AFRL-VA-WP-TP-2003-317

DEVELOPMENT OF A LOW-COST SIMULATOR FOR DEMONSTRATION AND ENGINEER TRAINING

R. Scott Burns
Matthew M. Duquette
Joseph B. Howerton
Richard J. Simko

JULY 2003

Approved for public release; distribution is unlimited.

AIR VEHICLES DIRECTORATE
AIR FORCE RESEARCH LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7542

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE ***(DD-MM-YY)*** July 2003	**2. REPORT TYPE** Conference Paper Preprint	**3. DATES COVERED** ***(From - To)***

4. TITLE AND SUBTITLE
DEVELOPMENT OF A LOW-COST SIMULATOR FOR
DEMONSTRATION AND ENGINEER TRAINING

5a. CONTRACT NUMBER
In-house

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
N/A

6. AUTHOR(S)
R. Scott Burns
Matthew M. Duquette
Joseph B. Howerton
Richard J. Simko

5d. PROJECT NUMBER
N/A

5e. TASK NUMBER
N/A

5f. WORK UNIT NUMBER
N/A

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Control Simulation and Assessment Branch (AFRL/VACD)
Control Sciences Division
Air Vehicles Directorate
Air Force Research Laboratory, Air Force Materiel Command
Wright-Patterson Air Force Base, OH 45433-7542

8. PERFORMING ORGANIZATION REPORT NUMBER
AFRL-VA-WP-TP-2003-317

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Vehicles Directorate
Air Force Research Laboratory
Air Force Materiel Command
Wright-Patterson Air Force Base, OH 45433-7542

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL/VACD

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-VA-WP-TP-2003-317

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
To be presented at the AIAA Modeling & Simulation Technologies Conference, Austin, TX, August 13, 2003.
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

The modification of an existing open-source flight simulator can be a useful training tool for new simulation engineers. New engineers can use this software as a starting point in the development of a full-scale flight simulator. The Modeling and Simulation Familiarization Tool (MSFT) is an ongoing project that utilizes open-source software in such a manner. The project requires engineers to design the simulator, modify the source code, and build the physical cockpit. The end products of the MSFT effort are a fully functional flight simulator with a realistic cockpit configuration and engineering training in the areas of simulation structure, computer coding/modification, simulation configuration, and cockpit design.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT:**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** (Monitor)
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	14	Richard Scott Burns **19b. TELEPHONE NUMBER** ***(Include Area Code)*** (937) 904-6567

DEVELOPMENT OF A LOW-COST SIMULATOR FOR DEMONSTRATION AND ENGINEER TRAINING

R. Scott Burns, Matthew M. Duquette, Joseph B. Howerton, Richard J. Simko
Air Force Research Laboratory
Wright-Patterson Air Force Base, OH 45433-7505

ABSTRACT

The modification of an existing open-source flight simulator can be a useful training tool for new simulation engineers. New engineers can use this software as a starting point in the development of a full-scale flight simulator. The Modeling and Simulation Familiarization Tool (MSFT) is an ongoing project that utilizes open-source software in such a manner. The project requires engineers to design the simulator, modify the source code, and build the physical cockpit. The end products of the MSFT effort are a fully functional flight simulator with a realistic cockpit configuration and engineering training in the areas of simulation structure, computer coding/modification, simulation configuration, and cockpit design.

MOTIVATION

Flight simulation requires an understanding of aeronautics, computer science, and electrical engineering. As a result, facilities that develop flight simulators often employ many types of engineers. In the buildup of a flight simulation, developers must be knowledgeable about all components of the simulation in order to integrate the software and hardware elements. Most engineering curricula do not prepare engineers for a career in simulation development, thus initial simulation training is needed to give the engineer a basic understanding of flight simulation. On-the-job cross training is one option for providing engineers with this necessary familiarity.

