NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

STRUCTURED VERSUS OBJECT-ORIENTED DESIGN OF A NAVY BATTLE GROUP LOGISTICS SIMULATION SYSTEM

by

Bernadette Clemente Brooks

March 1993

Thesis Advisor: C. Thomas Wu

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Computer Science Dept. Naval Postgraduate School; 6b. OFFICE SYMBOL (if applicable) CS	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
Structured Versus Object-Oriented Design of a Navy Battle Group Logistics Simulation System

12. PERSONAL AUTHOR(S) Bernadette Clemente Brooks

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM 01/92 TO: 03/93	1993, March 25	251

16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	Structured Programming, Object-Oriented Programming, Logistics Support Systems

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*
This thesis deals with the design of a Navy battle group logistics simulation system to support battle group logistics coordinators. BGLCSS 2.0, the Battle Group Logistics Coordinator Support System, was designed and developed using a structured programming paradigm. A subset of BGLCSS 2.0 was then designed using an object-oriented programming paradigm. We present the components of each of these designs in C and C++.
Our approach was to compare and critique these two designs with respect to the extent to which their respective programming paradigms meet the software goals of software reusability and ease of program extension and maintenance. We designed the graphical user interface using TAE Plus which generated code in both C and C++, providing an easy way to transport the interface from a C implementation to a C++ implementation in the future. The designs of this real world Navy tactical decision aid clearly demonstrate the problems associated with using structured programming paradigm and the benefits of using an object-oriented programming paradigm, especially for large systems.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL C. Thomas Wu	22b. TELEPHONE (Include Area Code) (408) 646-3391; 22c. OFFICE SYMBOL CS/Wu

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

Approved for public release; distribution is unlimited

Structured Versus Object-Oriented Design of a Navy Battle Group Logistics Simulation System

by

Bernadette Clemente Brooks

B. S., Psychology, Georgetown University, 1980

M.A., International Studies, The Johns Hopkins University, 1988

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

March 1993

Author:

Bernadette Clemente Brooks

Approved By:

C. Thomas Wu, Thesis Advisor

Roger Stemp, Second Reader

CDR Gary J. Hughes, Chairman,
Department of Computer Science

ABSTRACT

This thesis deals with the design of a Navy battle group logistics simulation system to support battle group logistics coordinators. BGLCSS 2.0, the Battle Group Logistics Coordinator Support System, was designed and developed using a structured programming paradigm. A subset of BGLCSS 2.0 was then designed using an object-oriented programming paradigm. We present the components of each of these designs in C and C++.

Our approach was to compare and critique these two designs with respect to the extent to which their respective programming paradigms meet the software goals of software reusability and ease of program extension and maintenance. We designed the graphical user interface using TAE Plus which generated code in both C and C++. This mechanism provides an easy way to transport the interface from a C implementation to a C++ implementation in the future.

The design of this real world Navy tactical decision aid clearly demonstrate the problems associated with using structured programming paradigm and the benefits of using an object-oriented programming paradigm, especially for large systems.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank the students and faculty of the Computer Science and the Operations Research Departments at the Naval Postgraduate School for their interest and support throughout this work. In particular, I would like to thank Dr. C. Thomas Wu, Roger Stemp, and CDR Gary Hughes for their guidance and encouragement throughout this project. It was former Computer Science Department Chairman Prof. Robert McGhee's original suggestion to take on this project as part of my duties as a civilian computer systems programmer at the Naval Postgraduate School Computer Science Department and to pursue this research area as part of my thesis work.

I. INTRODUCTION

A. BACKGROUND OF BGLCSS

The Battle Group Logistics Coordinator Support System (BGLCSS) is a logistics simulation modeling tool to be used by battle group logistics coordinators to track, plan, and predict F-76 ship fuel, F-44 aircraft fuel, and ordnance states for ships in a battle group. Tracking these states involves applying various usage rates for each commodity and ship type based on the passage of time and the planning and scheduling of battle group events [SCHRADY 90]. Battle group events that can be planned include underway replenishment and consol. Events that can be scheduled include ship stationing events, raids, strikes, anti-submarine warfare (ASW) prosecutions, changes in ship or battle group course and speed, fuel and ordnance transfers outside the battle group, and changes in ASW or anti-aircraft warfare (AAW) threat level.

BGLCSS 1.0, written in Turbo Pascal 6.0 for a DOS environment, is the predecessor to BGLCSS 2.0 and was originally developed by the Operations Research Department at the Naval Postgraduate School. After the program was tested during Commander Second Fleet, Fleet Exercise (C2F FLEETX 3-90) in June, 1990, and during C2F FLEETX 1/91-2/91 in November, 1990, it was decided to move the program to the Navy Tactical Command System Afloat (NTCS-A) Unified Build 2.0. This system consists of a set of applications including the Joint Operational Tactical System (JOTS II) [INRI 91b].

JOTS II is an automated Command, Control, and Communications Display and Decision System designed to meet the tactical situation assessment needs of battle group/force commanders, surface warfare commanders, ship commanding officers, and shore command centers [INRI 92c]. JOTS II has digital interfaces with a variety of military communications systems and other shipboard computer systems. It processes tactical information received from other systems and automatically correlates this data with its existing tactical contact or Track Data Base Manager (TDBM). This tactical database is then used to generate computer graphics images at color Sun workstations [INRI 92a].

The workstations operate using a modified version of the standard commercial UNIX operating system SunOS 4.1.1. Applications are written in either C or Ada and use the X Window Manager and Government Off The Shelf Software (GOTS) programming tools. Among these GOTS tools, the Wizard Tool Kit, is a Motif-based C function library used for building graphical user interfaces [INRI 92d].

BGLCSS 2.0, a tactical decision aid within the NTCS-A Unified Build System architecture, is shown in Figure 1. It is written in C using a structured programming paradigm and, instead of using the Wizard Tool Kit for building the graphical user interface, uses Transportable Applications Environment Plus (TAE Plus), a User Interface Management System (UIMS).

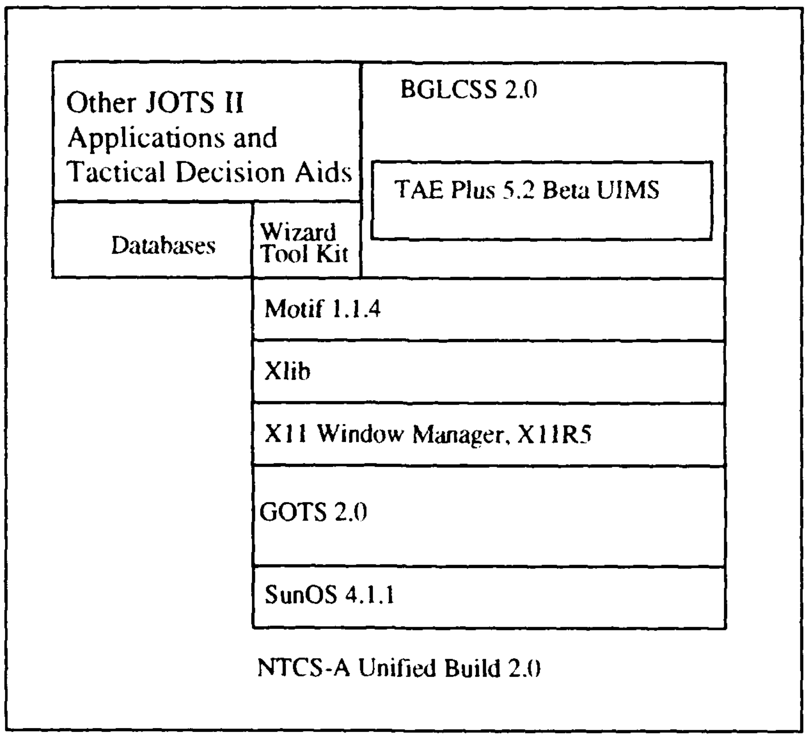

Figure 1: BGLCSS 2.0 Within the NTCS-A System Architecture

TAE Plus is a visual graphical user interface builder developed by the National Aeronautics and Space Administration Goddard Space Flight Center (NASA GSFC) and distributed by NASA's Computer Software Management and Information center (COSMIC), a non-profit unit of the University of Georgia. It supports rapid building, tailoring, and management of graphic-oriented user interfaces. The main features of TAE Plus include: 1) the Work Bench, an interactive tool that supports the design and layout of an application's interface; 2) the Window Programming Tools (Wpt) Package, a set of application callable subroutines used to control a user interface during execution time; and 3) the Code Generator, which automatically generates code for the interface in C, C++, Ada, or TCL (TAE Command Language) [NASA 91a].

This thesis deals with the design, implementation, maintenance, and extension issues of structured programming versus object-oriented programming for real-world applications such as the BGLCSS tactical decision aid within the NTCS-A Unified Build.

B. OBJECTIVES

This thesis was embarked upon to determine whether there are significant differences between the structured and the object-oriented implementation of the same application. The object-oriented paradigm promises, among other things, a more reliable end product, easier maintenance, and easier extension. We seek to demonstrate the benefits of using an object-oriented approach for a real-world application such as BGLCS and to argue that an object-oriented approach is particularly suited for large, multi-component systems such as the applications within the NTCS-A Unified Build.

C. SCOPE

The Navy Space and Warfare Command (SPAWAR) project specifications demanded that BGLCSS 2.0 be written in C using a structured programming paradigm. While much of this project has been devoted to developing robust algorithms to realistically simulate the logistics events in C, this thesis critiques the use of the structured programming paradigm, especially in large systems.

The decision to use a visual graphical user interface UIMS such as TAE Plus instead of the Wizard Tool Kit, a set of low-level Motif functions, significantly decreased overall development time. Furthermore, we propose that by using TAE Plus with C++, NTCS-A applications could benefit significantly not only from improved software development time, but also from improved product reliability, improved program maintenance, and easier program extension. Due in large part to the breadth of the BGLCSS application, only a subset of the structured paradigm implementation has been re-designed using an object-oriented paradigm in C++.

