Computer Generated Forces (CGF) User Interface (UI):
A Common Front End for CGF Applications

Derek Bryan  
Brian Teer  
BMH Associates, Inc.  
5365 Robin Hood Road, Suite 100  
Norfolk, VA 23513-2416  
(757) 857-5670  
dbryan@bmh.com, teer@bmh.com

Keywords: User Interface, C-IOS, JNTC, NASMP, VAST

ABSTRACT: The use of Computer Generated Forces (CGF) simulations to support military applications is rapidly increasing. This is primarily due to the low costs associated with Modeling and Simulation (M&S) development and the wide range of domains that M&S applications support. As a result, there are a large number and type of CGF applications throughout the Department of Defense (DoD). Currently, all of these CGF applications maintain distinct and independent user interfaces that require individual training in order for CGF operators to achieve a level of proficiency. The CGF User Interface (UI) will provide standardization for both user-level and software-level interfaces to several CGF simulation applications and, thus, reduce the effort and costs associated with employing multiple, concurrent CGF applications. This paper explains why a common CGF UI is needed and who will benefit from it and provides a brief introduction to the CGF UI architecture.

1. Introduction

Computer Generated Forces (CGF) simulations have been in development since the late 1980s. Since that time, the Department of Defense (DoD) has found numerous ways of using CGF and related technologies to reduce the effort and costs of its programs. Examples of the way CGF applications are used today include training systems support, experimentation, analysis, emulation, and stimulation.

Currently, no single CGF application can satisfy all of the requirements listed above for all domains. For that reason, the DoD has developed numerous CGF applications, each with its own strengths, weaknesses, and intended use. Consequently, each program must investigate and determine whether it is more appropriate to develop the desired capabilities within a single CGF application or integrate multiple CGF applications. Each approach has advantages and disadvantages, which must be evaluated on a case by case basis bounded by the intended use. The (common) CGF User Interface (UI) is being developed to reduce the impact of this realization on DoD programs.

The remainder of this paper shall serve as a mechanism for describing the CGF UI program, targeted users and operations, and anticipated program-level benefits.

2. Program Overview

Initial CGF UI activities were sponsored by the Office of Naval Research (ONR) as part of the Anti-Submarine Warfare (ASW) Air Virtual At-Sea Training (VAST) program. The first phase of ASW Air VAST, completed in November 2003, developed a prototype of a deployable tactical team trainer for the SH-60B aircrew. The trainer consists of four stations: an Air Tactical Officer (ATO), a Sensor Operator (SENSO), a pilot station, and an Interim Instructor/Operator Station (IIOS). The IIOS consists of a Plan View Display (PVD)\(^1\) for controlling CGF and monitoring the synthetic battlespace, a process manager, an application manager, a logger/playback application, and an instructor/trainee digital voice application.

The IIOS, and specifically the PVD, were used to generate a comprehensive set of system-level and functional requirements for the CGF UI. The two volume set of requirements was submitted and approved by ONR and PMA 205 as a baseline set of requirements for the CGF UI. PMA 205 expressed interest in the CGF UI due to its potential use in the Naval Aviation Simulation Master Plan (NASMP) program.

---

\(^1\) A PVD is a two-dimensional map display of the synthetic environment playbox.
PMA 205 and NAVAIR Orlando will be developing a Common IOS (C-IOS) for distributed simulator platform environments that includes the ability to configure, monitor, and control CGF [1]. The NASMP program conducted an extensive CGF evaluation and concluded that two CGF applications, Next Generation Threat System (NGTS) and Joint Semi-Automated Forces (JSAF), were required to support near-term NASMP objectives. The BMH CGF UI team is collaborating with both PMA 205 and NAVAIR Orlando throughout the CGF UI development life cycle to ensure that, if desired, the CGF UI can be part of both the C-IOS and the NASMP tool set.

In addition to NASMP, the Joint National Training Capability (JNTC) program is sponsoring Fiscal Year (FY) 2004 CGF UI development. It is anticipated that JNTC will employ multiple CGF simulations, including Joint Conflict and Tactical Simulation (JCATS), JQUAD², and Joint Theater Level Simulation (JTLS) to name a few. Specific CGF UI requirements for JNTC are currently being developed and will be added to the existing requirements volumes early in 2004.

