A GRAPHICAL INTERFACE AND A DATA FILTERING SCHEME FOR JOINT THEATER LEVEL SIMULATION

Dr.LTC.Erdal Çayırcı*,Prof.Muhittin Gökmen+,Assoc.Prof.Sema Oktuğ+,Tacettin Ayar+, Gülşen Cebiroğlu-Eryiğit+, Tolga Çöplü+, Evren Okçu+, and Ltjg. İlker Akgün*
+War Games and Simulation Center, Istanbul Technical University, Maslak, Istanbul, Turkey
*Turkish War Colleges War Game and Simulation Center, Yeni Levent, Istanbul, Turkey

erdal@ harpak.tsk.mil.tr, gokmen@cs.itu.edu.tr, oktug@cs.itu.edu.tr, ayar@cs.itu.edu.tr, gulsen@cs.itu.edu.tr, coplu@itu.edu.tr, okcu@itu.edu.tr, iakgun@harpak.tsk.mil.tr

SUMMARY

Joint Theater Level Simulation (JTLS) [6-12] is the constructive simulation used to support joint/combined exercises at operational and higher levels in Turkish Armed Forces. However, JTLS does not fit all the requirements in these exercises. The tactical considerations may sometimes have a major impact on the decisions at higher levels. Especially, the resolution of terrain data does not allow simulating the required tactical details in JTLS. Moreover, participants of an exercise should interact with a simulation by using either real or realistic C2 systems. The standard user interfaces of JTLS, namely GIAC [1-5], MPP and IMT, cannot emulate these C2 systems in most of the cases. During the game, all the participants that use standard JTLS interfaces have the same perception of the operational picture if they are at the same side. This is not very realistic. JTLS cannot provide multiple perceptions for the same side. HOGAY is a generic graphical interface and filtering system developed to minimize these weaknesses of JTLS. In this paper we introduce the architecture of HOGAY, and the method to propagate data in HOGAY.

1.0 INTRODUCTION

Unless the constructive simulations are supported by appropriate user interfaces, they cannot fulfill the basic requirements, e.g., better immersion, being realistic, etc., of a computer aided exercise (CAX). If a realistic perception of the common operational picture cannot be provided to the users based on their side, level in the command hierarchy, communications tools and environment, the users cannot be satisfied even if the most complex and realistic combat models are used to calculate the attrition and mobility of units. Therefore, user interfaces are at least as important as combat models in military constructive simulations.

Joint theater level simulation (JTLS) has some shortcomings in this respect. In JTLS every terminal that belongs to the same side has the same view for the common operational picture. Every terminal of the same side learns about a unit as soon as any sensor of that side detects it. When the detection is fused, all the details related to the detected unit become available for the detecting side. This can make unrealistic impacts on the results of an exercise. A submarine commander can engage a surface ship more than 100 km away as soon as it is detected by a patrol boat, which does not have any link system. The graphical wargame interface application introduced, HOGAY, has been developed to overcome these difficulties caused by the user interfaces used in JTLS.

There are four factors affecting the design of HOGAY:

i) Requests of users: Players prefer to interact with JTLS using GUIs (Graphical User Interfaces) similar to real C2 systems. Requirements may change depending on the service that the player belongs to.

ii) Interaction with JTLS by using a single program: Currently, the player interact with JTLS by using four programs, namely GIAC (Graphical Interface Aggregate Control), MPP (Message Processor Program), IMT (Information Management Tool), and OPM (Online Player Manuel). The interface program developed will perform the functions of all these programs.

iii) Visualization of detection/fusion information with a propagation delay: Currently, detection/fusion information is obtained instantly by all the players of the detecting side. However, this information should propagate with respect to the delay parameters between unit pairs. Moreover, it should also be possible to form links among some units. Information obtained by a linked unit is immediately passed to the other units under the same link without considering physical distance among them.

iv) A list of utilities prepared by the users of the other interfaces is also taken into consideration.

The paper is organized as follows: In Section 2, the system architecture of HOGAY is explained. Section 3 introduces Model Client Interface, and basic processes implemented. Propagation of information and filtering is detailed in Section 4. Section 5 introduces Player and Controller Interfaces. Finally, Section 6 concludes the paper.

