PC-Based "MicroSims" in a Distributed Military Simulation Environment

William E. White, Lead MicroSim Engineer
Louis Wharton, Computer Scientist
Dave Kotick, Principal Engineer, Concept Development Integration Lab
Eric Anschuetz, Head, Technology Development & Integration Lab Branch
NAVAIR Orlando Training Systems Division
Orlando, Florida
U.S. Navy

ABSTRACT

This paper will provide insight on how low-cost PC-based simulators can provide meaningful training to the war fighter given current advancements in the video gaming industry. It will also describe a new opportunity to add a Department of Defense (DoD) compatible interoperability interface to PC Aviation Training Devices (PCATD) based on Microsoft's Direct Play technology used by many commercial-off-the-shelf (COTS) flight simulators. This paper will attempt to provide a vision of how the PCATD will fit into the Navy's future training capabilities, given the new possibilities for interoperability in a distributed military simulation environment. The initial results of NAVAIR Orlando's interoperability concept study will be presented.

ABOUT THE AUTHORS

Mr. William White is currently working on helicopter-based MicroSims at NAVAIR Orlando Training Systems Division, where he has led the effort to add an interoperability interface to PC-based commercial flight simulators. Mr. White has over 14 years experience with the Navy in real-time programming and embedded systems. He has developed technology and written several papers for the Joint US/UK Surface Ship Defense anti-torpedo program. This program has received several patents. Mr. White has also developed software to add an autopilot capability to the Navy's hovercraft fleet (LCACs). Mr. White holds a B.S. in Electrical Engineering from the University of Puerto Rico and has taken many advanced courses in Computer Engineering at New York University and Florida State University.

Mr. Louis Wharton is a Computer Scientist in the Modeling and Simulation Laboratory Branch of NAVAIR Orlando Training Systems Division. He is currently researching the feasibility of developing interoperable, low-cost simulators, based on commercial-off-the-shelf (COTS) gaming applications, in a distributed environment. He holds a B.S. in Computer Science from Florida Atlantic University.

Mr. David Kotick serves as the NAVAIR Orlando Concept Development and Integration Laboratory (CDIL) Principal Engineer. Mr. Kotick has over 20 years of experience in the Navy Research and Development arena. His works have been published extensively in technical journals, conference proceedings and NAVAIR technical and special reports. Mr. Kotick has co-written numerous papers dealing with complex subjects such as training system interoperability and digital communication techniques for training. He currently holds multiple patents (pending and granted) in the fields of digital communications and simulation technology. Mr. Kotick holds a Bachelors and Masters Degree in Electrical Engineering from the University of Central Florida. Mr. Kotick is an active and current member in the Experimental Aircraft Association (EAA) and Aircraft Owners and Pilots Association (AOPA). He holds both a Private Pilot and Ultralight Pilot Certificate.

Mr. Eric Anschuetz is the head of the Technology Development and Integration Laboratory Branch at NAVAIR Orlando Training Systems Division, where he has worked for over 16 years. For the past 10 years, he has specialized in distributed simulations and has developed simulation interfaces and gateways for both Distributed Interactive Simulation (DIS) and High Level Architecture (HLA) environments. He has authored or co-authored 16 papers and has played an active role in I/ITSEC joint simulation exercises since 1993. He received a B.S. in both Computer Science and Mathematics from Eastern Michigan University and has taken graduate courses in Computer Engineering at the University of Central Florida.

PC-Based "MicroSims" in a Distributed Military Simulation Environment

William E. White, Lead MicroSim Engineer
Louis Wharton, Computer Scientist
Dave Kotick, Principal Engineer, Concept Development Integration Lab
Eric Anschuetz, Head, Technology Development & Integration Lab Branch
NAVAIR Orlando Training Systems Division
Orlando, Florida
U.S. Navy

1.0 Micro-Simulator Background.

Use of micro-simulators, or MicroSims, in the Navy began with Ensign Herb Lacey, a student aviator at the Naval Air Station in Corpus Christi, Texas. Ensign Lacey spent some of his off duty hours building a T-34 Mentor training aircraft for the Microsoft Flight Simulator (MFS) 98 game and modified its scenery files to add Corpus Christi landmarks and visual references. Ensign Lacey later continued flying his simulated T-34 while attending an aviation preflight instruction course in Pensacola. Ensign Lacey, in effect, had produced a training tool that allowed him to practice procedures learned in the classroom. Ensign Lacey went on to finish primary flight training as one of the top student naval aviators in his class.

The Naval Education and Training program's Assessment Division studied the feasibility of using commercial simulation games as education tools and determined where it could fit in the flight-training syllabus. The Navy also utilized the University of Central Florida's Institute for Simulation and Training to conduct the Micro-Simulator Systems for Immersive Learning Environments (MISSILE) project. The collective research data supported the concept of using PC-based flight simulators in training programs.