The CGF UI FY04 schedule consists of four major milestones: a prototyping phase and three development iterations. The prototype development was completed in mid-January and was demonstrated to and approved by PMA 205 and NAVAIR Orlando. The purpose of the prototyping phase was to obtain feedback regarding the user interface component of the CGF UI and to mitigate any technical risk associated with the tactical map implementation. The first development iteration will result in an architecture baseline that consists of initial implementations of all software components and interfaces. The second development iteration will produce the feature baseline, which will include a set of high priority features (requirements) that are applicable to any CGF application, regardless of capability or intended use. The final and shortest iteration will implement features that were not developed in the second iteration, but were asked for by Subject Matter Experts (SMEs) after evaluating the feature baseline capabilities. The first external release of the CGF UI is anticipated in June/July of FY04.

The CGF UI development team is using level 3 Capability Maturity Model – Integrated (CMMI) practices and procedures. Examples of artifacts that the CGF UI team is developing include: system-level requirements, functional requirements, an architecture description document, a software requirements specification for each software module, a software design description for each software module, and a system test plan. The CGF UI team currently consists of a senior project manager, a project engineer, a technical lead, three SMEs, seven software engineers, and two test and quality assurance engineers.

3. Why is the CGF UI Important?

The CGF UI is important for many reasons. First, the development of a common UI for multiple CGF applications will allow programs to share operator resources more effectively. Shared operator resources will enhance cross-program productivity by increasing operators’ proficiency with a common tool.

Figure 1 depicts an overview of the way CGFs are employed by each of the military service communities. Currently, service and application-specific operators are required to support each of the CGFs. The CGF UI is intended to reduce, and in some cases eliminate, the

2 JQUAD is a suite of four simulation models used as part of the Joint Training Confederation (JTC) to simulate many components of command and control warfare. The four models are: Joint Electronic Combat Electronic Warfare Simulation (JECEWSI), Joint Command and Control (C2) Attack Simulation (JCAS), Joint Networks Simulation (JNETS), and Joint Operations Information Simulation (JOISIM).
Need for these service and application-specific CGF operators. The reduction in operator requirements is illustrated by Figure 2.

The requirements for the CGF UI were developed by existing CGF operators. These requirements ensure that the CGF UI will contain the features and controls that are necessary to efficiently interact with complex CGF environments. These CGF operators will also ensure that the CGF UI uses domain-oriented terminology in order to reduce training time associated with the use of the CGF UI. CGF operators will also be evaluating each of the CGF UI releases to ensure that the implementation not only meets the requirements, but that the UI is consistent and usable.

The CGF UI is not only standardizing front end interfaces and operations for CGFs, but it is also standardizing distributed protocols and software-level interfaces for the CGF back ends. The CGF UI team is collaborating with the Naval Training Meta Federation Object Model (FOM) (NTMF) and the Joint FOM (JFOM) communities to develop a CGF FOM. The CGF FOM will be composed of a number of Base Object Models (BOMs). Standard BOMs, such as the Simulation Management (SIMAN) BOM included in NTMF 1.0, will be used to the maximum extent possible. In addition, the CGF UI development team will be implementing a common communications interface. This interface will be included in each of the CGF back ends and will standardize communications with the CGF UI. By doing so, the majority of the implementation burden remains on the front end (CGF UI) and the changes to the CGF back ends are minimized.

Finally, the CGF UI architecture is being designed as an open, modular architecture that will maximize interoperability with other programs and applications. It is anticipated that components of the architecture will prove to be reusable in other contexts.

4. Who Will Benefit from the CGF UI?

The CGF UI is being designed to support start to finish operations with CGF applications independent of their intended use. For example, the CGF UI will be capable of supporting experimentation environments such as those developed as part of Fleet Battle Experiment (FBE) and training environments such as the Joint National Training Capability (JNTC) and NASMP. The CGF UI will initially support entity-level interactions but will be extended to support aggregate operations in future releases.

The primary functions of the CGF UI are the following: CGF systems administration, scenario planning and execution, and After Action Review (AAR). Each of these functions is intended to support a particular user. CGF systems administration is intended to support operations typically provided by a federation manager or engineer. Federation managers and engineers are responsible for understanding and maintaining the health and status of the overall simulation infrastructure including network and application state and performance. CGF operators will use the scenario planning and execution functions to define, instantiate, and monitor CGF scenarios. The AAR functions can be used by CGF operators, analysts, or instructors to replay, critique, highlight, and record interesting events that occurred during simulation execution.