D. ORGANIZATION

Chapter II of this thesis provides an overview of the structured programming and the object-oriented paradigms. Chapter III presents an abbreviated presentation of the BGLCSS 2.0 graphical user interface design using TAE Plus. Chapter IV covers and analyzes the structured design and implementation of the structured programming version of BGLCSS in C. Chapter V covers and analyzes the object-oriented programming design of BGLCSS in C++. Chapter VI summarizes the work accomplished and provides recommendations for program maintenance and extension. The appendices contain the graphical user interface panels, the C and C++ program listings.

II. BACKGROUND OF THE PROGRAMMING PARADIGMS

A. GENERAL

Discussion about the merits of object-oriented programming have inundated technical journals over the past few years and has appeared even in non-technical journals. In a recent article in *Business Week* the question was asked, "Will object-oriented programming transform the computer industry?" The article goes on to describe in layman's terms the differences between "the old way", i.e. structured programming, and "the new way", i.e. object-oriented programming, of designing and writing programs. Structured programming was characterized by three terms: confusion, hand crafting, and breakdowns. In contrast, object-oriented programming was described by three counterpart terms: understanding, reusing, and repairing [HAMMONDS 91].

Not all supporters of the object-oriented paradigm agree that there is a sharp boundary between "the old way" and "the new way." Rather, object-oriented principles can be said to have *evolved* from the lessons learned from years of structured programming just as structured programming principles have evolved from the lessons learned from years of machine and assembly language programming. Holub, for example, cautions against looking for the "major shift in paradigm" promised by some object-oriented paradigm enthusiasts. In his book about programming with objects in C and C++, he introduces object-oriented principles by using code written in C within a structural programming paradigm, followed by code written in C within an object-oriented paradigm, and finally the object-oriented C++ version. He believes that object-oriented programming is merely a collection of useful programming techniques that can be applied to any computer language. An "object-oriented language" provides a few built-in mechanisms for operations that can be performed explicitly in a non-OOP language. Holub even argues that a program can be written in C in an object-oriented way just as easily as a program written in C++ and provides ample examples of code written in C that mimic some object-oriented principles [HOLUB 92]. It is not enough, however, that a program be written in an object-oriented way in C as in C++. Much of this code is very difficult for C and C++ programmers alike to understand even though they are accustomed to reading C code within a structured programming paradigm, the paradigm for which it was designed. Using a structured programming language to perform object-oriented techniques is a less than optimal fit of resources and is not the best application of the object-oriented paradigm.

It can even be argued that C++, while designed to support object-oriented programming, may not be the best language for object-oriented programming. Shiffman [SHIFFMAN 92] maintains that the pure object-oriented programming language Smalltalk is far easier to use than C++. With Smalltalk, it is possible to write more comprehensible, more maintainable programs and class reuse between applications is far more prevalent in contrast to C++. Given the SPAWAR project specifications to use C to implement BGLCSS 2.0, the advantages of using a pure object-oriented programming language such as Smalltalk are outweighed by the benefits of using C++. A move to C++, because it is a superset of C, provides a more evolutionary transition to an object-oriented paradigm than moving from C to an entirely different language with its own syntax and conventions.

B. GOALS OF SOFTWARE DEVELOPMENT

To distinguish between these two paradigms and determine whether or not the object-oriented programming paradigm is better than the structured programming paradigm, we need to identify specific goals of good software development. Meyer [MEYER 88] cites several "quality factors" of software development: correctness, robustness, extendibility, and reusability. Correctness is defined as the ability of software to exactly perform its tasks, as defined by the requirements and specifications. Robustness refers to the ability of software to function even in unintended conditions. Extendibility is a subjective measure of the ease with which a program can be changed to conform to a change in program specifications. For instance, a simple design is easier to adapt to specification changes than a complex one. Furthermore, the more autonomous or decentralized the program modules, the higher the probability that a simple change will affect just one module rather than trigger off a chain of reaction of changes over the whole system. Finally, reusability is the ability of software to be reused, in whole or part, for new applications and thereby significantly reduce development costs. With these software goals in mind, we can now discuss the extent to which each paradigm addresses these goals.

C. STRUCTURED PROGRAMMING PARADIGM

Machine and assembly languages reflect the architecture of the machines on which they operate. Like these machines, they are composed of data, arithmetic expressions, assignments to memory locations, and control flow. Among the problems associated with using these languages to develop software are the difficulties in debugging problematic code and the high cost of this tedious process. These problems were part of the impetus behind the development of higher-level languages such as Fortran and later, C. The C programming language has been described as a higher-level language that "rebuilds" an underlying machine to make it more convenient for programming [SETHI 90]. In general, the development of programming languages from the first generation languages to the current generation of languages has been a *continual* search for improved correctness, robustness, extendibility, and reusability. Five characteristics of the structural programming paradigm are of interest to our discussion: separate code and data, built-in data types, top-down functional decomposition and function-based design, distributed functionality, and limited code reuse.

1. Separate Code and Data

Procedures and functions operate on data and are defined and coded separate from each other (see Figure 2). This is particularly apparent in pre-ANSI C programs (see Appendix A). The data structures and simple data types are usually defined in header files, i.e. files with a ".h" at the end of the file name. The functions that operate on the data are usually defined in separate files with a ".c" at the end of the file name. When a driver program uses these functions and data, it is necessary to indicate to the C compiler preprocessor to put a copy of the header file in the driver program file when compilation occurs. Separately, the names of functions found outside of the main program are declared to be external to the file. The names of procedures or functions and global data must be unique so that they do not conflict with one another. There are two problems with the separation of these program components. First, procedures and functions can be called and passed the wrong data. Second, by unintentionally accessing data from procedures and functions, changes to data become uncontrollable in large systems where no one knows where a particular data item is being changed or why. These two weaknesses associated with the separation of code and data frequently reduces program correctness and robustness.

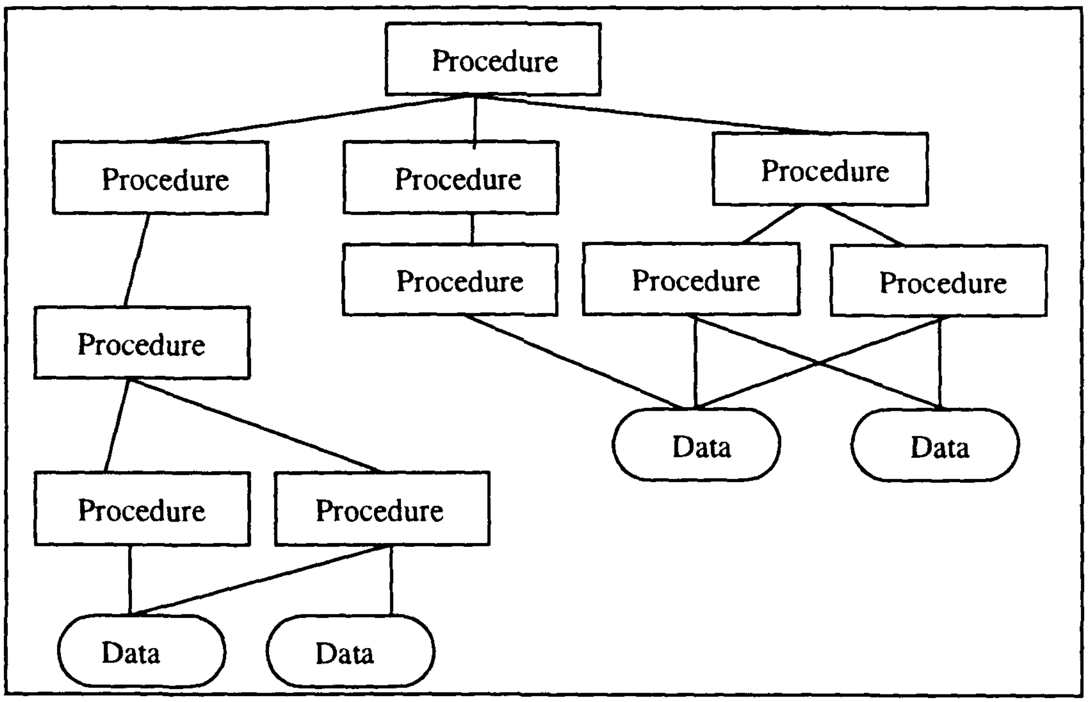

Figure 2: Structural Program Procedures Separate From Data

2. Built-In Data Types

While most languages implement integers and floating point numbers as built-in types, most do not implement complex numbers, calculations with physical units (barrels, tons, etc.), date time groups, and latitudes/longitudes. If these types are needed, they cannot be added easily. There are two problems with being limited to built-in data types. First, libraries become long lists of specialized functions rather than general functions handling a variety of types. The GOTS library is a case in point [INRI 92b]. It contains numerous functions that perform conversions from one built-in data type to another as shown in Table 1. This library is a proliferation of very specific functions that convert one built-in type representation of a real-world entity to another built-in type representation.

Table 1: A SAMPLE OF GOTS DATA CONVERSION ROUTINES

Function	Description	Arguments	Return Type
dtg	date time group string to integer	char tme[]	int
dtg_to_a	integer value of time to date time group string	int tme	char*
lat_to_A	double value of latitude to latitude string	double lat	char*
ll_to_A	double values of latitude and longitude to latitude/longitude string	double lat, double lng	char*

Since each GOTS library function requires the data type of the function arguments to be specified and only built-in types are allowed, it is difficult to write more general and therefore flexible library functions. This undermines reusability of the library functions. Second, it is possible to perform meaningless operations on the real world data being modelled. As an example, a date time group is often represented as an integer and yet it is still possible to add two date time groups together. This contributes to a poor representation of the real-world entity being modeled in software.