In support of these studies, eight PC Aviation Training Devices (PCATDs) based on the Microsoft Flight Simulator and Ensign Lacey's earlier work were delivered to Corpus Christi. For six months data was gathered, including flight performance and test scores. The Navy compared the data against the same data gathered from a control group of students who did not use the simulation application. The resulting conclusions demonstrated that the PCATD does demonstrate a positive transfer of training for crew resource management and aircrew coordination. Also, best training results are achieved when introducing new tasks associated with Familiarization, Basic Instruments, and Radio Instruments early in flight training. The study further concluded that cost and time savings can be achieved through use of PCATDs along with a significant reduction of anxiety in the cockpit and improved trainee confidence. Students tended to meet the standards more quickly (as evidenced by above average scores), and attrition was reduced due to fewer total unsatisfactory scores. Student feedback reinforced the findings of this study. Numerous students reported improvement in problem areas after using the PCATD lab.

The Chief of Naval Education and Training (CNET) pursued MicroSim development in fiscal year (FY) 1998. The effort culminated with the delivery of MicroSims to the Naval Aviation Schools Command (NASC) and Training Wings Four, Five, and Six. The MicroSims delivered were based on the Microsoft Flight Simulator with a training-aid shell added. The NAVY-developed training shell provides scenarios for each of the training wings dependent upon their individual training requirements. The CD-ROM containing the training shell also included flight panels which emulate the T-34C, T-44, TC-12, and the TH-57, as well as flight scenery files for the training sites. Training Wing Four received 10 MicroSim units, Training Wing Five received 14 units, and Training Wing six got 8 units. The systems are currently used on a voluntary basis with limited and inconsistent indoctrination to educate students on their capabilities. These systems operate in a standalone mode and do not have the ability to interoperate with other trainers. Furthermore, the training aid shell has not been institutionalized as a part of the formal curriculum. Usage of the MicroSims is limited to non-training hours and dependent upon individual student motivation and familiarity with the aids.

In a more recent study, conducted by NAVAIR Orlando, the MicroSim Program Office

investigated ways of formally incorporating this technology into the curriculum. The study explored the possibility of replacing the current Aviation Communications/Navigation Training Device (Device 2B47) with MicroSim workstations. The 2B47 device provides extensive task practice in airway navigation, flight task management, fuel and time computations, aircraft maneuvering, real-time position reporting, point-to-point navigation, and aircrew coordination. The study concluded that the MicroSims at Training Wing 6 could support the same learning objectives as the 2B47 device with the addition of some training scenarios. The current 2B47 device is simplified "chair flying", or, in other words, a cognitive two-dimensional environment. It is not a three-dimensional environment with operational flight controls. The study states that the MicroSim environment, with its visual system, instrument emulation, and flight controls offers training advantages over the 2B47 device.

The non-military aviation community, including the Federal Aviation Administration (FAA), has also embraced training with PCATDs. Currently PCATDs are approved for instrument training under FAA Advisory Circular (AC) 61-126, issued on May 12, 1997. Approved PCATDs are distinct from flight training devices (FTD), qualified under AC 120-45 (Airplane Flight Training Device Qualification), and flight simulators qualified under AC 120-40 (Airplane Simulator Qualification). An FAA-qualified PCATD is an instrument training device designed to help prepare a pilot to fly in instrument conditions. A PCATD qualified device can be used as part of an integrated ground and flight instrument training curriculum where it can be used for up to a maximum of 10 loggable training hours in an FAA-approved Federal Aviation Regulation (FAR) Part 141 school, or in an acceptable instrument training curriculum under FAR Part 61.

The FAA recognized early on that qualified PCATDs are highly beneficial when used under the guidance of an authorized instructor to achieve learning in procedural tasks such as departures and arrivals, navigational aid tracking, holding pattern entries, instrument approaches, and missed approach procedures. The Aircraft Owners and Pilot Association (AOPA) Air Safety Foundation (ASF) studied PCATDs for several years in the early 1990's. Their study included a formal test in a controlled academic environment. In 1994, the ASF published the results of their studies which showed that in preparing students for Instrument Flight Rules (IFR) procedures, it was clear that those who had the benefit of PCATDs prior to the lesson learned faster in the aircraft than those who went straight to what they call the world's worst classroom: the cockpit.

Other formal studies sponsored by the FAA and NASA (through Embry-Riddle Aeronautical University and the Institute of Aviation, University of Illinois) concluded PCATDs to be as good as, and in some cases better than other ground trainers The data also indicated that the PCATD is an effective training device for teaching instrument tasks.