5. What the CGF UI Is

5.1 Concept of Operations

The CGF UI will support two primary modes of operation: offline and online. Multiple CGF UI operators will be able to participate in either mode of operation concurrently. The primary and secondary modes, along with their interactions with different simulation network components, are shown in Figure 3.
In the offline mode, a single CGF UI will be able to run independently or multiple CGF UI(s) will be able to run simultaneously and communicate with one another. In either case, there will be no communication with CGF simulation applications. The offline mode will support two secondary modes of operation: administration and planning.

When the CGF UI is brought online, communications are established with the central data store and the CGF applications on the simulation network. The online mode will support four secondary modes of operation: administration, planning, execution, and AAR.

5.1.1 Administration

The CGF UI will support a set of administrative features that will be available to a CGF UI administrator. These include the following capabilities:

- Configure the CGF UI system. This includes any remote configuration needed for CGF simulation applications; the centralized data store; any required gateways; an AAR logger/playback system, if applicable; etc.
- Save and restore a CGF UI system configuration.
- File distribution, which allows large amounts of data to be distributed quickly throughout the simulation network. This is particularly useful in environments where configuration data may change frequently.
- Customize the type, amount, and content of the information that is presented to the CGF UI operators.
- Change operator permissions, allowing an administrator to restrict CGF UI features based upon permission settings.
- Remotely control the CGF simulation application machines. This includes the ability to launch, quit, and monitor the status of CGF applications, execute scripts, etc. To facilitate this, the CGF UI operator will be able to set any needed environmental variables and command line arguments. This is a system-level interaction.
- Start, stop, pause, and resume CGF applications. This is an application-level interaction.
- Assess the current state of a CGF application’s hardware resources (e.g., CPU utilization, available memory, list of running processes, etc.).

5.1.2 Planning

Planning CGF scenarios does not require any interaction with the CGF applications. In fact, it may take place in offline mode, using a CGF UI that is not connected to the simulation network. Ultimately, planned scenarios are saved to the data store, making them accessible to all machines in the simulation network.

Planning CGF scenarios involves specifying all scenario-level data (e.g., geographic location, time frame, etc.), entities, and mission data. In some cases, some of this data can be imported to facilitate the process. Furthermore, it is possible for multiple CGF UI operators to plan scenarios collaboratively; each operator can plan a subset of the scenario, and the subsets can then be merged into a single scenario.

The CGF UI will support the following planning capabilities:

- Import planning data.
- Control the map display.
- Create, modify, delete, save, load, and unload overlays.
- Create, modify, and delete overlay objects (i.e., points, routes, and areas).
- Create, modify, delete, save, load, and merge scenarios.
- Create, modify, and delete entities (e.g., aircraft, ships, tanks, etc.).
- Create, modify, and delete missions assigned to specific entities.
- Create, modify, and delete obstacles.

5.1.3 Execution

All planning capabilities will be available to CGF UI operators in execution mode. However, the CGF UI operator’s actions can be activated to take immediate
effect in execution mode, as opposed to being stored in a scenario for future execution as they will be in planning mode.

The CGF UI will also support the following execution capabilities:

- Load a previously planned scenario for execution. This causes the CGF UI to send appropriate messages to the CGF applications to create and task the planned entities.
- Monitor the battlespace.
- View entity status and statistics.

5.1.4 AAR

The CGF UI will support the following planning capabilities:

- Control the map display.
- Monitor the battlespace.
- Use SIMAN to start, stop, pause, and resume the AAR playback. Note that AAR mode requires the use of a logger/playback system in conjunction with the CGF UI.

5.2 Architecture

5.2.1 Objectives

The primary objectives of the CGF UI architecture are the following:

- Multiple, concurrent CGF UIs should be interoperable with multiple concurrent CGF UI-compliant CGF applications (potentially of different types).
- Run-time execution should support Local Area Network (LAN) or Wide Area Network (WAN) environments.
- The CGF UI should be based upon standards for distributed communication, and it should support multiple simulation/application protocols.
- Planning should be supported both on and off the simulation network.
- The CGF UI should be compatible with both Windows and Linux operating systems.

5.2.2 Constraints

The CGF UI uses the High Level Architecture (HLA) protocol for its simulation communications. This imposes the following constraints on compliant CGF applications:

- Each CGF application must be a federate in the CGF UI HLA federation. A CGF UI communications interface module will be provided to reduce the burden of this implementation for non-HLA CGF simulation applications.
- Non-HLA simulation messages must be translated to HLA via an appropriate gateway.