Simulation software design requires an in-depth knowledge of programming languages and concepts, operating systems, and computer hardware. Aerospace engineers seldom learn more than basic programming concepts as part of their engineering curricula, but simulation requires significantly greater programming knowledge due to the complex flight models, human interfaces, graphics, and sound. Computer scientists and computer engineers, on the other hand, face a different challenge in the modeling of aircraft dynamics. Flight simulations often involve aerodynamics, six-degree-of-freedom equations of motion, and propulsion system models, all of which fall outside of most computer science curricula.

Cross training can benefit all engineers new to flight simulation. Virtual simulations involve more concepts than only aircraft models and computer programming. Most aerospace engineers do not interact with flight controls or instrumentation over the course of their engineering curriculum. As a result, the interface between the pilot and aircraft is relatively unknown by engineers without aviation experience. An understanding of proper cockpit ergonomics is also necessary in the design of a flight simulator. The controls and instrument displays should be arranged in a realistic manner to make the layout representative of a real aircraft. Developing a simulation that satisfies the needs of its users (especially simulations that are developed for specific air crew tasks) is an important aspect of flight simulation.

In general, on the job training is required to supplement most university's engineering curricula. However, some simulations may be too advanced or mission critical for new simulation engineers to begin this training. One solution is to give new simulation engineers an operable, existing simulation project that requires upgrades or the addition of desired features. The engineers can also design and develop a cockpit for the simulation. This training method provides practice with not only flight simulation, but promotes the use of a systems engineering design process to create a deliverable simulator package. It also provides the engineers with an overview of the simulation components involved in more advanced flight simulators.

OVERVIEW

The Modeling and Simulation Familiarization Tool (MSFT) was developed as a low-cost training device for engineers new to the modeling and simulation (M&S) field. MSFT consists of a product-focused project with customer-based requirements and deadlines, resulting in the development of a standalone aircraft simulator. The project provides training in simulation buildup, cockpit design, and the engineering design process.

MSFT was specifically conceived to train six engineers at the Air Force Research Laboratory's Air Vehicles Directorate. The engineering team consisted of four aerospace engineers, an aerospace engineering co-op, and a computer science co-op, all having limited experience in the M&S field. MSFT was developed to train the team on the fundamentals of M&S through the development and construction of a simulator package that would be used for a real world application. The project followed a standard simulator design process including the design and construction of a cockpit, modification of a simulator software package, and system test and evaluation.

The engineering team was tasked with developing a low-cost combat flight simulator package for an Air Force recruiting squadron. The project began with the definition of the customer's requirements, which principally consisted of a durable, transportable flight simulator with software easy to maintain, control, and operate. After defining the requirements, the team developed designs for the physical cockpit and the simulator software. During the development the customer was briefed on the progress of the project and design changes were made to the software as well as the cockpit based on customer feedback, resulting in an iterative design process. Upon completion of the cockpit and software package, the simulator package was tested to ensure proper operability. MSFT provided an environment for a small group of engineers to develop and design a standalone customer-based project from the start through completion.