2.0 GENERAL ARCHITECTURE OF HOGAY

There are three basic units forming HOGAY, namely Player Interface (PI), Controller Interface (CI), and Model-Client Interface (MCI). There is a dedicated MCI for the players of each side. These are names as *MCI-PI-violet*, *MCI-PI-orange*, etc. Moreover, there is an MCI for the controllers called *MCI-CI*. Players use PIs to interact with JTLS. CI is a PI with some privileges. It can fetch data for all sides and send some special administrative orders to JTLS and MCI. MCI is an interface between JTLS and end-users (players and the controller). PIs and CIs work on the Windows environment. They are developed by using Visual C++ and Delphi. MCI is implemented on Linux by using the C programming language.

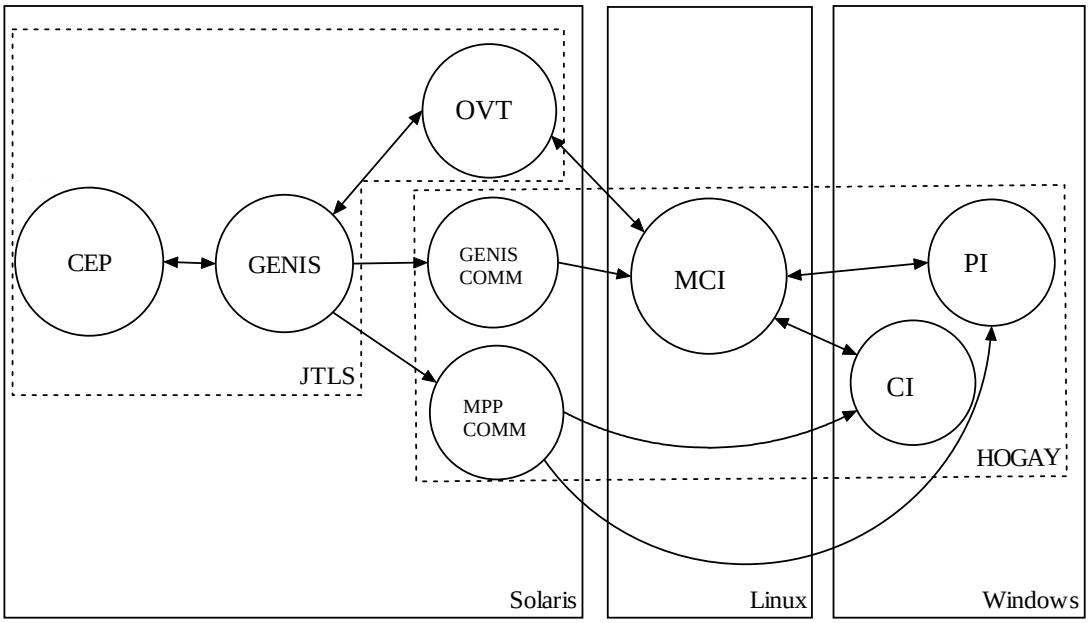

Figure 1: System Architecture

HOGAY has two more components also developed in UNIX environment by using the C programming language: GENIS-COM and MPP-COM. These components are included in the system design to improve the interoperability and reusability of HOGAY. MCI, PI and CI are independent from JTLS and other military simulation systems. These components use an application layer protocol designed for HOGAY to communicate. GENIS-COM and MPP-COM also use the same protocol to communicate with MCI and/or CIs and PIs. On the other hand, GENIS-COM and MPP-COM uses standard GENIS libraries to pass messages to or from GENIS. Therefore, adapting HOGAY to the version changes in JTLS is only limited to modifying GENIS-COM and MPP-COM which are also HLA compliant.

3.0 MODEL CLIENT INTERFACE (MCI)

MCI is a parallel processing program responsible from transmitting the GENIS data to PI-CI modules and the PI-CI orders to GENIS. There is a dedicated MCI for the players of each side. The MCI of a given side downloads the data specific to its side. *MCI-CI* downloads data for all sides and presents this data without any delay. *MCI-CI* also modifies the delay information related to the propagation of information among units based on the orders given by the controllers.