3. Top-Down Functional Decomposition and Function-Based Design

The top-down functional approach to program design is based on the premise that software development should be an incremental refinement of the system's abstract function *rather than the data it represents in the real world.* This process has often been described as a tree where the nodes of the tree represent elements in the decomposition and the branch nodes represent a refinement of its parent node. The primary benefits of the top-down functional approach is that it is logical and promotes organization and discipline. However, it fails to account for the evolutionary nature of software system development. Additionally, by focusing on the function rather than the data, a top-down approach does little to promote reusability. During the life of the system, it is the data and not the functions that are the most stable part of a system. The trade-off with this top-down approach is that it is fairly easy to design and develop an initial structure for the short term. However, in the long-run, as the system changes, it will be necessary to constantly redesign the system instead of merely extending the system. By focusing on the data instead of the immediate purpose of the system, the long-term benefits of reusability and extendibility can be achieved [MEYER 88].

Whether or not top-down design is used, structural programs are primarily function-based. Bottom-up design involves finding a set or sets of fundamental operations that are used throughout the program and writing procedures or functions to implement them. A top-down approach is then often used to connect the routines. In either a top-down or bottom-up approach, the emphasis is on the data upon which the program operates.

4. Distributed Functionality

Procedures and functions tend to be tightly coupled using this paradigm. That is, procedures and functions are distributed throughout a program and tend to know too much about other procedures and functions. Changes to one function may force changes in other functions. If, for instance, a new ship type such as the USS Arleigh Burke class needs to be incorporated into a program, all of the functions regarding ship type will need to be modified. Since these functions are hard-coded into the structure of a program, program extension becomes much more difficult to perform.

5. Limited Code Reuse

The structured paradigm inherently leads to repetition in programming. By using built-in data types and function-based design, code reuse can be achieved only by the low level method of cutting and pasting code from one program, and explicitly modifying it for its new data type (see Appendix A, InsertBGEvent, InsertBGHeader, and InsertBGHeader functions). Conventional languages require general purpose libraries which are limited to a long list of special purpose sort routines for sorting arrays of integers, floats, etc., instead of a generic sort function that is data-type independent.

D. OBJECT-ORIENTED PROGRAMMING PARADIGM

The object-oriented programming paradigm is a software design and development model incorporating several techniques explicitly aimed at code reuse, improved program reliability, and easier software modification, extension, and maintenance. The object-oriented programming paradigm is particularly well suited for the development of large sophisticated software systems that inherently evolve over time. Characteristics associated with the object-oriented paradigm include encapsulation, data abstraction, inheritance, polymorphism, persistence, delegation, and generacity. The first four characteristics are supported by the 2.1 version of C++, and, since C++ was the language chosen to design a subset of BGLCSS 2.0, this thesis will concentrate only on encapsulation, data abstraction, polymorphism, and inheritance issues. Generacity, a characteristic added to C++ version 3.0 and also known as parametric polymorphism, will be briefly mentioned. Although encapsulation and data abstraction are used within other paradigms, it will be shown that they are far more powerful within an object-oriented paradigm than within a structured programming paradigm.

How does the object-oriented paradigm lead to a more reliable end product, and easier software modification, extension, and maintenance? In general, the process begins by producing a more realistic representation of the real world entities involved. For example, by combining ship functions and ship data into a ship *object*, the real world ship entity is more coherently represented in code. An object is defined in terms of the data it encapsulates and the operations on the data that are allowed by the set of interface functions (see Figure 3).

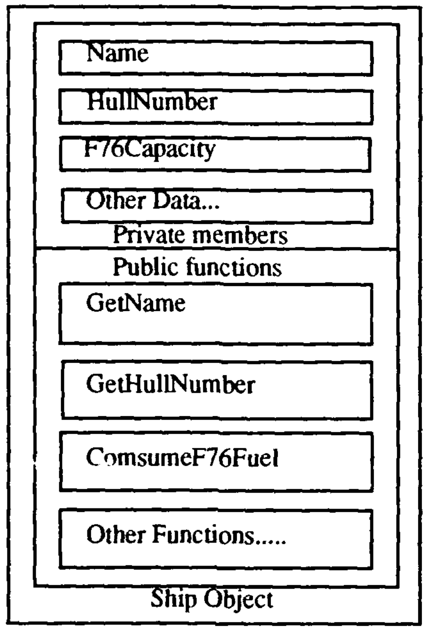

Figure 3: Object Encapsulating Related Functions and Data

A real world ship has attributes such as its name, hull number, F-76 fuel capacity, etc., and can perform various functions such as stationing, changing course and speed, etc. Ship attributes and the functions it can perform are said to be *encapsulated* within a ship object. The operations that can be performed on the encapsulated data are also known as the object's interface functions and the implementation of these functions is internal to the object. For example, a ship object frequently changes its course and speed which affects its internal fuel consumption and levels. These are attributes internal to the definition of a ship object.

Encapsulation allows for information hiding and the object-oriented paradigm's method of information hiding goes beyond local variables. Information hiding protects data from uncontrolled access and change. Variables that are local to a procedure or function, for instance, can only be changed within the function or procedure. In this way, they are protected from access from other functions and procedures. Global variables, on the other hand, are vulnerable to unintended change or access by the functions and procedures in their scope.

Objects, in C++, can have data or *function members* which are declared to be either public, protected, or private. These terms refer to the level of access to these members. For instance, all of a ship's data members can be declared to be private to the rest of the program. In this case, only a ship's member function can access private data to that object. Instead of passing variables to functions, it is the variables that are receiving messages in the form of functions. The variables themselves are designed to control which functions can modify which variables in the program.

1. Data Hiding and Data Abstraction

Data hiding is a practice whereby the programmer restricts him or herself to the public interface of a type for purposes of accessing or changing the value of an object of that type. The advantage of data hiding is that it encourages the programmer to protect data from unintended access and modification.

Data abstraction is defined as the activity of creating a model or concept of a real-world phenomenon at such a level that inessential details can be ignored. It is data typing combined with data hiding. In C++, the programmer creates user-defined types using the class mechanism. Abstract data types in C++ are not built-in data types such as integers or floating point numbers but are treated as though they were built-in. The term, user-defined data type, is a more appropriate term. C++ is an extensible language in that the language can be extended to include user-defined data types.

There are two benefits to data abstraction. First, it is easier to design and implement a system that is built out of entities that incorporate data hiding.The focus can be on the properties of the procedural interface of the various entities. These properties are typically far simpler and more abstract than the algorithms that implement them. Second, the decoupling of the interface from the implementation allows program entities to be reimplemented without having to modify any other part of the program. C++'s most important abstract data type is the *class*.

a. Class

A class is an internal template. In C++, it is the extension of a C struct. A struct (in other programming languages, it is known as a record) is a group of several different types of data in a single entity. An object is actually an instance of a class that occupies memory. For example, a class could be used to represent a ship. Although the actual data structure of a class is basically a struct with functions associated with it, it is considered as a single entity. A definition of a class in a C++ program conceptually introduces a new type into the language. The most significant feature of a class is that objects of the class can be treated the same way objects of a fundamental type such as integer [HOLUB 92].

b. Member Functions

A C++ class differs from a struct in C in that it contains member functions as well as data. The member function can be declared inside the class body, but are usually declared in a prototype form in the class definition and are defined later on. The need for member functions is related to data hiding mentioned earlier. Member functions can access private member data values. Member functions are one way to work with private data. Putting the member functions and member data together makes the data active and more cohesive.

c. C++ Constructors and Destructors

C++ provides constructor and destructor functions and almost every class contains these special member functions. They initialize and clean up class objects. Constructors provide a way to ensure that objects are defined with initial values without violating the constraints of data hiding. Destructors are important to free up object memory that was allocated by the constructor. A class can have several different constructors to allow objects to be able to be initialized in different ways with different argument. One of the greatest advantages of constructors and destructors is that they simplify software maintenance. If, for instance, the specification for the format for ship hull numbers were to be changed, only the class member data and functions would be changed. In structured programming, a "simple" change such as this one, can change hundreds or even thousands of lines of code that are closely tied to the format of the data [PERRY 92].

2. Inheritance and Class Relationships

Inheritance is used to describe special relationships between classes. It is generally used to achieve two goals: 1) It can be used as an abstraction tool to organize classes into hierarchies of specialization; and 2) It can be used as a code reuse mechanism to create a new class that bears strong resemblance to an existing class with added refinements.

In the first case, inheritance reflects a semantic relationship between classes where one class's member functions are used to refine some attributes of an existing base class while inheriting the remaining attributes. This form of inheritance could also be a mechanism to embody the strong similarities between two existing abstractions such that an object of a base class can be interchanged by an object of a derived class. The ability of a program to handle many forms of a class as though they were the same, an example of polymorphism, which is further discussed in the next section, is an important abstraction tool that shields the programmer from the implementation details of derived classes.

In the second case, inheritance is used to share code. While a class may not exhibit all the properties of its base class, it may be similar enough that the reuse gains make inheritance worthwhile. For example, although an ASW threat level event and a raid event are not interchangeable, they may have enough code in common that a common evolution through inheritance could lead to code sharing.

Coplien identifies four specific relationships between classes that affect class design and class hierarchy design: the *is a*, *has a*, *uses a*, and *creates a* relationships. The *is a* relationship is one where specialization or subtyping is involved. A destroyer *is a* type of ship and therefore, a destroyer class could be derived from a ship class. Another relationship between classes is called composition or a *has a* relationship. For example, a ship *has a* F-76 fuel state. Composition is implemented by either designating one class as a member of another or by one class referencing another object. A *uses a* relationship exists when a member function of a class takes an instance of some other class and uses it as a parameter. A *uses a* relationship exists when one class member functions calls on the services of some other class. A *creates a* relationship is similar to a uses a relationship but it is between an object and a class instead of between objects. In other words, an instance of one class, during the execution of one of its member functions, makes a request of some other class to create an instance of an object of the class. The identification of these relationships significantly influences the details of class design and class hierarchy design [COPLIEN 92].