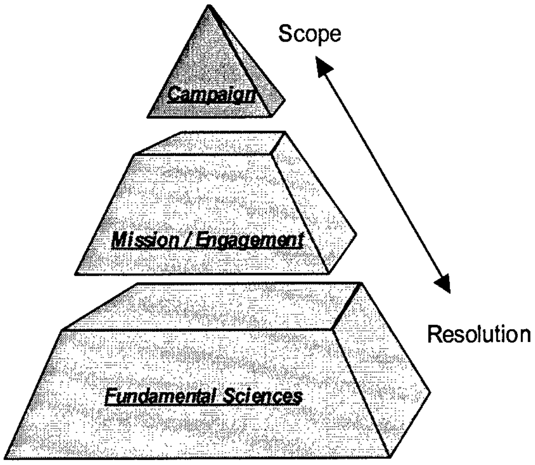

Figure 1 SBR&D Pyramid

Within the Department of Defense, simulation based research and development (SBR&D) is separated into three main levels: fundamental sciences, mission/engagement and campaign. (Figure 1) Simulations that fall under the fundamental sciences involve one aircraft and are typically high in fidelity. These simulations look to measure performance, such as the flying qualities of an aircraft, and are sometimes called engineering simulations. The mission/engagement level includes simulations involving one versus one and many versus many. Mission/engagement simulations are usually lower in fidelity per vehicle than fundamental sciences simulations, but the scope of the mission/engagement is larger. This type of simulation can include head to head combat between different fighter aircraft with pilots in the loop, which can evaluate the effectiveness of one weapon system on another. Campaign level simulations encompass entire battles (large scope), but typically have the lowest fidelity per vehicle. These simulations are best represented by large-scale war games. In general, as the simulations move up the SBR&D pyramid, the fidelity of the simulated systems decreases while the scope of the simulations increases. The MSFT simulation falls in between the fundamental sciences and mission/engagement levels due to the scenario selected by the user.

SIMULATION STRUCTURE

It is through exposure to the various software components that engineers are able to learn the aspects of simulation design. Given the level of complexity that needs to be achieved in a full-featured aircraft simulation, asking engineers who are unfamiliar with simulations to develop the components from scratch would be a daunting task. However, exposing new engineers to existing software that requires modification is more manageable. The MSFT tool approaches the issue of engineer training from the latter perspective. Taking an existing software package and modifying its code to add functionality serves the purpose of training engineers by exposing them to existing methods of simulation.

The MSFT consists of a combination of several software items that work in concert to form the overall flight simulation. Several of these components are open-source software codes that are designed to be portable, extensible, and relatively easy to integrate into larger software suites. The major components are shown in Figure 2 and are explained as follows.

Flightgear, an open-source aircraft simulation, formed the basis for the flight simulation.[1] Flightgear is a complete aircraft simulation that can be downloaded from the Internet. Its General Public License (GPL) allows the source code to be freely distributed and modified by anyone. Furthermore, Flightgear is written in C++ with its graphics engine accessing standard OpenGL functions – making its source code easily portable between several operating systems including Windows, Irix, Linux, MacOS, and Solaris.

Flightgear was designed to support several flight models that have been developed over the course of its evolution, namely the UIUCsim, LaRCsim, JSBsim, and Yasim, each having unique features. LaRCsim is a predecessor of JSBsim and is not used by most aircraft models. The UIUCsim was developed to support research at the University of Illinois.[2] YAsim and JSBsim are the two most commonly used flight models, Yasim being a low-fidelity model and JSBsim a higher fidelity model. The MSFT project utilizes JSBsim since the project required a higher fidelity model.

Simgear provides network interface, user input, sound and graphics, a standard set of libraries to compute position based on the WGS 84 model, a sky-dome model, weather, a magnetic variation model, and a standardized interface to design instrument panels. It was envisioned by the Simgear team that a standard set of Simgear libraries could be used in multiple simulations. Not only does Simgear act as the basis for Flightgear, it can be used in other simulations. Like Flightgear, Simgear is GPL licensed.

Flightgear and Simgear use the XML standard for aircraft configuration, simulation settings, and instrument panel configuration. Designing a flight model is a straightforward process with XML input since each variable is clearly defined in the XML syntax. The instrument panel is similarly designed in XML. The Simgear instrument panel system allows for basic OpenGL commands to rotate and translate panel items. Panels consist of layers of RGB textures that are overlaid and manipulated based on the Flightgear property tree.

The Flightgear property tree is an extensible set of variables that are accessed by the simulation. Variables are defined initially in the XML settings files and can be manipulated via code, network interface, or user input.

The portable game library, PLIB, sits below Simgear and acts as a higher-level interface to OpenGL.[3] PLIB includes libraries for 3D rendering, sound and joystick interface, and font rendering. PLIB allows source code to be easily portable between operating systems while maintaining a high level of functionality. Flightgear uses PLIB as the engine for its scene graph generation and 3D model interpretation. The simplicity of PLIB allows advanced simulations without the need for cumbersome coding for 3D transforms or model format parsing.

