computer. But it is possible to develop some client/server applications which can handle networking with some additional work.

Although system is extendible with additional applications, it is not possible to extend the functionality for the existing applications. In other words, instead of replacing existing plot display, one should write a new application which provides a new plot display.

4. New Design Layout

Extendibility is the key point for our design. Due to the wide range of requirements it is not easy to meet all within a single application. Instead, it is possible to provide a set of tools and a framework which users and other developers can extend the functionality as they require.

The concept of Flight Data Visualization will draw the outlines of the framework. Simple extensions will be developed to prove the design.

Application framework will be based on modular architecture similar with Mathwork's Simulink[2] or Visual Solutions' VisSim[3]. Although both applications are intended to simulate/solve dynamic systems, their structure may be a start point for our design. A similar approach is used by SimAuthor's FlightViz[4] but in a limited manner.

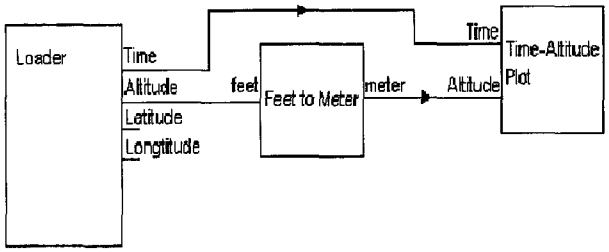
Application framework will provide the base communication between the plugins and manage the simulation. Plugins will provide the main functionality such as data importing, data processing, recording or data display. The unit which performs a particular job in the simulation will be called as a Block. Each plugin may provide several blocks to join the simulation. Application framework is responsible to manage these blocks and connections (channels) between the blocks.

An application programming library named "Processing

1) KCEI is a Korean company focused on 3D computer graphics and simulation systems : www.kcei.com
2) Simulink is a trademark of Mathworks which is used to design and simulate continuous- and discrete-time systems : www.mathworks.com
3) VisSim is a trademark of Visual Solutions Inc. which is used for the modeling and simulation of complex continuous nonlinear dynamic systems : www.vissim.com
4) FlightViz is a trademark of SimAuthor Inc. A visualization and communication system, which instantly creates real-time, intereactive, fidelity graphical depictions of flight and simulator data : www.simauthor.com

Blocks"(pbLib)[5] will be used maintain the basic structure. With this Simulink like structure, users may easily customize their simulation system for their needs. The pbLib provides essential blocks such as simple mathematical operations and some more complex blocks with a "MathPaser" (Mathematical expression solver)[6].

A simple data visualization system consist of two block groups : Data Sources and Data Displays. An user developed data source block can load the flight data file and create its own data plugs to output necessary values. Each data display block receives the data from its own consumer plugs (i.e. Time and altitude plugs for a time-altitude plot). It is also possible to add data processing blocks to perform unit conversions (i.e. Feet to meter conversion for altitude). Figure 4 shows a basic diagram for this case.



<Figure 4> Basic block layout.

While block structure serves as the backbone of the simulation, it has nothing to do with user interface which in this case essential to draw the plot. Actually block structure is designed as a command line tool which is fully configurable via API. Therefore another programming library will be developed to serve as graphical user interface (GUI). QT[7] is chosen as user interface library due to its easy to understand and maintain structure and cross platform nature.

Each plugin which performs user interface functionality will attach the main application and creates its own menus, windows and handles user commands. An API to interact with pbLib will be developed based on QT. Also essential plugins will be developed such as Block Manager to add/remove blocks, connect them each other and change execution order. Main application is responsible for the management of plugins and handles the communication within all plugins. It assures that each plugin will get user request and can perform their response functionality. It also works as real-time controller to manage the user commands such as play, stop, forward, rewind etc.

5. Conclusion

We believe that evolving needs for flight data visualization systems require dynamic and extendible applications. In other words, development process should never end. Configurable interfaces may allow users to customize the functionality but still developers has to work for most complex cases. A toolset which contains re-usable components for display and data interaction will reduce the development time for further development in flight data visualization systems.

Since all functionality is provided via plugins, it is possible to extent current applications or construct new ones according to user requirements. Custom data recorders and exporters may be added to the system to export data for third part applications such as spread sheets and plot drawing tools. Custom processing blocks may be added to the existing system such as the one with built in scripting support. Or extend the capabilities to work on distributed systems.

References

[1] Chris, Mitchell. & Walter, Gekelman. Real-time physics data-visualization system using Performer. Coputers in Physc, Vol. 12, No. 4, July/August 1998, pp 371-379

[2] Dennig, James., Clark, Nicholas., Korthuis, David., Prince, Michale. & Kim, Hyun-Soo. Bid Document : F/A-18 Memory unit data visualization project. http : //wonderwoman.cse.msu.edu

[3] Ronald, L. Small., Stephen, D. Lakowske., Jerry, Bresee. & Gerry, Callejo. A future direction in pilot training. Specific Applications in Pilot Training, September 1999, pp 281-285.

[4] Roth, S. A., Lucas, P., Senn, J. A., Gomberg, C. C., Burks, M. B., Stroffolino, P., J., Kolojejchick, J. A., & Dunmire, C. "Visage : A user interface environment for exploring information." Proceedings of Information Visualization, IEEE, San Francisco, October 1996, pp. 3-12.

5) pbLib is an extendible library which allows users to construct their systems with blocks. More info can be found at www.machsim.com

6) MathParser is an extendible mathematical expression parser which parses and solves expressions in text format. More info can be found at www.machsim.com

7) QT is Trolltech's object oriented, cross-platform, C++ GUI development kit. More info can be found at www.trolltech.com

[5] Rouff, Christopher. & Robbert, Mary Ann. Developing the cooperative mission development environment. ACM International Conference on Supporting Group Work, Phoenix, AZ, November 1997
[6] SimAuthors Inc., FlightViz, www.simauthor.com
[7] Spirent Systems, GRAF-VISION Flight Data Animator, www.spirent-systems.com
[8] SystemWare Incorporated., FDAS, www.sysware.com