2.0 Micro-Simulations in a Military Environment.

Beginning in 2001, NAVAIR Orlando's Concept Development and Integration Laboratory (CDIL) began a concept study to look at the applicability of leveraging previous gaming technology efforts into an interoperable team training environment. The main objective of this concept demonstration was to provide an open standard, bi-directional interface for low-cost, PC-based aviation simulators. Through new software development, the CDIL has demonstrated the ability to use COTS gaming technology like MFS, or X-Plane from Laminar Research, in a distributed interactive simulation environment. MFS can now participate as an entity in a synthetic environment in which a team training exercise is conducted. This creates new opportunities far beyond the current usage of PCATDs in the aviation community.

IEEE 1278.1A is the open IEEE standard for Distributive Interactive Simulation (DIS). DIS is a government/industry/academia initiative that defines a protocol for linking simulations of various types, at multiple locations, to create realistic, complex, virtual environments for the simulation of highly interactive activities. DIS exercises involve a mix of virtual entities with different levels of programmed behavior. The DIS standard defines Protocol Data Units (PDU) that are exchanged on a network between simulation applications allowing for connectivity and interoperability between training systems. For the purpose of our study, we focused on the DIS standard (and specifically, the Entity State, Fire, and Detonate PDUs.)

With the ability to inject entities, defined by Entity State PDUs within a virtual DIS exercise, into the

Microsoft Flight Simulator (MFS) environment, we can now "view" the exercise in our gaming domain and therefore interact with it. By the same token, we also have the ability to transmit our entity state and other information residing and controlled within the MFS game environment. Other trainers participating in this DIS exercise can now "see" the entity controlled by MFS in their world. Developing software to handle the Entity State PDU was our first priority.

2.1 Use of Micro-Simulations in a Distributed Exercise

In a Military environment the exercise or scenario can be generated by a deployed device such as the Multi Mission Team Trainer (MMTT). The MMTT system provides simulations for Combat Information Center CIC team training, Air Intercept Control training, and Ship's Warfare Coordinator Tactical training. It also provides tactical sensor and command and control simulations for use by ship and ship/air combat teams, as well as battle group staff supervisors. The MMTT system may also connect or network with the Battle Force Tactical Trainer (BFTT) in support of a team training objective. BFTT-equipped ships allow for stimulation of shipboard communication, navigation, sensor, propulsion damage control, and weapon systems into a common synthetic battle space tactical environment. Multiple platforms on a common network can participate in distributed exercises. A deployable and interoperable COTS Flight Simulator (based on MicroSim technology) will fit well into this team training environment. For example, any part of a team coordinated exercise that involves an air asset can be executed from a laptop. Even if the laptop offers a low fidelity representation of the air asset from the standpoint of tactics, controls and weapons, it allows a pilot to become familiar with the timing of his or her section of the overall mission and the situation awareness. In addition, the crewmember can practice interactions with other members of the mission given different circumstances. Crewmember responses and interactions to a cross section of possible situations within the mission can be practiced and evaluated.

3.0 Benefits

A low cost flight simulator such as MFS can be used as a powerful training platform. First of all, we are leveraging off an investment in a product that spans more than 20 years. Through third party development, there is a wealth of information and development tools available. Microsoft offers a full group of Software Development Kits (SDK) that open new doors for expansion. The SDKs provide insight into many areas of interest such instrument panel and scenery development, multiplayer capabilities, configuration file formats, recording and playback, and many others. The learning curve can be steep sometimes, and examples are limited throughout these SDKs, but they provide a good starting point for software development. The Multiplayer SDK was critical to finding a methodology for inserting DIS entities into the MFS world. It is based on Direct Play technology which is used by most COTS flight simulators. The data packets are still unique to each game, but with knowledge of the packets, plug-ins can be developed to allow games from different companies to operate in a multiplayer session. For example: a plug-in is already available that will allow MFS and X-Plane, from Laminar Research, to join a common DirectPlay session and interoperate.

We are not limited to Microsoft, and there are advantages and disadvantages to every product. DIS compatibility can be developed for any COTS simulator based on DirectPlay, which is the industry standard. In an initial evaluation of X-Plane in the early stages of our study, we found that although the flight dynamic models appeared better than MFS, there was no known methodology for designing gauges and panels. We did, however, design software that used the data supplied through X-Plane's UDP port to drive a MFS panel by feeding the data directly to the callback function of each gauge. Instead of driving the gauge with the Token variable driven by the MFS core, we used the values from X-Plane. The disadvantage is that two computers would be required. X-Plane just does not have the depth of world-wide third-party support that MFS has. The MFS list of third-party applications, aircraft, scenery, gauges, panels, data recording and playback, development tools, and many other categories is extensive.