The MSFT team chose Windows as its development environment. With some additional effort, Flightgear can be compiled in Windows using Microsoft Visual Studio but since Flightgear was and continues to be developed primarily for Unix-based operating systems, the team decided to use Cygwin, a freeware UNIX environment for Windows allowing Flightgear to compile with no changes in its code.[4] Very fast desktop computers are inexpensive and easy to maintain compared to

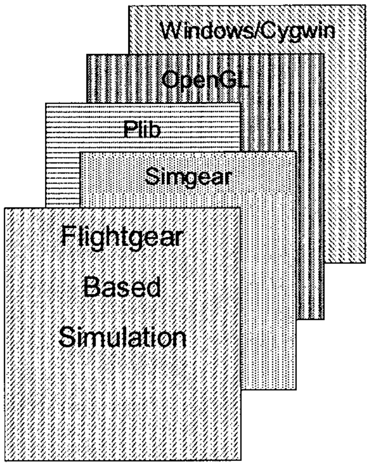

Figure 2 Simulation Structure

proprietary UNIX systems and most engineers are familiar with the Windows operating system. The Windows/Cygwin solution allows Linux-like functionality on a system that users already know – allowing users to become more proficient with simulation design without unnecessarily exposing them to a new computing environment.

CODE MODIFICATION

While learning the inner workings of Flightgear is useful in developing an understanding of simulation, one purpose of this project was to develop a specialized simulation. Specialization of Flightgear to enable combat simulation would require significant augmentation of the existing code. The requirements for the AFRL simulation included

- Enabling multiple moving objects in three-dimensional space.
- Integrating a target designator (TD) and weapons firing control into the Flightgear interface.
- Designing a weapons model that allows for multiple weapons tracking multiple targets simultaneously.
- Modifying the flight model to replicate a modern fighter aircraft.

Flightgear is currently designed for single aircraft operation. A multi-player capability is being developed, but has not yet been implemented. Multiple moving three-dimensional objects would need to be implemented before a combat simulation is feasible in the environment. A simple solution was to create data files with position and orientation data. A data line was then read into the simulation to move the models in three-dimensional space. Routines are present in Flightgear to position three-dimensional objects but the current manager assumes the position is fixed. Currently, there is no multiplayer function in the code, only a single player against "dumb" opponents.

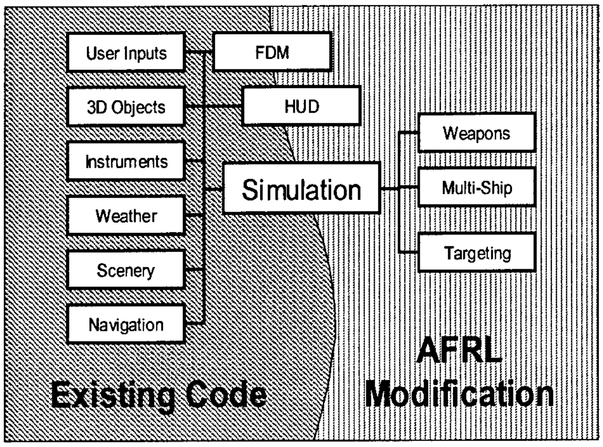

Figure 3 Modification to Flightgear

Adding a target designator (TD) required a modification of the existing Flightgear heads-up display (HUD) code, which was done using OpenGL commands. Routines were designed that translated the TD box in the X-Y space on screen to match the position of the designated target in three-dimensional space. This offered an important lesson to the design team, given that an important factor in virtual simulation design is transforming coordinate systems. Routines were also created that maintained the status of each target in the scenario, the remaining weapons, and control of weapons firing.

The weapons modeling offered an opportunity to design a simple six-degree-of-freedom flight model and target-tracking model. The missile flight model was based on modern short-range infrared guided missiles. Each missile was fired from a specified starting point (based on a location relative to the pilot's view point) and accelerated to a top speed. At each computation, the missile turned in both azimuth and elevation to match the location of the designated target. A turning-rate limit was applied to better simulate existing missiles. After a specified "burn" period, the missile would stop accelerating and would glide towards its target. The target management system included a "lock-on" notice that would notify the user that the missile was in range. If the missile were fired outside the "locked-on" range, there would be very little chance of a successful hit.