Class hierarchy design can take on two different forms: single or multiple inheritance. Single inheritance allows the creation of a new class by using an existing class as a model. In other words, it is possible to inherit one class from another, creating new classes from existing ones. The original, or base class, shares its code with the new class. Code sharing can make programming much easier because less code needs to be rewritten. In some cases, new features need only be added to the new, derived classes. Multiple inheritance, by contrast, is the creation of a new class from multiple existing classes. C++ implements both single and multiple inheritance.

3. Polymorphism

When writing C++ programs, the same function name can be used with different types of objects. Polymorphism means "many different forms." Perry asserts that polymorphism is one reason why C++ programs can be written faster than C programs. For instance, in C++, various objects can be printed out with the same function name:

```
Ship.print(); // Prints the Ship object
BGEvent.print(); // Prints the BGEvent object
```

By using the same function name for different objects to perform the same operation such as printing, the "clutter" is removed from the code. There is no need to name the functions specific to the data type being manipulated as shown in the following C code:

```
PrintShip(); /*Prints a Ship data structure*/
PrintBGEvent(); /*Prints a BGEvent data structure*/
```

On the surface, the ability to use the same function name print() for several different objects merely seems to be a syntactic advantage of removing "clutter." However, what is really happening is that the same message or function name causes different responses depending on which object is receiving the message. One benefit of polymorphism is that it models the real world object better than the structured paradigm by using one command to be issued to different objects.

With polymorphism, each time a new object is added to a program, no changes to the existing code are required. All that is required is that a new function is added to the object. Therefore, polymorphism facilitates program extension by eliminating the need to alter the existing program code.

Parametric polymorphism, a mechanism supported by C++ 3.0 as templates, has been argued to be the mechanism allowing for the highest degree of code reuse of all of the object-oriented mechanisms. Templates in C++ are similar to generic packages in Ada. Templates define families of types and functions. They are an alternative to inheritance hierarchies constructed for code reuse. For example, a template class can be designed to implement a doubly linked list and can be used for any type. This kind of reuse is based on the reuse of source code rather than object code. By contrast, much of the functionality of a base class can be reused by a derived class at the object code level. Template classes can be used to build generic class libraries.

E. COMPARISON OF THE PARADIGMS

In comparing the two paradigms, it is useful to point out each paradigm's strengths and weaknesses in order to determine which paradigm contributes more to the goals of software engineering as described by Meyer, i.e. correctness, robustness, extendibility, and reusability.

The strengths of the structured programming paradigm are limited. First, in comparison to developing code in an ad hoc manner, structured programming and the top-down functional decomposition approach provides organization to complex system development. Fortran, assembly language, and machine code programming pale in comparison to code written using a structured programming paradigm.

The weaknesses of this paradigm, however, are numerous. First, by separating the code from the data, there is the possibility of passing the wrong data to the code and data can be accessed and changed in uncontrollable ways. This reduces the correctness and the robustness of a program. Second, most languages associated with the structured programming paradigm are limited to built-in data types with the exception of Ada and Modula-2. In general, the ability to create a general library of functions results in either a proliferation of library functions or the lack of them. This weakness affects reusability. Also, meaningless operations such as addition can be performed on entities such as a date time group. Third, the down side of a top-down functional decomposition approach is that it inherently leads to requiring repeated redesigns when program extensions are required. Fourth, with functionality distributed throughout a program, changes to one function may force changes to other functions. This compromises program extension efforts as well as program correctness and robustness.

The object-oriented programming paradigm provides more techniques to achieve the software goals of correctness, robustness, extendibility, and reusability than in the structured programming paradigm. It can be said that some elements of the structured programming paradigm have been incorporated within the object-oriented programming paradigm. An obvious example is the emphasis on an organized, top-down, high-level approach to design.

Although data abstraction and data hiding are supported in the structured programming paradigm, they are more powerful within the object-oriented paradigm. The primary reason for this stems from the combination of data and functions within an object of a class. The object of a class provides a more realistic representation of the real-world entity and, when used within a program, limits the degree to which a change in an object's data or functions affects other parts of the program and forces a cascade of changes. Polymorphism, a technique used when calling functions of the same name for objects of different classes, makes program code easier to read. Template classes, a C++ feature to create generic classes, allows the building of truly general class libraries which is not possible with a structured programming paradigm.

III. BGLCSS 2.0 GRAPHICAL USER INTERFACE DESIGN

A. TAE PLUS OVERVIEW

The TAE Plus version 5.2 beta release was developed under SunOS 4.1.1 on a SPARCstation 2. Components used to develop TAE Plus include Release 4 of the X Window System (X11R4), Open Software Foundation (OSF) OSF/Motif 1.1.4, InterViews 2.6 from Stanford University; and GNU g++ version 1.40.3. As far as C++ code generation is concerned, the TAE Plus installation source code tree was subsequently rebuilt to link statically with SunC++ 2.1. While the above components constitute the only officially supported platform for this beta release, TAE Plus version 5.2 beta was also tested with OSF/Motif 1.1.3. The TAE Plus installation guide urges users to upgrade to OSF/Motif 1.1.4 (or 1.1.3) as soon as possible to take advantage of the many problems fixed by OSF. However, the release of TAE Plus used for this thesis is expected to work with versions 1.1.1 and 1.1.2 of OSF/Motif as well, although behavior may vary [NASA 92b].

TAE Plus is a graphical user interface builder used at three different levels. First, it is used at the WorkBench level as shown in Figure 4 to visually design panels (windows) and items (buttons, selection lists, labels etc.).

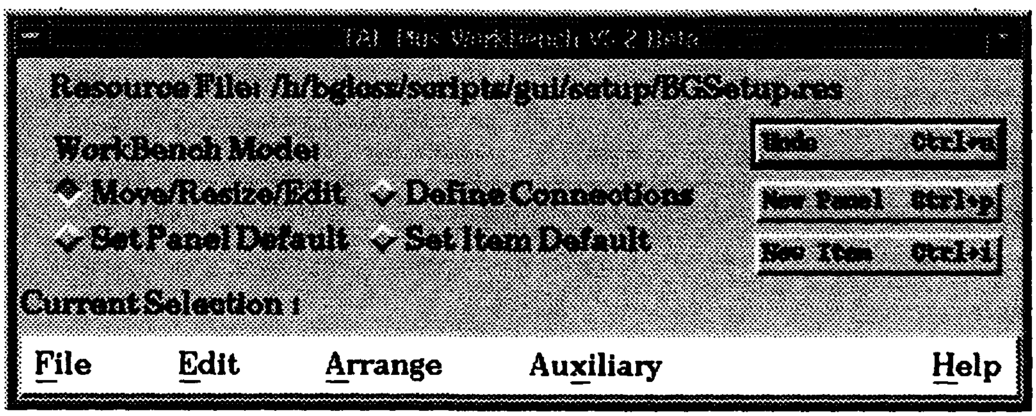

Figure 4: TAE Plus WorkBench Panel

Also at this level, panels and items are named, defined, and various constraints and details such as whether or not the item is an "event-generating" one, are added. The term "event-generating" should not be confused with the battle group events simulated in BGLCSS 2.0. An item is event-generating if it causes another panel to be displayed or if a call to a function is made. At this point, all of the information created for an interface is contained in a resource file. The format for the name of a resource file is the application name appended by a `.res` suffix (see Figure 5).

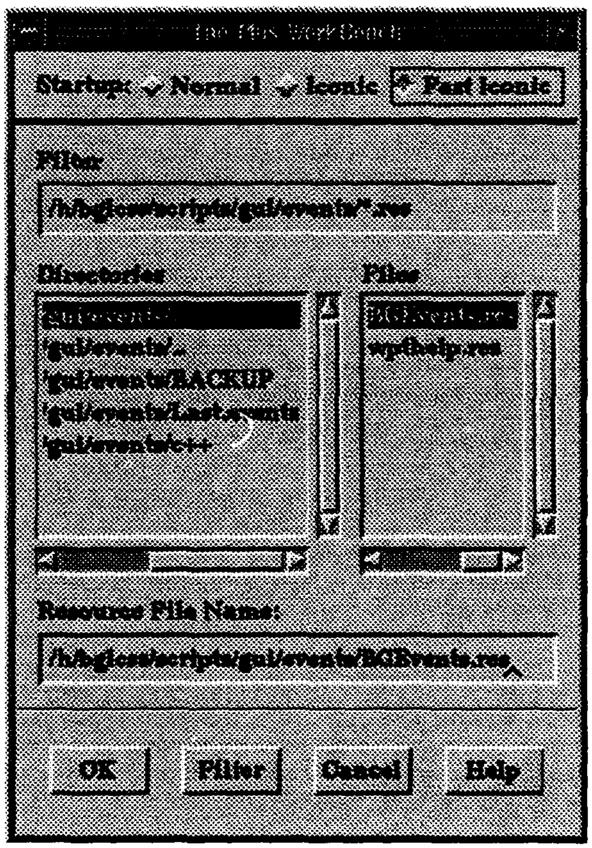

Figure 5: TAE Plus WorkBench Resource File Selection Panel

Second, TAE Plus generates code for all of the event-generating items in the application panels. While an in-depth discussion of event-driven programming is beyond the scope of this thesis, it is important to understand the difference between an event-generating item and a non-event-generating item. An event-generating item is an item such as a button or a selection list that, when pushed or when an item is selected, another interface activity takes place. A non event-generating item is an item such as a text keyin item where users enter data.

Third, at the source code level, it is the application programmer's task to integrate the calls to application library functions with the code generated by TAE Plus. When this project began, TAE Plus Version 5.1 generated only Ada, C, and TAE Plus Command Language code. In the meantime, a beta version of TAE Plus was released providing code generation in C++. Therefore, it was possible to build a graphical user interface and generate C code for the structured programming version of BGLCSS 2.0 as well as C++ code for the object-oriented programming version of a subset of BGLCSS 2.0. The integration at the source code level will be discussed in chapters IV and V with respect to each of the implementations: in C and in C++.