Significant advantages of MicroSims are that the systems are portable, easy to maintain, and run on a low cost COTS PC platforms. They have been accepted by the aviation community as good part-task trainers. In a team training environment, with MicroSims participating in a DIS exercise, a pilot can rehearse a mission. The focus is not on high-fidelity operation of the aircraft, but on timing,

situational awareness, and interaction between team members in the context of the mission. Each team member will have a known responsibility in order for the mission to be considered successful. Furthermore, each team member will be expected to interact with other team members in certain ways, given predefined conditions. Each person has a window in which to execute his or her subset of the mission. Coordinating all of these tasks into a coherent mission time line is critical. The focus of using MicroSims to represent air assets in a DIS exercise is to rehearse this time line and allow team members to become familiar with each other's responsibilities in a controlled environment.

In an exercise involving our current MFS MicroSim, an instructor can create a MFS MicroSim flight plan that places each pilot at a given starting location (initial condition). All parametrics of the mission flown can be recorded for after action review. The pilot or instructor can play it back individually with standard VCR controls available through MFS, or all simulators can be controlled remotely where playback of each assigned section is based on the global mission clock for the DIS exercise. Playback can be done as many times as needed without the distractions of real flying. The pilot may want to focus strictly on visual landmarks or instrument navigation. If the pilot is interested in situational awareness, the area of interest may be played back as an observer that follows the aircraft. If the interest is in instrument navigation, the pilot may want to play back the flight overlaid on a map, which will show the position of the aircraft relative to airports, VHF Omni Direction Range (VORs), Non Directional Beacons (NDBs), airways, localizers, and other reference points. The map view can be used during playback to show the track over the ground. The map view further gives the mission coordinator the ability to drag the airplane to a new location and then set a specific heading, altitude, and airspeed to use when returning to the cockpit. The map view also serves as the airport/facility directory. To look up information about an airport, VOR, or NDB, a mission coordinator would switch to map view, zoom in, and select the item of interest. MFS includes the worldwide Jeppesen Navigation database of airports and navigation aids. For new students, and as a refresher, the map view is an excellent means for repositioning the aircraft relative to navaids in order to drive the instruments to see how the indications on the VOR, Instrument Landing System (ILS) and Automatic Direction Finder (ADF) have changed.

In addition, most COTS simulators allow complete control over weather conditions. The mission coordinator can create cloud layers, fog, crosswinds, or any other conditions applicable to the mission including day or night time settings. Third party products offer access to most Token variables used in setting weather conditions for MFS. The research and development team at NAVAIR Orlando has developed a Graphical User Interface that allows the mission coordinator to control the weather conditions for all copies of MFS participating in the mission from one central location. Another nice feature is the ability to program system and instrument failures into each team member's subset of the mission. The mission coordinator can later analyze the effect that the failures had on the overall mission and team coordination through the playback features mentioned previously. The failures are very realistic. For example, if the vacuum system dies, the gyro in the attitude indicator slowly winds down and the indicator gradually tips and dips, just as in an aircraft with a failed vacuum pump.

Although the focus is not on high-fidelity operation of the aircraft, the instrument panels are designed to be very accurate. Gauges have the correct form-fit and placement of the real gauge. When scanning the panel, the pilot can expect to find instruments in the correct location. An instrument panel can be built that spans across several displays. This does, however, drive up costs and reduces portability because more PCs or monitors are involved, but this is still a relatively low-cost solution compared to traditional high-end simulation systems. The same applies to the outside view. In our lab, we have used simple laptops, 50-inch plasma displays, and even a 2.5 meter dome driven by an Illumens lens to create a chin bubble effect for helicopters. The outside view can also span across several displays. Flight dynamic models are accurate enough to create the feel of a real aircraft. If flaps are extended, a pilot will notice a pitch-up force as the center of lift moves. The flight dynamic model does consider moments of inertia and center of gravity. Roll into a steep turn, and the pilot will need to pull back a bit on the stick to hold the nose up while lift decreases as a factor of the bank angle. A pilot still has to anticipate and compensate for such aerodynamic effects, even if the forces and control displacements don't precisely match those in a real airplane or helicopter.

4.0 Limitations and Shortfalls

With the experience we have acquired with this concept study, we have developed a list of issues associated with inbound insertion of DIS entities into MFS and the creation of the outbound DIS Entity State PDU. Microsoft has shown interest in our study and has requested the list in order to incorporate some improvements in their multiplayer capabilities that will help us with our future work.

The SDKs also offer good insight into several technical areas for interfacing with MFS. This is why they were described earlier in the paper as 'good starting points'. These documents are offered free to the public, however, there is no official support from Microsoft.