The MCI software consists of three fundamental processes and two supplementary processes for each new client connection, as shown in Figure 2. The fundamental processes are the *Main*, *Genis_Comm Reader* and *Updater* processes. *Main* does the elementary operations and initiates the required processes for each new client connection. *Genis_Comm Reader* is the process responsible of listening the GENIS socket and storing the incoming data. *Updater* updates the MCI data to be send to PI/CI modules. *Main* creates two new processes *PI_reader* and *PI_writer* for each new client connection. *PI_reader* receives the orders coming from the interface modules PI/CI and sends them to OVT. *PI_writer* sends the MCI data to the related interface module.

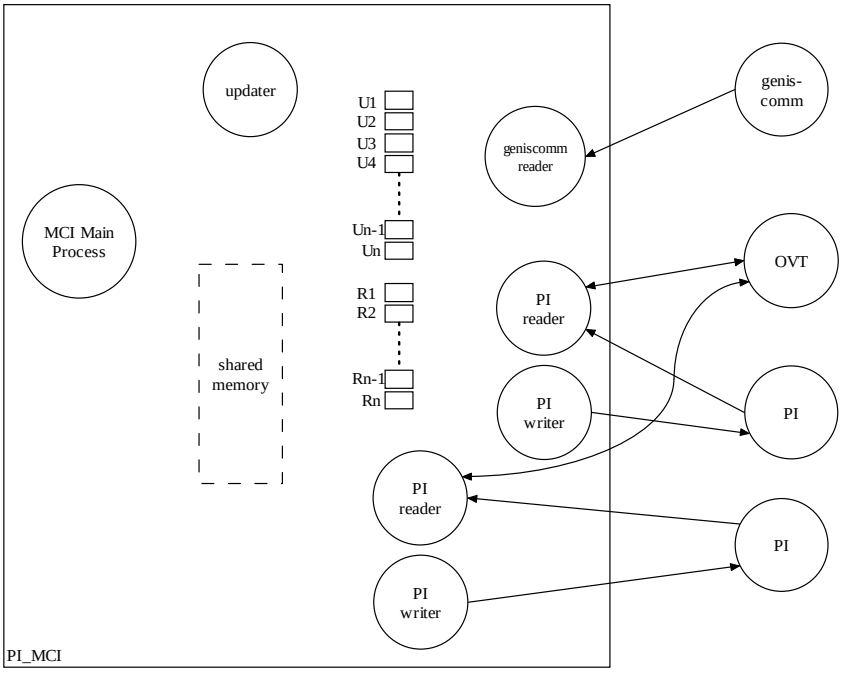

Figure 2: MCI Process Architecture

4.0 PROPAGATION OF INFORMATION

One of the most important contributions of HOGAY is the filtering module implemented in MCI and PIs. To pace with the changes in the game, its implementation is distributed between these two modules. Here a sample scenario, shown in Figure 3, is given to describe the filtering operation. In this sample scenario there are three players named O1, O2 and O3. Players are thought of as virtual units with zero delay values to the units under their authority. In this scenario, O1 and O2 have two units under their command and O3 has one unit under his command. Units are named B1 to B5.Delays between units are defined as GXY. (Here X and Y represent unit indexes.) For example delay between B1 and B3 is defined as G13.There are two relationships:

- Player-Unit Relationship
- Unit-Unit Relationship

A player can have more than a single unit under his/her control. For example player O1 controls units B1 and B2. Detections that reach to these units or the detections made by these units should be visible without any delay in the PI of O1. On the other hand, the other units can learn the detections made by units after a delay based on some parameters such as the position of units in the command hierarchy, the communications capabilities, jamming, links, and the distance between units. Delays between units are kept in a dynamic database. Delay values can be changed by the controller during the game.