B. TAE PLUS WORKBENCH

At the WorkBench level, the first step in creating an application is to create an initial panel and to define the panel's specifications such as the panel for the battle group course and speed event as shown in Figure 6.

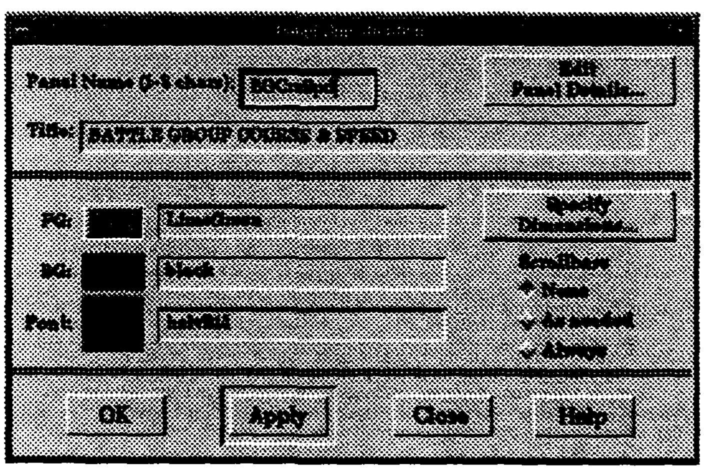

Figure 6: TAE Plus Panel Specification Panel

By pressing the Edit Panel Details button on the panel depicted in the last figure, the Panel Details Panel is displayed (see Figure 7). It is used to define the panel's help file and icon details. Applications built with TAE Plus require an initial panel(s) to be designated as in the BGEvents module of BGLCSS 2.0 (see Figure 8).

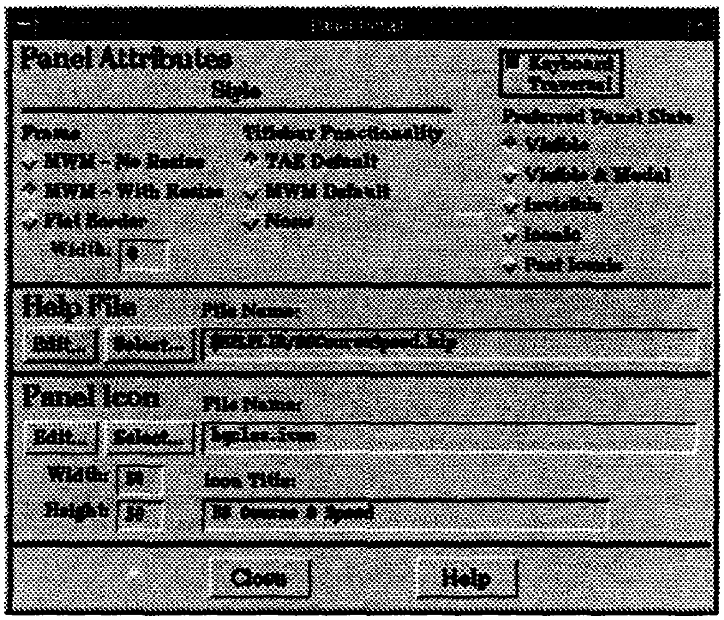

Figure 7: TAE Plus Panel Details Panel

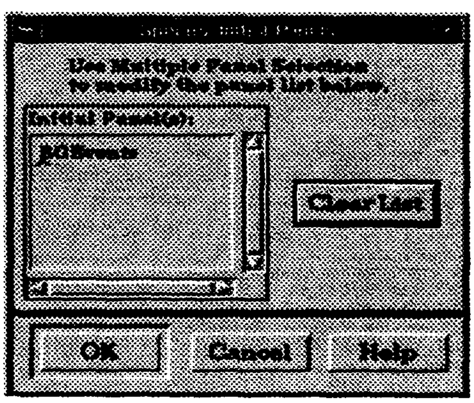

Figure 8: TAE Plus Specify Initial Panels Panel

Panels can be connected to event-generating buttons or calls to application functions using the Connection Specification Panel as shown in Figure 9.

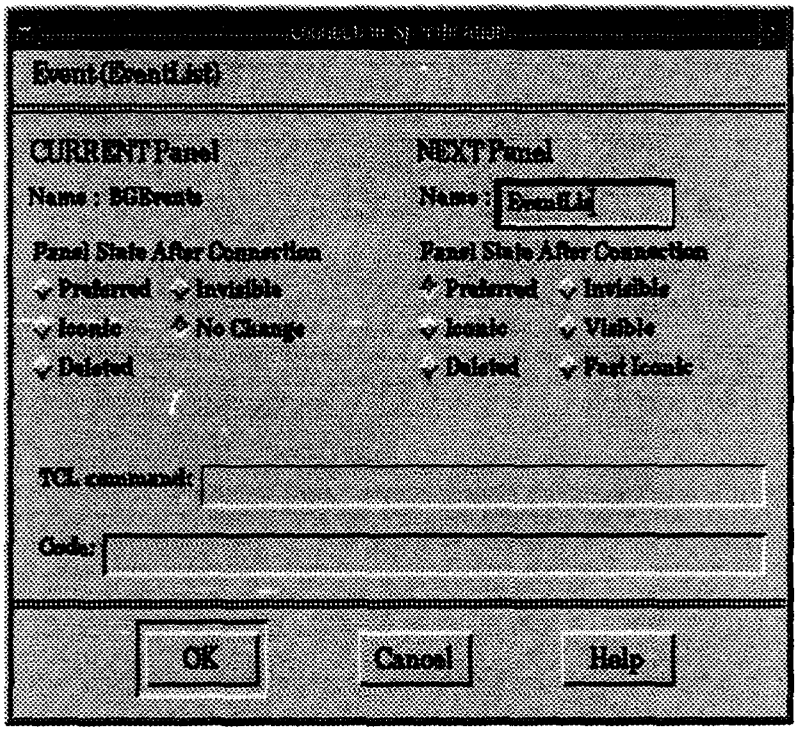

Figure 9: TAE Plus Connection Specification Panel

Once a panel is created, items can be defined and added to the panel using the Item Specification Panel in depicted in Figure 10. When an item is defined, one of its most important attributes is its data type. Based on data type, TAE Plus provides general type checking on user-entered data. For example, if an item is defined as a real number, and the user enters anything else, i.e. an integer or a string, an error message is displayed by TAE Plus to the user indicating that the data type is invalid and restored the previous entry. This general data type checking provided by TAE Plus eliminates the need to write error handling routines of this nature by the applications programmer.

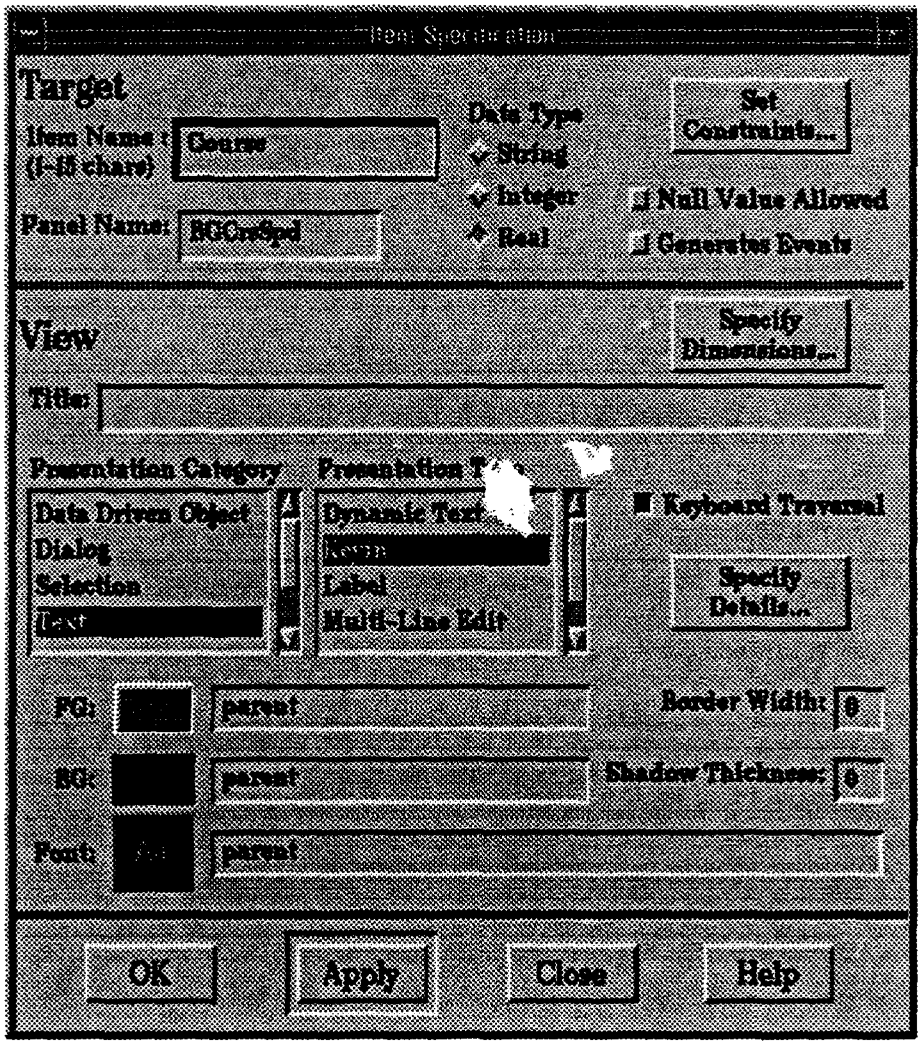

Figure 10: TAE Plus Item Specification Panel

User entered data can be further restricted by setting constraints as show in Figure 11. For instance, a constraint on the range of valid values can be specified on a text keyin real data type item. A course heading, in the real world, has a range between 0.0 and 359.9. When real value is entered that falls outside of the specified range, an error message is displayed to the user and the previous value is restored to the text keyin interface item.