4.1 The 'Nature of COTS'

In 2002, Microsoft held their first annual "Navy Symposium." Vice Admiral Alfred G. Harms Jr. gave the opening keynote speech in which he specifically mentioned the Navy's interest in COTS Flight Simulators. This, along with other efforts from NAVAIR employees associated with the MicroSim projects in past years, as well as the results of our concept study, has helped in opening a line of communication with Bruce Williams, who is Microsoft's Product Planner for Flight Simulation Products. This communication has helped in understanding the data format presented within the MFS environment. Once a product is developed based on an SDK for a particular revision of MFS, there is no assurance that the product will continue to work with the next revision of MFS without some rework. Unfortunately, this is a common occurrence, since backward compatibility is not something that can be relied on in software development. One quick example would be the Navy Training Shell used with MFS. When Microsoft released MFS 2002, it changed the Adventure Programming Language (APL) used in MFS 2000 to the new ABL, or Adventure Basic Language. Microsoft's goal was the same: to provide developers of both adventures and lessons access to, and control of, aircraft and game parameters. The new ABL is a scripting language used by several Microsoft games. Since it is an interpreted language, it does not have to be compiled. There are several incompatibilities between APL and ABL, and the Training Shell was not compatible with the new ABL. By the time the Training Shell was updated, Microsoft was already close to the next release of the product! Another example of a change scheduled for version 2004 that will affect the training shell is new floating point Basic Graphic Language (BGL) opcodes. This new list of opcodes will be all that is available to render scenery/aircraft polygons and lines for MFS version 2004. Older opcodes that are still supported in MFS 2002 will not be available in MFS 2004. The main reason for this is that the new floating point BGL opcodes are more efficient and improve the frame rate. Scenery Enhancements and aircraft will have to be updated for MFS 2004.

4.2 Entity and Effects Limitations

In a multiplayer session (as used in a DIS exercise) a session is controlled by a single host through DirectPlay technology. The session is limited to 16 players, and each session is independent. The 16 players in one session have no interaction with the 16 players of another session. In order to mitigate this problem, we designed a filter to display the most important 16 entities. When players (a DIS entity is treated as a player) join and leave a session, a small disruption in the outside view is visible with MFS. Since this disruption causes the visual display to appear to "jump", this has to be kept to a minimum.

Special effects in MFS are controlled through emitter points that spew forth particles based on programmed behavior that can be controlled through a configuration file. Rules of physics can be applied creatively to create explosion effects, wakes, smoke trails, or any other effects. However, there are several problems that affect our work. First, a special effect must be pre-positioned at a specified latitude and longitude. A special effect can be activated conditionally, as the BGL opcodes (Macro Language) does offer "if" statements that can evaluate a Token variable associated with the aircraft being flown, in order to render the effect only when certain conditions are met. The macro language does not offer the ability to set a latitude and longitude in order to position the effect in real time. So in the case of the Hellfire missile (munitions), we cannot create a detonation effect at the desired point if the target is moving. So detonation effects have to be prepositioned on targets that do not move. The ability to fix this is one of the items on our wish list and in our future plans.

Once a Hellfire impacts a target, we have developed a methodology for switching to multiple

views to show cumulative damage to the target based on the statistical effect of the impact. A Detonation PDU is broadcast to relay this information to other devices on the DIS network. When a Hellfire is launched, a Fire PDU is broadcast to signal this event. The Hellfire missile has an associated Entity State PDU that describes its trajectory to the DIS exercise. This Entity State PDU joins the MFS session and occupies one of the available 16 MFS players, but can be seen on all copies of MFS participating in the DIS exercise. Other DIS devices will also pickup the Entity State PDU for the Hellfire if they are setup to handle munition entities. Unfortunately there are no smoke effects associated with the Hellfire or any player/DIS Entity that joins a session. The smoke effects and other visual enhancements such as rotor movement, skid marks, wake of a ship and others are tied to the aircraft, helicopter, ship, or land vehicle controlled in a particular copy of MFS. In all other copies of MFS that are in the session, the vehicle will appear as a body with no effects. A helicopter will have no moving blades, the ship will have no wake, and so on. So in the case of the Hellfire it made no sense to build an accurate representation because it would be so small that it could not be seen. To solve this problem, we created a large red tracer that is visible in the MFS world and follows the path of the Hellfire. Unfortunately, special effects have a detrimental effect on the frame rate, even on the best PCs. If MFS were also to display effects for players that have joined a session, it would reduce the frame rate even further.

Many third party vendors have used the aircraft smoke system of MFS 2002 to create the effect of firing some type of munition. The problem with this is that there is no real-time control over the effect. In other words, it is a canned repetitive effect. The Hellfire or any other munition launches straight out and falls to the ground creating a detonation exactly the same way every time relative to the position of the aircraft or land vehicle being controlled by MFS. It is very difficult to even estimate the latitude and longitude at which the impact takes place. We would like to have real-time control over the smoke system.