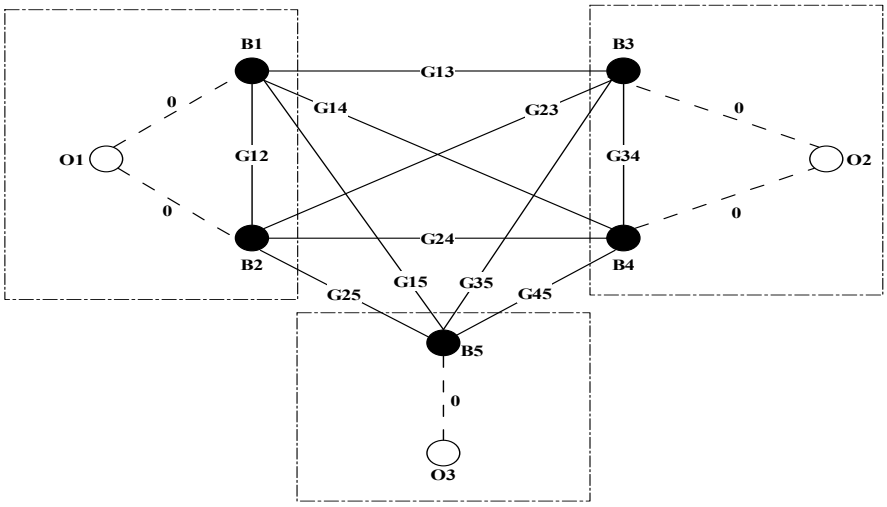

Figure 3: Sample Scenario

When MCI is run for the first time, or when the unit-to-unit or player-to-unit relations are changed, the shortest delay path between the units and players are calculated by using Floyd's all pair shortest path algorithm, and the results are kept in a matrix called *Floyd-Matrix*. Since Floyd's algorithm is a time consuming one, $O(N^3)$, it is assumed that the matrix is symmetrical, i.e. GXY is equal to GYX. When a PI connects to an MCI, the row related to the player of the PI from *Floyd-Matrix* is downloaded to the PI. This delay information, which gives the delay between any unit in the simulation and the player, is updated every time when the related row in the delay matrix in MCI is modified.

PIs do not reflect every update to the player as soon as they receive it from MCI. First they determine how long does it take to convey this update to the player based on the delay values. PI finds out the source unit of the update. If it is an update about a friend unit, that friendly unit is the source. However, when this is a

detection data, or an update about an enemy unit, PI needs to find the detecting unit. In HOGAY this is done by checking the location of the detected data against the ranges of open sensors available in the friend forces. The closest owner unit of one of these sensors that can make the detection is accepted as the source unit. After this point it is only a lookup in the delay relations data, which is determined by related row of *Floyd-matrix* to find out the required delay for the update in the operational picture of the unit. The updates are inserted into a linked list based on their update time determined by using these delays, and made available to the player as their turn in time come.

Filtering can also be bypassed, i.e. when an information update of a friend or enemy unit has arrived, it is shown on the player's screen directly without any delay.

5.0 PLAYER/CONTROLLER INTERFACE

PI is the interface unit that displays the propagated simulation data downloaded from MCI via TCP/IP. PI is the integrated form of GIAC, IMT, MPP, OPM. It includes all the options of these programs and more. Architecture of PI is composed of four DLLs and the main graphical interface program, as in Figure 4.

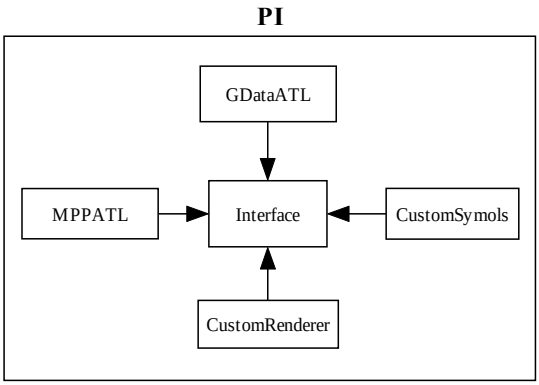

Figure 4: Architecture of Player Interface

GDataATL is a COM DLL implemented by using Visual C++. This module connects to MCI to download the simulation data and stores the updated version of simulation data. The filtered data is fired to graphical interface when triggered by time packets. Communication architecture of *GDataATL* is given in Figure 5.