The CGF UI architecture itself is subject to the following constraint:

- The existing simulation communications employed by the CGF applications must be unaffected by the CGF UI architecture.

5.2.3 Description

Figure 4 shows the notional architecture for the CGF UI system. The main system components are the CGF UI; CGF simulation application; the simulation network; the centralized data store; and the protocol gateways, such as the Distributed Interactive Simulation (DIS)/HLA Gateway.

Notable aspects of the notional architecture include the following:

- This architecture is a component and message based architecture. Consequently, the reliability of the architecture is based upon the reliability of the individual architectural components and the simulation network used for message passing.
- The component and message approach also maximizes flexibility and scalability. The number and configuration of architectural components is limited only by the simulation network capacity.
- A common communications interface module will be used in both the CGF UI and CGF applications to support CGF UI simulation (i.e., planning, execution, and AAR) communications.
- For system administration purposes, the CGF UI will communicate directly with each machine in the simulation network (i.e., the machines upon which the CGF UIs and CGF application are running) via Remote Method Invocation (RMI). This requires a daemon to run on each machine. Secure Socket Layer (SSL) is used to ensure these communications are secure.
- HLA is the network protocol used for all other CGF UI communications. The CGF FOM
The actual HLA simulation data, including both data sent between CGF applications and data sent from the CGF UI to the CGF applications, is sent via the Runtime Infrastructure (RTI). RTI communications are not secure by default. In order to secure RTI communications, additional hardware must be used at each site on the simulation network to encrypt and decrypt the RTI messages.

- The CGF UIs will share a centralized data store, which contains information required to save and restore the state of all CGF UIs as well as user preferences, planning outputs, etc. Because the data store is centralized, the data is accessible to any machine in the simulation network.

- All modules will be designed to encapsulate their details behind appropriate Application Programming Interfaces (APIs). This will enable the details of a module to be changed without affecting the other modules. In fact, it is conceivable an entire module could be replaced (e.g., another GUI could be substituted for the CGF UI GUI), provided the replacement module implements the same API.

The CGF UI will be data driven to the maximum extent possible, thus allowing the data to change without requiring code changes.

- In keeping with the previous point, data will be handled generically throughout the system to minimize the ripple effect of changes. This extends to the module APIs and the FOM specification. For example, it will be possible to add or change mission definitions without changing any interfaces or FOM definitions.

### 5.2.4 GUI Module

The GUI module is the only module that directly interacts with the operator. It will be designed to support standard GUI features of “typical” Windows
applications, and it will present a consistent look and feel. Domain-specific terminology will be used to the maximum extent possible, with the goal being to allow an operator familiar with the domain but unfamiliar with the simulation specifics to use the CGF UI with a minimum amount of training.

5.2.5 Planning & Execution Module

The planning & execution module will support the planning and execution capabilities described in section Error! Reference source not found. The primary planning focus is to allow a CGF UI operator to specify and develop CGF scenarios. In this context, a scenario is an exercise or operation that spans a period of time (e.g., 24 hours), operates in a specific geographic region, and consists of entities with assigned missions. The primary execution focus is to allow an operator to monitor, interact with, and control a running scenario.

The CGF UI will provide planning data choices (e.g., entity types, mission types, etc.) based upon a single domain-centric definition that is intended to apply across all relevant platforms. A process will be developed to control the evolution of this data definition, allow proposal and approval of changes, publish new versions, etc.

Scenarios will be saved in a simulation independent format. They will not be passed to the CGF applications for execution. Rather, when an operator loads a scenario for execution, the CGF UI will send the appropriate interactions to cause the CGF applications to create and task the entities as planned.

5.2.6 Tactical Map Module

The tactical map module will allow a CGF UI operator to monitor a two-dimensional view of the battlespace and interact with the synthetic environment. It will provide capabilities for creating, modifying, and deleting points, routes, areas, and overlays, as well as basic map functions, such as panning and zooming. This module will also encapsulate functionality associated with multiple coordinate systems, including at a minimum latitude & longitude, Military Grid Reference System (MGRS), and Universal Transverse Mercator (UTM).

The current design calls for use of OpenMap to provide a base level of map functionality that is customized for CGF UI purposes. Elevation data will be represented in Digital Terrain Elevation Data (DTED) and feature data in Vector Product Format (VPF). Use of VPF will allow CGF UI to support custom data layers for specific program needs (e.g., an airfield layer). It will also support conversion from other formats such as the Compact Terrain Database (CTDB) format.