A slight modification of the JSBsim flight model was necessary to better simulate a modern fighter. A delay was added in the throttle routine for spool-up and spool-down of the engine model. Other Flightgear developers have since upgraded the JSB model to include this feature.

SIMULATION CONFIGURATION

Flightgear and Simgear were designed so that configurations are entered through XML files. XML is rapidly becoming a standard in many computing applications and knowledge of it was in itself a useful result of this project. Two important aspects of the simulation required XML-based configuration – instrument panel design and flight model design.

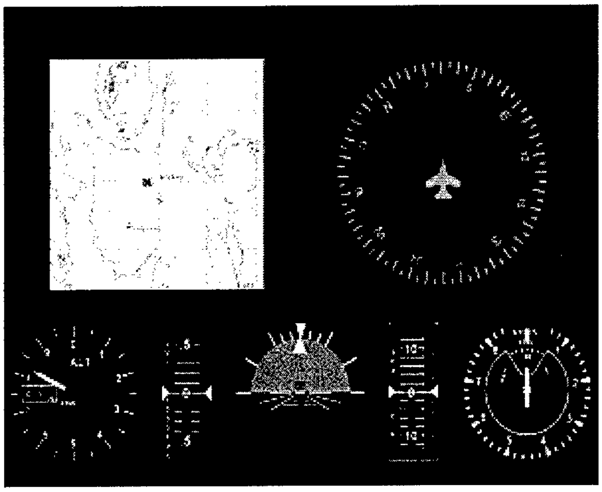

Figure 4 MSFT Instrument Panel

Designing an instrument panel offers engineers the opportunity to learn human interface design and modern aircraft instrumentation. For this project, instrumentation was based loosely on existing instruments in fighter aircraft. Each instrument is tied to a variable or set of variables in the simulation and translated or rotated as appropriate. For engineers who are unfamiliar with aircraft instrumentation, this provides an opportunity to become more familiar with the properties relevant to pilots in a manned aircraft, for instance, the magnetic heading versus true heading; or mean sea level versus above ground altitude.

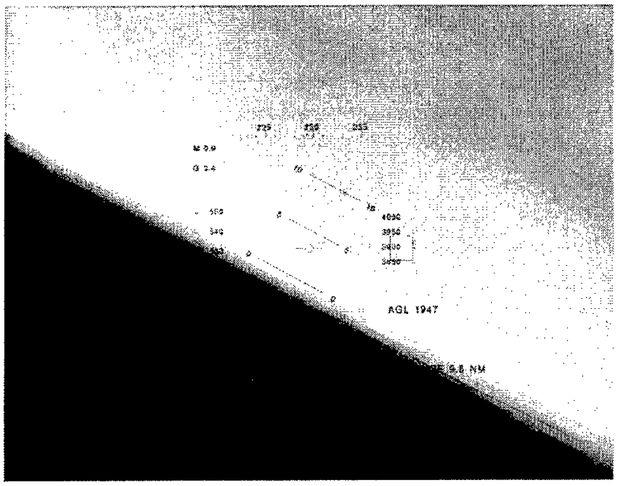

Figure 5 MSFT Out-the-window view

From an engineering standpoint, flight models are more involved and have a larger impact on the overall simulation. For this project, the existing flight model was modified slightly, as described above, to better simulate a turbine aircraft. Required aerodynamic data was then entered into an XML file. This aspect of simulation would be particularly beneficial to engineers with little experience in the aerospace field. Non-linear aerodynamics is inherent to the tabular style coefficient inputs. The standard configuration files allow for modification of stability coefficients, engine characteristics, and weight & balance.

Although the source code was not modified to change terrain and scenery, the scenery generated plays an important role in a virtual simulation. This project allows engineers an introduction to aviation principles such as radio navigation, airport operations, and pilotage (navigation by reference to visual landmarks). From a software development standpoint, engineers become more familiar with aspects of generating a three-dimensional scene. For instance, several new objects were created and placed in the scene, offering users a more realistic experience. Existing terrain was used but integration of new objects required a better knowledge of geographical coordinates.