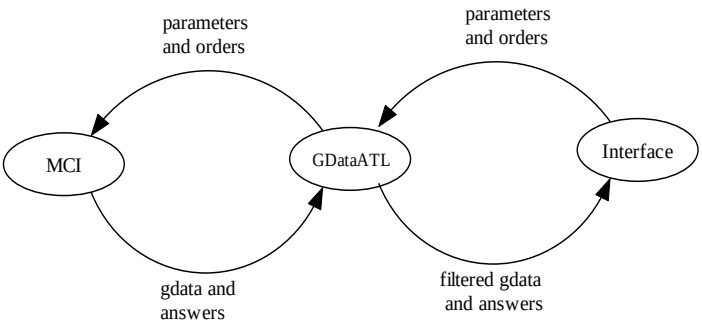

Figure 5: Communication Architecture of *GDataATL*

MPPATL is also a COM DLL implemented by using Visual C++. MPPATL connects to MPP-COM and fires the incoming messages to the graphical interface. *CustomSymbols* is a COM DLL designed for drawing the unit symbols in the graphical interface. This module is also implemented by using Visual C++. *CustomRenderer* is also a COM DLL implemented by using Visual Basic. *CustomRenderer* is used to draw user and unit lines, save or load them in layers.

Graphical Interface is the basic interface developed in Delphi. In Figure 6, PI and GIAC screens are shown. At the first glance, both of them are quite similar. The main difference is that PI uses raster maps while GIAC uses hexagonal maps. Apart from this, all the information panes, buttons, and windows are collocated in a dockable window in PI. This window can be dragged into any place, resized or minimized as needed. The order menus and the other menus are also positioned as in a standard windows application. Hence, it is easier to train the operators who are used to working in MS Windows environment.

PI has all of the utilities available in GIAC e.g., unit and user lines, display options, filtering options, panning, zooming, etc. Apart from them, it has some additional tools designed based on the lesson learned from the previous exercises. For example, three C2 systems, namely ICC, STACOS and a new C2 system, are implemented in PI. User selects one of them, and the screen looks like the screen of the selected C2 system. This does not change the tool bar or the menu but only the operational picture is shown in a screen that looks like the screen of the selected C2 system. Another important new tool is Local Tracks which are virtual units created by players to demonstrate probable units at that location. Local tracks can be created, deleted, moved or shared with other players. This new utility will be very useful in the detection/fusion steps or in electronic war models.

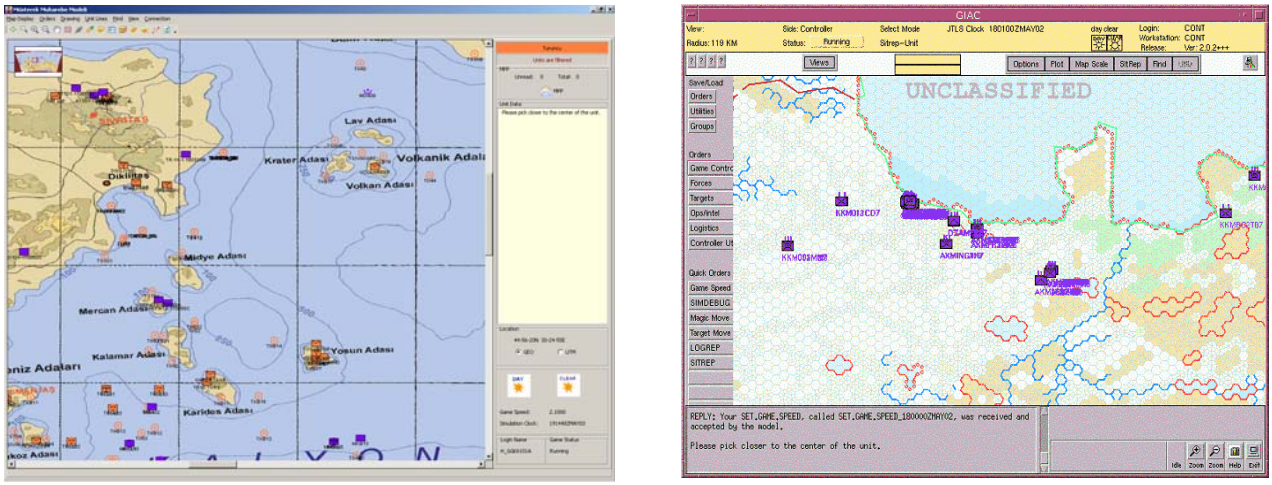

Figure 6: PI and GIAC Screens

The navigation on the map is also easier in PI where the location of the current screen in overall theatre is also shown in a navigation window. It is also possible to find out the length of a river or road as shown in the dockable information window in Figure 7. Another important tool available in PI is a very realistic three-dimensional (3D) flight simulator. Some screens from this real time flight simulator are shown in Figure 8. These screens are generated by using standard DTED formatted digital maps, satellite pictures and some additional data related to terrain features and textures.