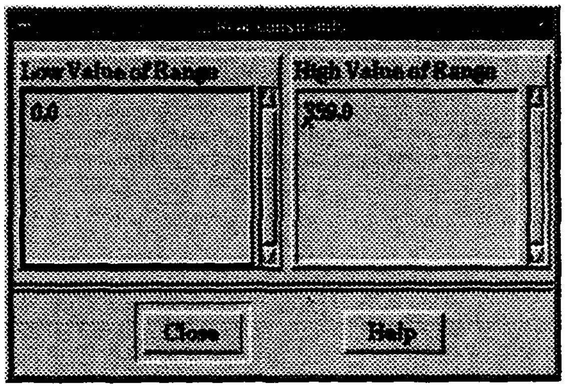

Figure 11: TAE Plus Item Constraints Panel

TAE Plus also makes the implementation of a help button easy to define. When a button is defined, the specific details offered to the designer are displayed in the panel shown in Figure 12.

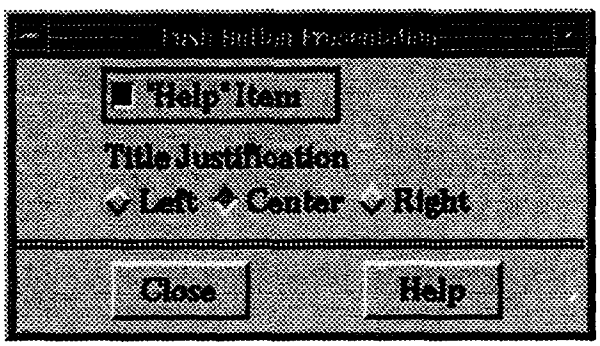

Figure 12: TAE Plus Push Button Presentation Panel

Furthermore, application specific error messages can be defined (see Figure 13). The message item provides a template for making information, warning, and error messages.

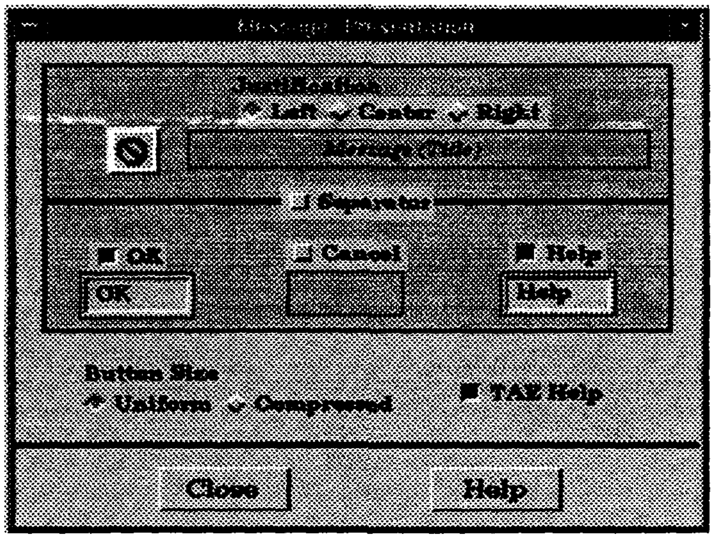

Figure 13: TAE Plus Message Presentation Panel

C. BGLCSS 2.0 GUI DESIGN

BGLCSS 2.0 consists of three program modules: Setup, Events, and Overview. When integrated into the NTCS-A Unified Build, BGLCSS 2.0 is one of several JOTS II Tactical Decision Aids (TDAs) as shown in Figure 14.

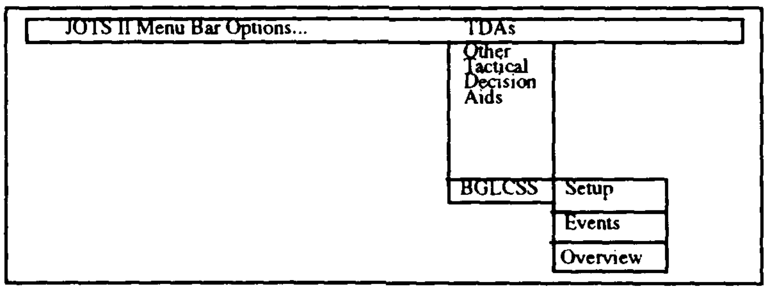

Figure 14: JOTS II Menu Tree

Each one of these modules was built using three separate TAE Plus application and in accordance with the User Interface Specifications for Navy Command and Control Systems [FERNANDES 92]. A script is used to call individual JOTS II applications. The design decision to separate BGLCSS 2.0 into three modules was based on the way simulation applications are used. In order to simulate logistics events, it is a prerequisite to define some settings and create the entities with which to work. By dividing BGLCSS 2.0 into Set Up, Events, and Overview, the user is directed to the three main services of the application. Once set up is performed, events can be simulated. In addition, although help buttons are present on every panel and provide clear instructions to the user about using the items on a specific panel, it was decided to include an Overview module where a "Help on Help" on-line user's manual could be provided.

Each module's C code was generated by TAE Plus based on the initial panel designated, connections between items and panels, and items designated as event-generating. A partial depiction of the files generated for the BGSetup module (see Figure 15).

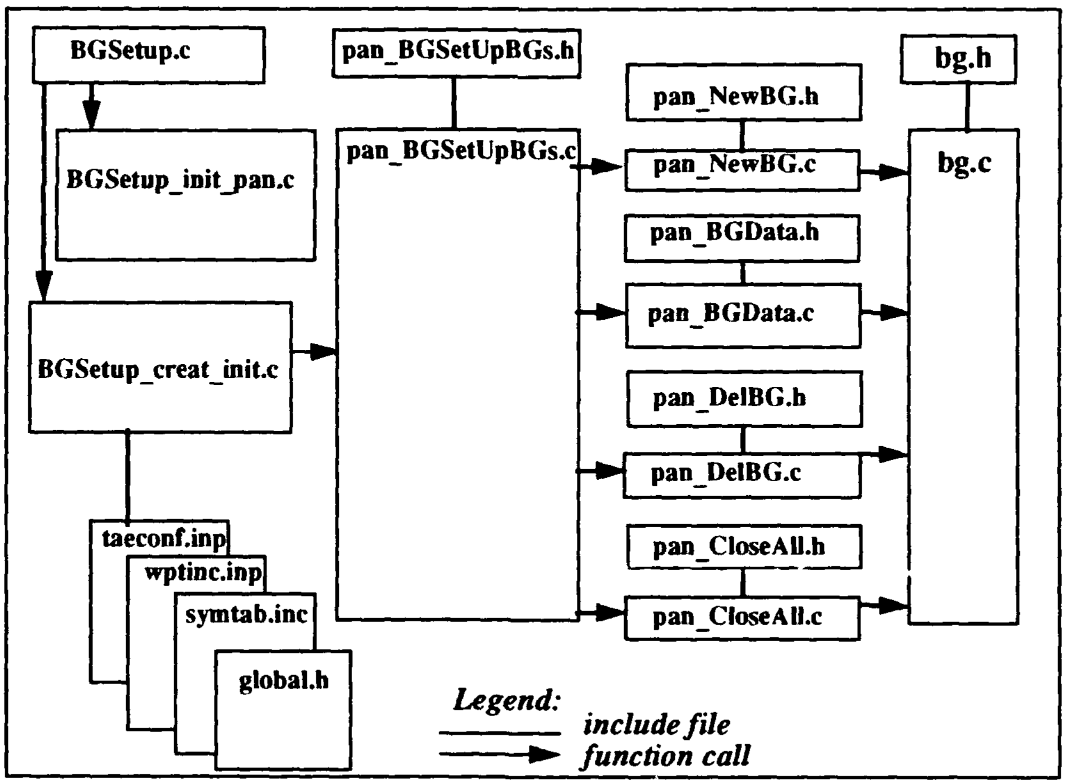

Figure 15: TAE Plus Files Generated and Function Invocation for Setup Module

The `BGSetup.c` calls two functions, BGSetup_Initialize_ALL_Panels () found in `BGSetup_init_pan.c` and BGSetup_Create_Initial_Panels() found in `BGSetup_creat_init.c`. The first function reads the resource file created by the TAE Work Bench and creates and initializes the TAE objects used for each panel in the application. The second function merely calls the specific function to create the initial panel, in this case, SetUpBGs_Create_Panel() found in `pan_SetUpBGs.c`.

After calling these two functions,the main event handling loop runs the entire program until the SET_APPLICATION_DONE flag is set. At this time, the program terminates. During the main event loop, the functions found in the panels NewBG, BGData, DelBG, and CloseAll are called when event generating operations are performed by the user.

In addition to the TAE Plus code generated files, TAE Plus provides two basic sets of library functions for applications programmers: the Windows Programming Tools package (Wpt) and the Variable Manipulation (Vm) package.

The Wpt package contains C functions that provide programs with graphic, window-based user interfaces. The purpose of these functions is to deliver user inputs to an application program. Based on information provided in the resource file, Wpt determines the desired form of user interaction and creates the appropriate displays on the screen. When the user enters or selects values for the program inputs, Wpt functions make the values or selection available to the program.

The Vm package consists of standard TAE data structures called Vm objects. When an application program begins, it reads the panel and item information contained in the resource file into these Vm objects. Wpt uses two Vm objects to acquire each set of inputs. The "target" Vm object describes the inputs to be acquired. The "view" object describes the presentation of the target parameters on the screen. After the user interface information is read into the Vm objects, the program passes pointers to the objects as arguments to Wpt functions.

The generation of C++ files is similar to that of the C files. Table 2 provides a sample of the files produced by TAE Plus in C++. These files are found in Appendix C.

Regardless of the programming language used, the following figures are the initial panels for each of the three program modules Set Up, Events, and Overview. For a more complete presentation of the graphical user interface, see Appendix A.