The MicroSim MFS interface also monitors all Detonation PDUs on the network and if any detonation event is intended for a particular copy of MFS, we can use the smoke system to create smoke and fire effects. This is the case where the Detonation PDU is inbound instead of outbound, as mentioned earlier. The only person that would see the effects is the person controlling the particular copy of MFS. His or her vehicle would show up as a body with no effects in all other copies of MFS on the session. But in this case, the smoke system can serve us well because the effects can be made conditional on damage reported by the inbound Detonation PDU to this one particular vehicle. Also, as a function of this PDU, we can damage aircraft systems to the point where the engine is dead. The ability to damage aircraft systems can also be controlled through third party add-ons.

Flight Simulator 2002 is not intended for managing and operating aircraft weapon systems. For example, in the case of our Hellfire implementation, we have a bright tracer that deploys from the general location of the aircraft. There is no correlation between the Hellfire missile (tracer) and the visual model of the aircraft or the dynamic model. The visual model may show Hellfire missiles on a Black Hawk helicopter, but they are not active, just part of the solid model. Aircraft in Microsoft Combat Flight Simulator (CFS) have an additional configuration file called the Damage Profile (dp) file. Although the format can be confusing, it does provide the capability of placing just about any kind of munition (guns, rockets, 500 lb or 2000 lb bombs, missiles, etc.) at specific points on the aircraft. The weight of the munition and placement, relative to the center of gravity, has the proper effect on the dynamic model. Also when a weapon is launched, visually there is one less weapon on the aircraft model.

The dp file does more than just weapon placement. The file also allows one to define boxes that define where the aircraft may be hit. The various functional parts of the aircraft are defined in this file as system entries. Each system will have a reference number and string. The string is used to generate damage messages. A system will also have a number called hit points associated with it. The number represents the strength of the system and is decremented every time the system is hit. There are also box maps that combine boxes with systems so the physical position of the system is defined. Furthermore, there is a defined list of systems that covers everything from the engines to the radio, control surfaces, fuel leaks, hydraulic failures, weapons, and many others. Then a large range of damage effects can be associated with each system. Damage effects become active as hit points are exhausted. Also, for each weapon attached to the aircraft, one can define many

parameters such as firing interval, muzzle velocity, pitch, bank, heading, weight, percent damage munition can cause, duration of time for computing ballistic trajectory, sound, and many others.

This all sounds good, but the first problem we encountered with CFS was that it was tailored to World War II, and the scenery was limited to the European and Pacific theaters. We resolved this problem by finding a method for importing MFS 2000 scenery into CFS2. We also found a method for importing our MFS 2002 aircraft into CFS2. This involves usage of a hex editor and a disassembler. We created a dp file for our helicopters. Methodologies for accessing internal Token variables work to a certain degree with CFS2, so we were able to generate an outbound DIS Entity State PDU rather easily. Third party options do not provide any information on weapons fired. The biggest CFS problem for us is that Microsoft has not released a Multiplayer SDK for the CFS series. This is another item on our wish list. Although it is based on DirectPlay technology, and the general sequence of packet movement is the same as MFS 2002, we do not have the multiplayer packet structures. It would be very time consuming to reverse engineer the packet structure. For this concept study, the ability to insert entities was extremely important, so we decided to stick with MFS 2002 as a starting point, given that a Multiplayer SDK was available.

Another important advantage CFS has over MFS and X-Plane is that it provides a utility that allows a user to build missions and establish success criteria. The mission builder allows placement of a whole range of enemy weapons and the ability to establish the effectiveness of each weapon. After importing MFS 2000 scenery into CFS2, we created a mission in the city of Chicago. We placed anti aircraft guns, missile batteries, and many other weapons all around the city. The mission was to destroy these weapons. Once the Multiplayer SDK for CFS becomes available, we expect to see some packets dedicated to weapons information. In multiplayer mode, all copies of CFS that are participating in a session do exchange munition impact information. We will be able to tie into this source of information to enhance a DIS exercise. The CFS multiplayer SDK is at the top of our wish list.

4.3 The Acceptance of 'Gaming' Technologies

Another disadvantage with gaming technology is acceptance by the DoD community. In the background section of this paper, the aviation community perspective was compared against the Navy perspective. We are looking to the future with our concept study. We are looking at team training possibilities in a distributed exercise, but the Navy has still not completely accepted PCATDs in their current form as a standalone training platform in the pilot training curriculum. The PCATDs are limited in numbers and still used on a voluntary basis. In our conversations with Microsoft and Laminar Research, one of the first questions that was asked was if the Navy has finally incorporated the usage of PCATDs into their curriculum. Unfortunately, the answer is still no.