With all of the modifications in place, the result is a combat flight simulation that offers users a variety of capabilities. An out-the-window view of the simulation is shown in Figure 5.

COCKPIT DESIGN

In addition to flight simulation software, a cockpit is often constructed as part of a flight simulator. For this project, a cockpit was a key customer requirement. The requirements of the project called for a fighter style cockpit, and since cockpit design is relatively foreign to most engineers, this area provided the engineers an opportunity to learn the basics of pilot-vehicle interfaces. There are several important aspects in cockpit design, including:

- Instrument panel layout and position,
- Control stick and throttle style and placement,
- Seat height, recline angle, width and placement,
- Overall cockpit size and construction method,
- Out-the window display method.

The customer requirements for the MSFT cockpit included:

- Transportable by two people,
- Fit through a 30 inch wide door,

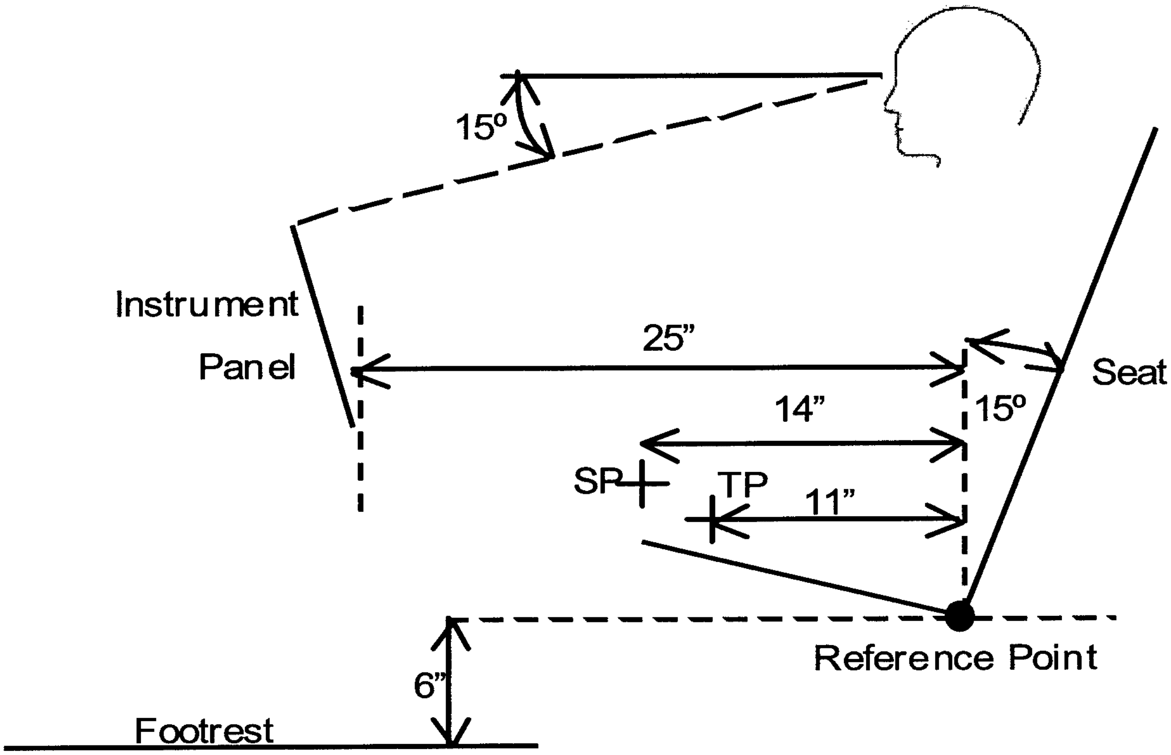

Figure 6 Side view of the cockpit showing key dimensions (all dimensions in inches) SP = Stick pivot point, TP = throttle pivot point

- Easily enter and exit the cockpit seat.