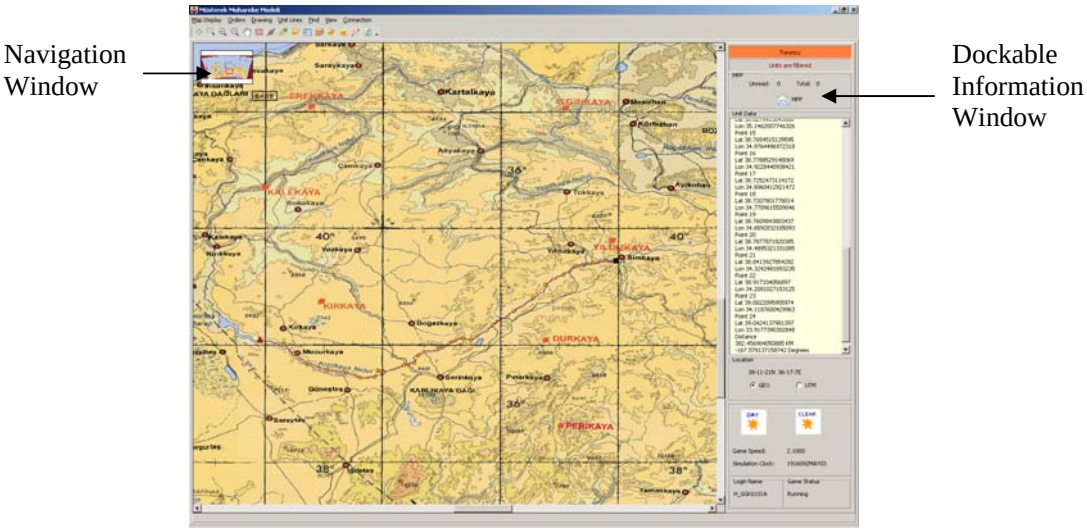

Figure 7: PI Utilities

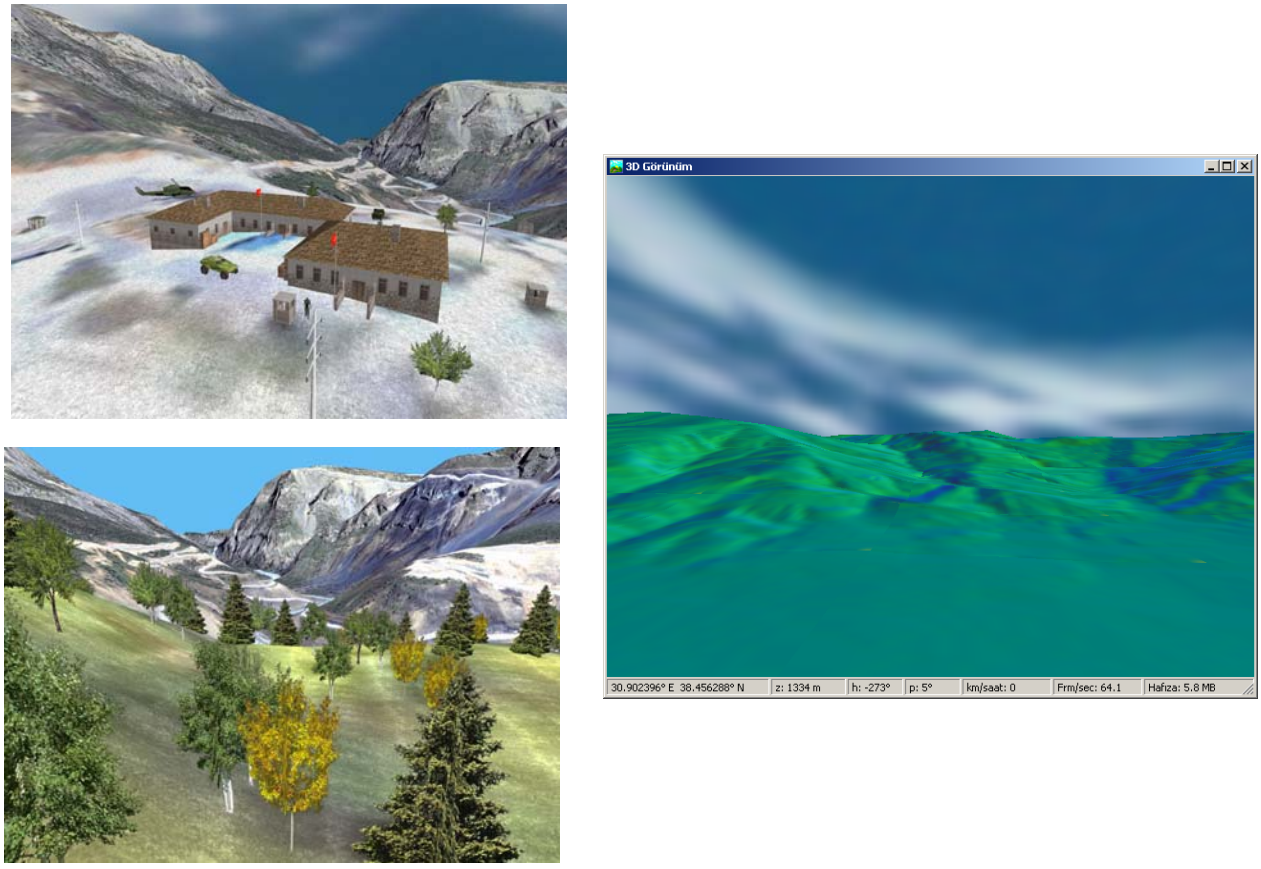

Figure 8: The Screens from the 3D Flight Simulator of PI

PI also presents a window where the information about the units and other assets in the theatre in tabular

form. This tool, which is an enhanced version of IMT used in standard JTLS configuration. Another tool is the equivalent of MPP in JTLS where the users can see the messages generated by JTLS for them. Users send their orders to JTLS also by using templates similar to the ones in JTLS. With all of these utilities HOGAY will become the standard user interface for the constructive simulation systems used in Turkish Armed Forces.

CI and PI are the same programs. During the authentication stage order menus are dynamically created according to the user's type. So the only visual difference is the orders they can give. CI is connected to *MCI-CI* to get the whole simulation data without filtering in *GDataATL* if the user is logged as a controller.

6.0 CONCLUSIONS

HOGAY is an interface system which can connect to JTLS, and provide users with filtered perceptions based on command hierarchy and propagation delay. Available C2 systems in the Turkish Armed Forces are also emulated in HOGAY. Hence, exercise participants are better immersed into situation, and