4.4 Instructor Station Limitations

Other limitations include the fact that the instructor operator station for MFS and X-Plane can host only one student. It does not work in multiplayer mode. The map view, which was mentioned earlier, is a valuable tool, but it does not work in multiplayer mode either. It will not show other players in the session. To compensate for this, we have developed an aircraft-centered RADAR application that does work in multiplayer mode. Typically, there are many devices participating in a DIS exercise that are designed to show the "big picture", known as stealth viewers. MFS does not have to cover this role through the map view, but it would be nice to see all the players while analyzing a flight in the map view.

5.0 Generating Outbound Entity Information

When we initiated this study, third-party options for accessing Token variables were available for MFS 2002, but not validated. FSUIPC (Flight Simulator Universal Interface Program Control) is a utility that allows access to an offset list of Token variables internal to MFS. This list has been developed over many years through trial and error. This is a third-party product and is freeware. When a new revision of MFS is placed on the market, there are no guaranties that all of the offsets are still valid, plus there may be new offsets that are not included in the list. It takes time for the flight simulation community to re-validate the list.

It took about nine months for FSUIPC to become reliable for MFS 2002. Prior to this, we had to come up with our own method for obtaining the data minimally required to generate a DIS Entity State PDU. We first used an alternative method that took advantage of the fact that every moving part or readout in a gauge has a callback function that drives it. Through some reverse engineering, we obtained access to source code for the callback function. From within the callback function, there is a command called "lookup" which will allow access to any of the token variables listed in the Panels and Gauges SDK. From one callback function for any gauge, we could execute "lookup" for all the data values that we needed, such as position, velocity, and orientation. No computations were done in the callback function since it executes seventeen times per second and we did not want to affect the operation of the simulation. Instead, the data was placed in a socket and handled by a receiving task that would wait for data to arrive in the socket. We quickly got away from the socket and used shared memory along with a synchronization semaphore. The synchronization semaphore would be given from the callback function after data was placed in memory. This, in turn, would release the receiving task, which we called "entityOut". The receiving task would perform all the necessary coordinate conversions to conform to the DIS standard. The disadvantage of this method is that we had to modify the aircraft panel to physically place the gauge with the modified callback function. All of the available panel editors are third-party tools that have their little nuances. Once a panel is edited, the gauges may not appear exactly in the original location, and they may not be perfect circles. The other disadvantage is that the update rate may be too low for some applications when data is only being updated by the gauge callback at seventeen updates per second. The advantage is that the delta time between updates is known. This is important when using a Stealth Viewer as the outside view. If the Stealth Viewer is going to dead reckon position between returns to smooth the visual between updates, the delta time has to be known. Unfortunately, not all of the velocity terms are available with this method, including acceleration and angular velocities.

Finally, we embraced FSUIPC as our preferred method for the outbound Entity PDU with MFS. With FSUIPC, the update rate is very high, no gauge is involved in the transaction, and all velocity and acceleration terms are available. Our control of the stealth viewer is smoother, but still not as smooth as we would like. This could be due to two reasons. First, we were using a Sleep function to control our main loop but are currently changing code to use a high-resolution timer. Second, we do not know how often the Token variables in the FSUIPC offset list are updated. So we do not have an accurate delta time yet, and we are not synchronized with the FSUIPC internal update rate. This is a discussion that we will be having with the maintainer of FSUIPC. One of the advantages of X-Plane over MFS is that it offers access to a wealth of data through a UDP port and you do not have to rely on third party products.

6.0 Capturing Inbound Entity Information

Inserting inbound DIS Entity State PDUs into MFS or X-Plane's view is quite a bit more difficult than generating an outbound Entity State PDU that describes the vehicle controlled by the particular application. By inbound, we mean taking a DIS Entity State PDU and allowing it to join a multiplayer session as if it were a copy of MFS or X-Plane wanting to join a session. The session is controlled by a copy of MFS designated as host. This is handled through Microsoft's DirectPlay technology. The nice aspect about DirectPlay is that it is a networking Application Programmers Interface (API) that provides networking services at the transport and session protocol levels. DirectPlay is media independent, which means that DirectPlay sessions can be run on TCP/IP networks, IPX networks, and even over directly connected modems and serial cables. The application developer does not have to worry about supporting new standards and protocols as they become widely deployed, such as IPv6, multicast, and other protocols. This means that DirectPlay applications can improve over time, and keep up with evolving network standards, without additional development by the application developer. This is why most COTS simulators are based on it. Our work with MFS is applicable to other COTS flight simulators through this mechanism.

DirectPlay operates in different topologies, and MFS uses a Peer-to-Peer topology. Basically, all clients in a session are interconnected. One client is designated as the host and takes care of administrative overhead associated with maintaining the session. Host migration is supported, so that if the host leaves the session,

another client would be designated as host without any effect on the ongoing session.