A key challenge in the cockpit design was balancing realism with the customer's requirements. Obviously, an exact replica of a fighter aircraft cockpit would not meet the transportability or use requirements set for this project. In general, tradeoffs are made between realism and usability when designing a simulator cockpit, but given that this was developed as a recruitment tool, realism was an important consideration when designing several aspects of the cockpit.

The basic metrics for seat, instrument panel, and controls layout were developed based on the expertise of in-house personnel as well as dimensions taken from other simulators at AFRL. Figure 6 shows, from a side view, the critical dimensions that establish the fighter cockpit look and feel. Most cockpit dimensions are based on the location of the vertex between the seat bottom and seat back. The reference point is noted in the figure. From this point, the instrument panel, footrest, and stick & throttle are placed. Modern fighter cockpits commonly have seat recline angles of 15 to 20 degrees, elevated footrests, and an instrument panel incline of approximately 15 degrees. The pivot points of the stick and throttle are typically 11 and 14 inches forward of the reference point, respectively. Both pivot points are also typically higher than the reference point, with the stick slightly higher than the throttle. The top of the instrument panel should fall approximately 15 degrees below the horizontal from the average pilot's view. Although these dimensions may vary among cockpit designs, they have shown to be suitable for simulation work conducted at AFRL.

Once the key dimensions were established, customer requirements were re-visited in order to determine the desired levels of functionality and ease-of-use. Since the simulator was to be used for demonstration purposes, it was determined that rudder function was not necessary. This eased the requirements for the footrest design by removing rudder pedals. The seat was built on a sliding platform to allow easy entry to the simulator. This eliminated the need for users to climb over cockpit structure.

The team chose a popular commercial product for pilot controls, the HOTAS Cougar from Thrustmaster.[5] This stick and throttle combination is based on the F-16 and connects to the USB port on a standard PC. The buttons and axes are easily configured via the supplied software.

The instrument panel was also simplified by including a handful of instruments that were based on actual fighter instrumentation. Figure 4 shows the eight MSFT instrument panel instruments: a heading indicator, weapons load-out (or moving map, depending on the flight mode), attitude indicator, vertical velocity gauge, angle-of-attack indicator, altimeter, and airspeed indicator. The simple panel simulated aircraft instrumentation without the unnecessary confusion of a more complicated panel.

The simulator was designed with the idea that the customer may use a variety of different out-the-window display methods. The preferred method was to use a high-resolution LCD projector projecting onto a screen approximately ten to fifteen feet away. This allowed for a large, rectangular viewing area and some sense of motion. Other configurations could include a monitor mounted close to the front of the cockpit or a large plasma TV setup for situations with limited space or brighter lighting.

The final cockpit was constructed of wood, with plastic sheeting forming an outer shell. It split in two sections for transportability and to fit through a standard width door. The instrument panel monitor was a 15-inch TFT display integrated into a removable hood that attached to the front of the cockpit. The seat rolled into position via a set of castors and a guiding plate. A perspective view of the final cockpit configuration is shown in Figure 7.

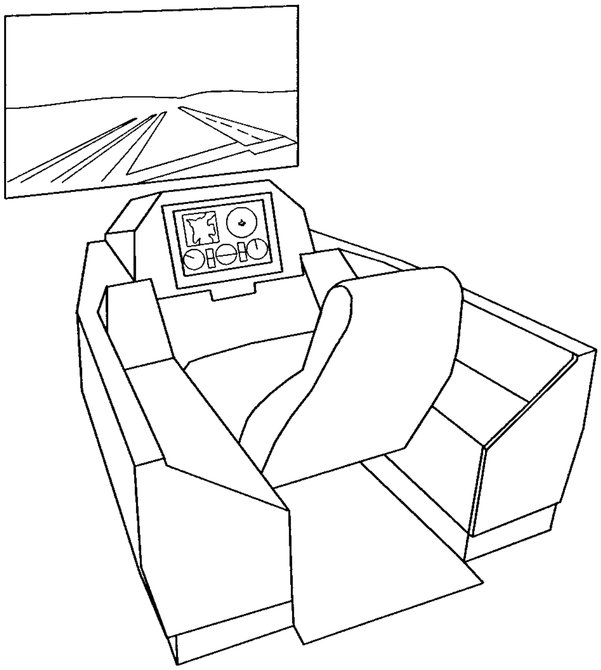

Figure 7 Artist's rendition of the simulator, showing the cockpit and a projected out-the-window display.