more realistic joint exercises can be conducted. HOGAY has also many new utilities which make the players learn how to use it easier and operate it more effectively. It was first used in a major exercise in 2003.

8.0 REFERENCES

[1] *AAIT GIAC 1.9 Version Description Document-DRAFT*, Los Alamos National Lab., Release 1.9, February 1999.

[2] *GIAC G Data System*, Los Alamos National Lab., Release 1.9, February 1999.

[3] *GIAC Model Controller's Guide*, Los Alamos National Lab., Release 1.9, March 1999.

[4] *GIAC Overview*, Los Alamos National Lab., Release. 1.7, May 1998.

[5] *GIAC Player's Guide*, Los Alamos National Lab., Release 1.9, February 1999.

[6] *JTLS Analyst's Guide*, Rolands and Associates Corporation, Ver.2.3, February 2001.

[7] *JTLS Controller's Guide*, Rolands and Associates Corporation, Ver.2.3A, April 2001.

[8] *JTLS Data Requirements Manual*, Rolands and Associates Corporation, Ver.2.3, February 2001.

[9] *JTLS Installation Manual*, Rolands and Associates Corporation, Ver.2.3, February 2001.

[10] *JTLS Lanchester Development Tool Guide*, Rolands and Associates Corporation, Ver.2.3, February 2001.

[11] *JTLS Standard Database Description*, Rolands and Associates Corporation , Ver. 2.1.3., October 1999.

[12] *JTLS Player's Guide*, Rolands and Associates Corporation, Ver.2.3A, April 2001.