In the discussion about generating outbound entities, it was mentioned that we ended up with a function called "entityOut" running in conjunction with every copy of MFS participating in the DIS exercise. For inbound entities, we have only one program called "entityIn". It can run on any of the computers participating in the DIS exercise. It does require the Internet Protocol (IP) address of the machine running MFS as host of the session. This program keeps track of all DIS traffic. Every Entity State PDU is accepted and compared against entities that have already joined the session. If the Entity has not joined the session and it passes our filter criteria, a player is created through DirectPlay. The architecture of the multiplayer system is based on information packets sent between the players of the session. Some packets contain extra information, and will have an embedded data structure. With the peer-to-peer methodology, all packets are sent to all players. Most of the Packet IDs and Packet Structures are defined in the Multiplayer SDK. These Packets are also available for X-Plane, which is why the SDK is so important. The data exchanged between players through DirectPlay is user defined.

When dealing with DirectPlay, one of the first steps is to instantiate a DirectPlay object. This will give access to the DirectPlay functions needed to control a multiplayer session. Once the decision has been made to allow a particular DIS Entity State PDU to join, a session has to be selected. There can be multiple sessions available on the network, and each session can host up to 16 players. Each session is independent of each other. The DirectPlay object provides functions for scanning for available sessions. In our case, we have just one session, but it still requires scanning to find it. If there are no copies of DirectPlay-compatible COTS simulators on the network configured as host, no sessions will be found. Once the session has been found, an Add-On is created for the particular Entity State PDU looking to join the session. When connecting the Add-On to the multiplayer session, a DirectPlay player will be created. The Add-On will communicate with the Host to validate the player created by the Add-On. The Add-On will also determine the characteristics of the other players in the multiplayer session, although we don't use this except for debug purposes. The reason is that the DirectPlay player created by the Add-On serves the multiplayer session by broadcasting a new POSITION_VELOCITY packet every time a DIS Entity State PDU that corresponds to it is received. Every DIS Entity State PDU that has joined the session will have a corresponding DirectPlay player.

It is beyond the scope of the paper to cover all steps associated with DirectPlay operation in the context of COTS simulators. Fortunately, DirectPlay is well documented, but there are also many objects that have to be initialized. For example, in order to minimize the time spent in a packet-sending loop, DirectPlay supports the concept of groups. A number of players can be assigned to a common group. Any packet destined for all the players in that group can be transmitted using a single DirectPlay "send" command. This is the method that we use, since all of our players belong to one group created during initialization. DIS Entity State PDUs are routed to the assigned player. The data is converted from the DIS coordinate system (WGS 84 Geocentric) to the Microsoft format. The POSITION_VELOCITY packet, along with other packets involved in the transaction, are broadcast to all players in the group with one "send" command. X-Plane uses a packet similar to MFS. In order to allow X-Plane to join the same multiplayer session, a special group would have to be created so packets that are intended for X-Plane are broadcast only to copies of X-Plane participating in the session.

7.0 Conclusion

This concept study has been conducted through NAVAIR Orlando's Concept Development and Integration Lab. Scenarios have been developed to demonstrate DIS interoperability with gaming technology. The demonstrations have been well received by the Navy. We believe that the obstacles that will have to be overcome in order to move forward are minor in comparison to the potential offered by this concept. The importance of Team Training cannot be overstated. COTS Flight Simulators have an important role to play as demonstrated in this concept study.

8.0 References

JIL Information Systems (Aug 30, 2002). Training Situation Document for Implementation
Of Micro simulators in Undergraduate Naval Flight Officer/Navigator Curriculum.

University of Central Florida Institute for Simulation and Training (Sept 15, 1999). Micro-Simulator Systems for Immersive Learning Environments project (MISSILE).

Microsoft (July, 2002). Multiplayer/Flight Instructor Software Development Kit.

Bruce Williams Flight Instructor (August 2000). Real-World Training Handbook.

Robert Di Benedetto (May 2001). Microsoft DirectPlay Overview.

Aircraft Owners and Pilots Association (AOPA) Flight Training, Bruce Landsberg (May, 2001). Safety Pilot: Meet the PCATD.

AOPA Flight Training, (Sept, 2001). Choosing Your Flight School and Regulatory Brief.

Aviation Stop Training (June 2000). Qualified PC Computer-Based Training Devices Take Off.

Aviation Research, Institute of Aviation, University of Illinois (Feb 2002). Incremental Training Effectiveness of Personal Computers Used for Instrument Training.

Aviation Research, Institute of Aviation, University of Illinois (July 2000). Evaluation of a PCATD to Meet Regency of Experience Requirements.

Federal Aviation Administration (May 1997). Advisory Circular 61-126.

Pete Dowson (December 2001). FSUIPC for Advanced Users.