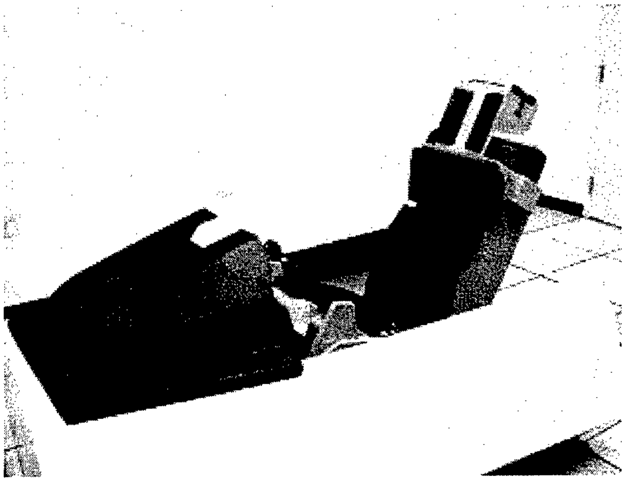

Figure 8 Second generation simulator based on an F-16 cockpit.

FIELD TESTING

After the simulation software was integrated into the cockpit, the system was tested at a high school technology fair. At the fair, the cockpit maintained structural integrity for approximately one hundred users, which initially validated the physical configuration and construction of the cockpit. Overall, the simulation software functioned well, but a few bugs were discovered over the course of testing. Simulator users also gave feedback on the scenarios and simulator functions, which proved very useful in the improvement of the software before its delivery to the customer.

FUTURE WORK

The MSFT team is in the process of developing a second-generation cockpit as part of a follow on project. The second project has a relaxed requirement for transportability in order to focus on increased realism. The MSFT team chose an F-16 inspired cockpit (Viper Pit) by Advanced Computer Concepts, Inc. as the basis for the next simulator.[6] The Viper Pit was modified to suit a user accessibility requirement similar to the first project by attaching a sliding seat mechanism. The result is shown in Figure 8.

CONCLUSION

The flight simulator package was delivered to the Air Force recruiters and it is currently being used at air shows and high school job fairs. The MSFT project served as an effective tool to train new engineers in the principles of modeling and simulation. Not only did the project provide hands-on experience, but it also introduced the team to the concept of balancing customer requirements against technological difficulties. By modifying existing simulation code and tapping the knowledge of more experienced simulation engineers and technicians, the engineers were able to develop skills that are directly applicable to other projects within AFRL.

Projects similar to the MSFT can be beneficial to groups other than professional simulation engineers. Undergraduate aeronautical engineering programs can integrate similar projects into their curricula to serve as a research tool. Currently, the University of Illinois is using a modified version of Flightgear as part of an aircraft icing research project.[2] The MSFT project can also be used as an aviation familiarization tool, since the instrumentation and flight models can be easily modified.

ACKNOWLEDGEMENTS

The authors would like to thank Mr. Mike Denning, Mr. Carl Rickard, and Mr. Martin Darbro of Protobox, LLC. for their continuing support in this project. Also, thanks to Mr. Dewey Meade for providing expert advice in cockpit construction.

REFERENCES

[1] http://www.flightgear.org, May 13, 2002

[2] Sehgal, B., Deters, R.W., Selig, M.S., “Icing Encounter Flight Simulator,” AIAA 2002-0817, Reno, NV, Jan 2002

[3] PLIB website, http://plib.sourceforge.net/, May 13, 2002

[4] http://www.cygwin.com, May 13, 2002

[5] http://www.thrustmaster.com, May 13, 2002

[6] http://www.viper-pits.com, May 13, 2002