Table 2: SAMPLE OF BGLCSS 2.0 SETUP C++ GENERATED FILES

File	Description
`BGSetup.cc`	Contains main procedure.
`BGSetup.h`	Encapsulates the resource file.
`BGSetup_init_pan.cc`	Initializes all panels.
`BGSetup_creat_init.cc`	Creates the initial panel set.
`pan_BGSetupBGs.h`	Contains the panel's class definition and instance parameter declarations.
`pan_BGSetupBGs.cc`	Contains the panel's class methods.
`item_BGSetupBGs.h`	Class definitions of all items in the panel and instance parameters declarations.
`Imakefile`	Machine-independent template for generating the Makefile for the application.

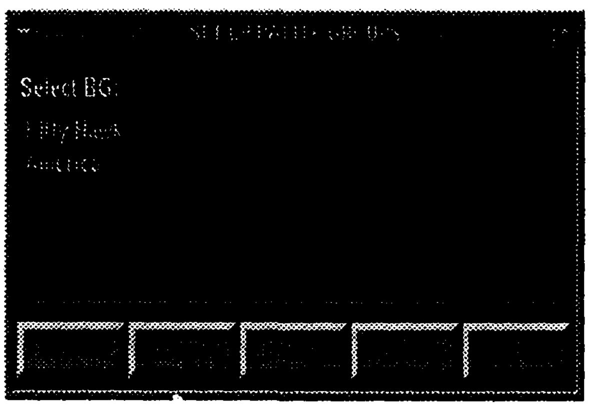

Figure 16: BGLCSS 2.0 Set Up Battle Groups Initial Panel

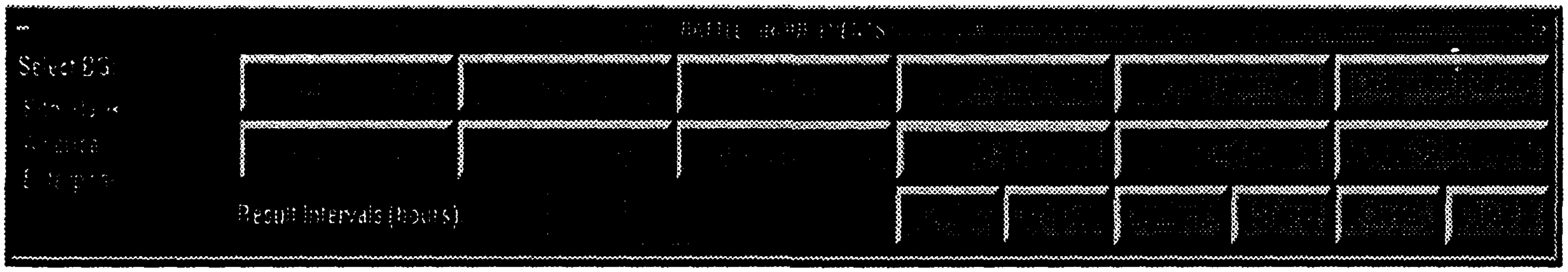

Figure 17: BGLCSS 2.0 Battle Group Events Initial Panel

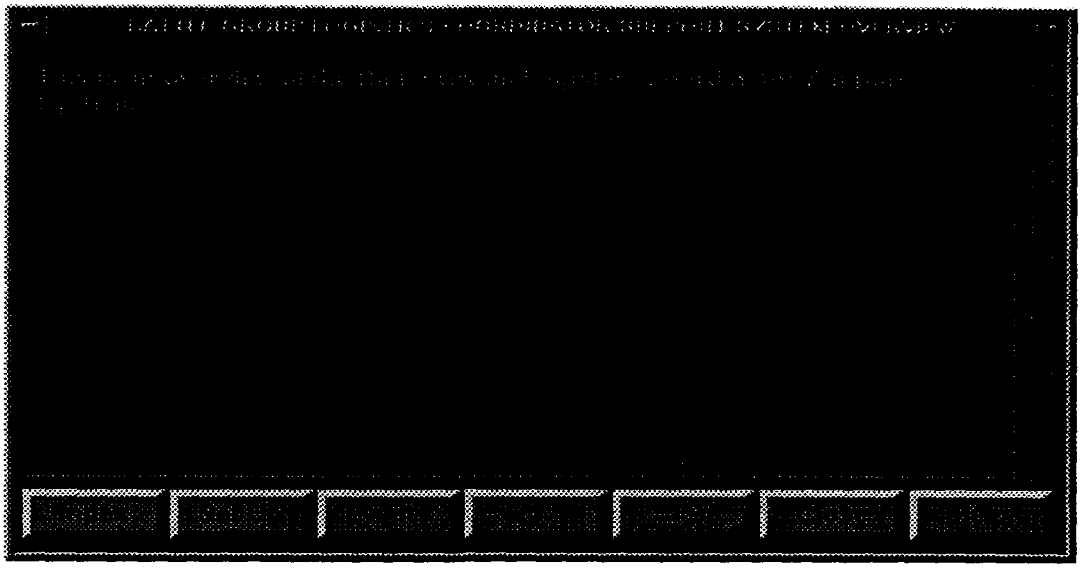

Figure 18: BGLCSS 2.0 Overview Initial Panel

IV. BGLCSS 2.0 STRUCTURED DESIGN

A. GENERAL

This chapter describes the design decisions and implementation issues of BGLCSS 2.0 using a structured programming paradigm. The following chapter will discuss the design decisions and implementation issues of a part of BGLCSS 2.0 using an object-oriented programming paradigm.

BGLCSS 1.0 was written in Turbo Pascal to run in a DOS operating system on an IBM PC-AT compatible microcomputer [SCHRADY 91]. According to the proposal for research statement of work,"Moving BGLCSS into JOTS involves much more than code conversion. The JOTS II environment includes communications interfaces and a variety of services which must be utilized to reduce the labor intensiveness of the current version of BGLCSS. The data, planning factors, and algorithms in the current version of BGLCSS will carry-over, but the program must be completely restructured."

In other words, by moving BGLCSS 1.0 into the JOTS II environment, the program would require restructuring solely in order to take advantage of the variety of JOTS II services and to reduce user data entry activity. Among other things, it was assumed that *the algorithms would carry over*. In fact, however, after extensive examination of the original Pascal code, this was found not to be the case.

The algorithms in question, as far as the code is concerned, represent the battle groups, ships, and events. It was thought that the original logic of the program, could be easily translated to another programming language, a new interface attached, and JOTS services incorporated. It was not feasible to translate or carry over any of the original Pascal code for three reasons.

First, the original program was developed in a rapid, iterative, and ad hoc manner and the Pascal code is a classic example of one of the weaknesses of developing software without a high-level design. The code was largely undocumented and no high-level descriptions of algorithms, function definitions, nor variable definitions were provided.

Without any of this program documentation, the original BGLCSS code was virtually untranslatable to C.

Second, the data structures used to represent the simulated events were ill-suited for simulation purposes and introduced unnecessary inefficiencies and redundancies into the program.

Finally, the organization and layout of the user interface was complicated and difficult to use. For these reasons, BGLCSS 2.0 was completely restructured to take advantage of the variety of JOTS II services and, furthermore, to redesign and redevelop the algorithms, data structures, and user interface.

The following sections present the structured design and implementation of BGLCSS 2.0. The components of the design are: the program specifications; symbolic constants; data structures for battle groups, ships, and events; library design of battle group, ship, and event functions; graphical user interface design; and application integration issues.

B. PROGRAM SPECIFICATIONS

The goal was to produce a program that simulates the occurrence of a number of logistics events and calculates the usage of fuels and ordnance based on specific planning factors. Program specifications were abstracted from the user's manual for BGLCSS 1.0 [SCHRADY 91].

C. SYMBOLIC CONSTANTS

In BGLCSS 1.0, symbolic constants were scattered throughout the multiple file program. Consequently, this added an unnecessary layer of confusion when trying to understand where the constants were defined. As a result, all of the symbolic constants in BGLCSS 2.0 are contained in one file, bg.h. shown in Figure 19. The preprocessor directive **`#define`** gives names to constants, also known as literals. By using these literals, a change can be easily made in one place and take effect throughout the program.

```
#define MAXNAME            25
#define MAXF76COEF          3
#define MAXSHIPTYPES        8
#define MAXORD            100
#define MAXRATES            6
#define MAXACFT            20
#define MAXTHREATLEVELS     2          /* Low, Medium, High */
#define MAXENGAGEMENTS      2          /* Raid, Strike, ASW */
#define MAXINTERVALS        3
#define HOURSINDAY         23
#define MAXUSETYPES         5
#define MAXBGSHIPS         30
#define MAXBGS             10
#define MAXORD            100
#define MAXSHIPS          100
#define DTGLENGTH          15
#define MAXLENGTH          25
#define F76DATA            "/h/bglcss/scripts/data/F76.dat"
#define BGDATA             "/h/bglcss/scripts/data/BGData.dat"
#define BGSHIPS            "/h/bglcss/scripts/data/Ships"
#define NAVYSHIPS          "/h/bglcss/scripts/data/NavyShips.dat"
#define EVENTSDATA         "/h/bglcss/scripts/data/Events"
#define HEADERSDATA        "/h/bglcss/scripts/data/Headers"
```

Figure 19: Symbolic Constants for Battle Groups and Ships

D. DATA STRUCTURES

There are two different kinds of data used in this program: battle group data and event data.

1. Battle Groups

Battle groups are represented by an array of battle group records or structs. Each battle group is a struct containing a name, a designation, a structure containing settings information, a structure containing location information, an array of ship structures, and an array of capacity information structures as shown in Figure 20. The battle groups are represented by arrays of structures. The most frequent activities associated with the battle groups involve data access, i.e. reading or writing data. Insertions and deletions of structures to the battle group are performed without regard to the order of structures within an array. Access to structures of an array is performed easily using an index to an array.