5.2.7 System Administration Module

The system administration module will provide capabilities for customizing the CGF UI system configuration. This includes customizing the type, amount, and contents of the information that is presented to the CGF UI operators. Some of these capabilities, such as setting operator permissions, are administrative and thus will be performed only by CGF UI administrators. Others, such as specifying which type (domain, force, etc.) of CGF should be simulated by a given CGF simulation application, are operator-level tasks. Note that it is possible for one user to be both a CGF UI administrator and CGF UI operator.

In addition, the system administration module will allow a CGF UI operator to monitor the status of the network and the CGF simulation applications and their machines. This information will assist an operator in planning a CGF scenario and configuring his or her CGF simulation load. This module will also allow an operator to control the state of the CGF simulation applications via SIMAN commands.

5.2.8 Communications Interface Module

The communications interface module will serve to encapsulate all aspects of system and simulation level communications away from the rest of the CGF UI and CGF simulation applications. This includes all details of the FOM (see section 3). Furthermore, this module will also encapsulate the details of CGF UI system federation membership, including joining, subscribing, resigning, etc.

This module is designed such that a CGF application subscribes to specific areas of interest and implements callbacks to support the functions of those areas. The module will filter messages that do not apply to the registered areas, meaning the CGF application will not need to handle them or even be aware of them.

5.2.9 Data Store Interface Module

The data store interface module will encapsulate access to the data store. Ultimately, the data store implementation may change over time, but the current plan is for a centralized database, combined with a collection of files stored locally on each CGF UI machine. This interface module will minimize the effects of changes to the data store implementation as they occur over time.
All data created and saved in online mode will be stored in the centralized data store, thus providing access to the data from any machine on the simulation network. When a user creates and saves data in offline mode, it will be saved to the local machine; when the user subsequently connects the machine to the simulation network and enters online mode, the data will be synchronized with the centralized data store.

Consequently, an operator will be able to save preferences, scenarios, etc. from one CGF UI and access them from another CGF UI. This will facilitate collaboration on planning efforts.

### 5.3 Technologies

The CGF UI development team is using mature and proven technologies to ensure that the implementation of the CGF UI is both sustainable and maintainable. The CGF UI front end application is being implemented in Java. The communications interface module makes use of an Agile FOM Interface (AFI), derived from JSAF, which is written in C++ and is accessed via the Java Native Interface (JNI).

The back end communications interface will be implemented as a C++ library that can be linked into C or C++ CGF applications. A JNI wrapper will be available for this library to provide support for Java-based CGF applications.

The GUI will use the Java Swing components. Portions of the GUI will be data driven, with certain forms automatically generated from data specifications. This will enable the GUI to be extended in certain areas simply by adding or changing data. Where data files are needed, they will be formatted as Java properties files or in Extensible Markup Language (XML).

### 6. What the CGF UI Is Not

The CGF UI is not a complete replacement of all existing CGF user interfaces. It is unreasonable to expect that the CGF UI will be able to configure and control CGFs at the level that is currently provided by each CGF’s native UI. Initial releases of the CGF UI will provide features and capabilities for the majority of CGF operations and interactions. Future releases of the CGF UI will provide advanced configuration and control operations that are applicable to multiple CGF back ends. It is anticipated that a reduced number of native CGF user interfaces will be required while the CGF UI is further developed.

The CGF UI will not solve system-level interoperability issues. Existing interoperability issues such as terrain correlation, model fidelity, and coordinate system translations will still exist with the use of the CGF UI. The CGF UI will address operator-level interoperability concerns such as human-computer interaction and familiarization.

### 7. Summary

This paper presents the purpose, plan, and intent of the CGF UI. The CGF UI development team will continue to collaborate with organizations throughout the CGF community to ensure that the CGF UI will be a useful and maintainable product that will reduce overall program costs and enhance operator effectiveness.

### 8. References


### Author Biographies

**DEREK BRYAN** is the Technical Director for the Norfolk office of BMH Associates, Inc. In that role, he performs both senior project management and software engineering roles on multiple projects, including the CGF UI project. Derek holds a B.S. in Computer Science from James Madison University and is pursuing a M.S. in Modeling & Simulation from Old Dominion University.

**BRIAN TEER** is a senior software engineer at BMH Associates, Inc. He is the project engineer for the CGF UI project. Brian holds a B.S. in Electrical Engineering and a M.S. in Computer Science, both from North Carolina State University.