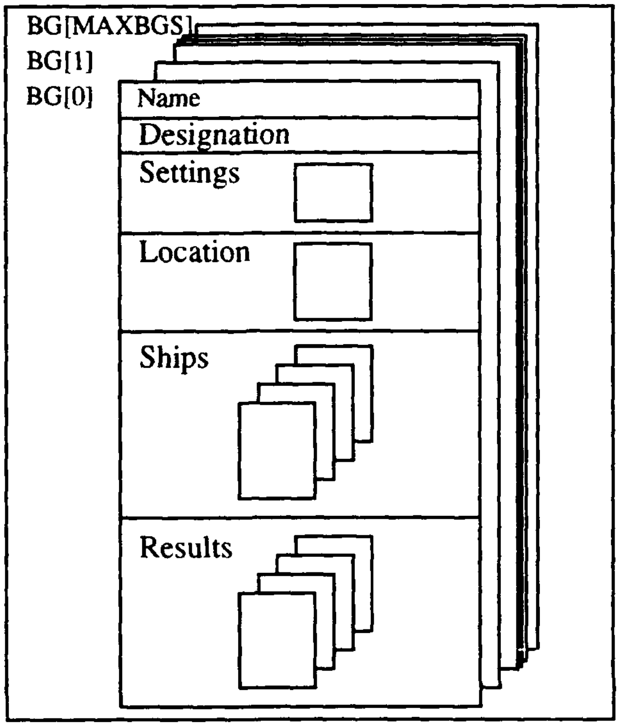

Figure 20: Battle Group Data Structures

Figure 21 shows the C code that defines the battle group data struct. This definition is found in the bg.h file contained in Appendix B.

```
typedef struct {
        char            Name[MAXNAME];
        char            Designation[MAXNAME];
        SettingsInfo    Settings;
        LocationInfo    Location;
        ShipInfo        Ships[MAXBGSHIPS];
        CapacityInfo    Results[MAXBGSHIPS];
}BGInfo;
```

Figure 21: Battle Group Information Type Definition

The first four variables contained in the settings struct shown in Figure 22 are "trip-wire" reserve levels that are set by the user so that when reserve levels fall below these levels, the user is notified. FuelRes represents F-76 fuel reserve level, CLFFuelRes represents CLF fuel reserve levels, OrdRes represents ordnance reserve levels, and CLFOrdRes represents CLF ordnance reserve levels. MaxF76 and MaxF44 each represents the maximum fuel capacities for F-76 and F-44 fuels respectively. StationSpeed represents battle group ship stationing speed, UnrepSpeed represents underway replenishment battle group ship speed, and AcftShipSpeed represents battle group aircraft carrier ship speed. Each of these variables are represented by a floating point type.

PredictStart is used to represent the integer value of the date time group supplied by the user. This variable represents the first time interval for which commodity percent capacities are to be computed. The following array of integers, PredictHours, is used to hold the number of hours to offset each of the three intervals of time for which commoditiy percent capacities are to be computed.

```
 typedef struct {
        float   FuelRes,
                CLFFuelRes,
                OrdRes,
                CLFOrdRes,
                MaxF76,
                MaxF44,
                StationSpeed,
                UnrepSpeed,
                AcftShipSpeed;
        int     PredictStart;
        int     PredictHours[MAXINTERVALS];
}SettingsInfo;
```

Figure 22: Settings Information Type Definition

The location information struct is used to hold the current battle group location information as shown in Figure 23. This same struct definition is also used within the ship struct.

```
typedef struct {
        int     Dtg;
        float   Speed,
                MaxSpeed;
        double  Latitude,
                Longitude,
                Course;
}LocationInfo;
```

Figure 23: Location Information Type Definition

Capacity information contains three arrays as shown in Figure 24. The F76Capacity and F44Capacity arrays are designed to each contain MAXINTERVALS floating point values, one for each time interval.

```
typedef struct{
        float       F76Capacity[MAXINTERVALS],
                    F44Capacity[MAXINTERVALS];
        OrdCapInfo  OrdCapacity[MAXINTERVALS];
}CapacityInfo;
```

Figure 24: Capacity Information Type Definition

2. Ships

The ship struct shown in Figure 25 contains string variables (in C, strings are represented as arrays of chararacters) for the name and hull number of the ship. The Type Combatant is represented as an enumerated type having one of the following values: Air, Combatant, Station, and Shuttle.

Identical to the battle group struct, the ship struct also contains a location struct as shown in Figure 26 .

```
typedef struct {
        char            Name[MAXNAME];
        char            Hull[MAXNAME];
        CLFType         TypeCombatant;
        LocationInfo    Location;
        F76Info         F76;
        F44Info         F44;
        int             Approach,
                        BreakAway;
        OrdInfo         Ord[MAXORD];
        AcftInfo        Acft[MAXACFT];
}ShipInfo;
```

Figure 25: Ship Information Type Definition

```
typedef struct {
        int     Dtg;
        float   Speed,
                MaxSpeed;
        double  Latitude,
                Longitude,
                Course;
}LocationInfo;
```

Figure 26: Location Information Type Definition

The next variable is a struct, containing data specific to the F-76 state as shown in Figure 27. The first seven variables are integers used to represent the ship capacity in gallons, the receive rate in gallons per minute, the transfer rate in gallons per minute, the current number of gallons on hand, the most recent estimated number of gallons on hand, the integer value of the date time group of the on hand reading, and the date time group of the estimated on hand value. The last variable is an array of coefficients used in predicting fuel consumption [SCHRADY 90].

```
typedef struct {
    int       Capacity,
              ReceiveRate,
              TransferRate,
              OnHand,
              EstOnHand,
              OnHandDtg,
              EstOnHandDtg;
     float    Coef[MAXF76COEF];
}F76Info;
```

Figure 27: F-76 Ship Fuel Information Type Definition

The struct shown in Figure 28 contains data specific to the F-44 aircraft fuel state and is almost a mirror image of the previous F-76 struct mentioned above except that the F-44 struct does not contain a coefficients array.

```
typedef struct {
   int   Capacity,
         ReceiveRate,
         TransferRate,
         OnHand,
         EstOnHand,
         OnHandDtg,
         EstOnHandDtg;
}F44Info;
```

Figure 28: F-44 Aircraft Fuel Information Type Definition

An array of OrdInfo structs, as shown in Figure 29, is contained in the ship struct. Each OrdInfo struct represents data specific to one ordnance item. The variables contained in this struct are similar to the previous two fuel structs except that the OrdInfo struct contains an array of use rates. MAXRATES refers to the six different use rates: low, medium, or high average threat levels, and raid, strike, or ASW events.

```
typedef struct {
        OrdName    Name[MAXNAME];
        int        TotalNumber,
                   Capacity,
                   Range,
                   TransferRate,
                   OnHand,
                   EstOnHand,
                   OnHandDtg,
                   EstOnHandDtg;
        OrdUse     UseRate[MAXRATES];
}OrdInfo;
```

Figure 29: Ordnance Information Type Definition

3. Events

A doubly linked list is a data structure consisting of structs connected to each other by pointers to the next and to the previous struct, as shown in Figure 30 .

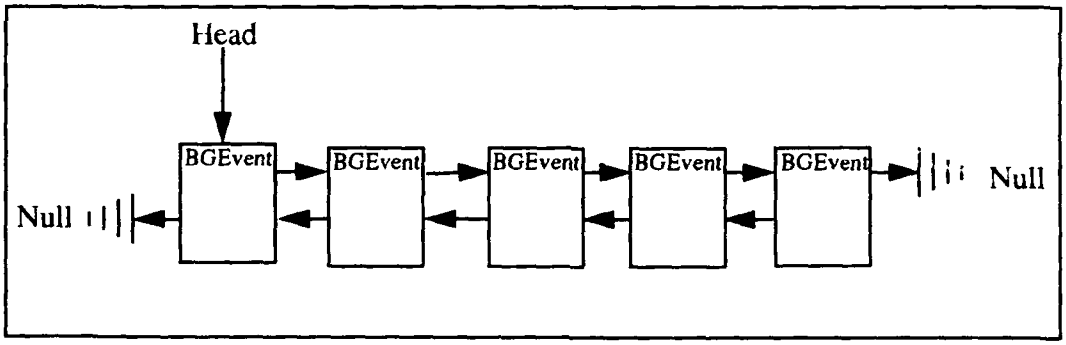

Figure 30: Battle Group Event List

The decision to use such a data structure was based on the intended use of battle group event information. Since the purpose of BGLCSS is to dynamically simulate events in time, the easy insertion, sorting, and deletion of events is the dominant criteria for selecting the appropriate data structure. A doubly linked list fulfills this criteria. It allows easy movement forward and backward between list element structures, sorted insertions and deletions are easily performed by manipulating pointers. The pointers on the ends of the list both point to null.

There are three different doubly linked list data structures used to implement the battle group events: the battle group event list itself, the related event list, and the header list. The battle group event list is represented by battle group event information structs in a doubly linked list. The code used to define a BGEvent is shown in Figure 31 .

```
struct BGEvent {
        struct BGEvent      *Prev,
                            *Next;
        int                  Dtg,
                             CreateTime,
                             PredictInterval,
        BGEventType          EType;
        PredictType          PType;
        CalcType             CType;
        ThreatType           TType;
        UnrepInfo            Unrep;
        DirectionInfo        Direction;
        StrikeInfo           Strike;
        RaidInfo             Raid;
        ASWInfo              ASW;
 };
typedef struct BGEvent BGEVENT;
```

Figure 31: Battle Group Event Type Definition

The first two variables are the pointers to the previous and to the next BGEvent structs. The Dtg variable is the integer representation of date time group for the start of the battle group event. The next integer variable is the system time stamp of the event's creation time. This is a unique time stamp for each event in the event list. The next four variables are used when evaluating each event in the list. EType is used to identify the type of event, i.e. battle group course and speed, AAW threat level, etc. PType is used to identify whether the BGEvent PredictType is an *orphan, child, parent,* or *interval*. The first three values will be discussed in the next section. The *interval* value refers to a BGEvent which is actually a sentinel marking the time interval for calculation purposes. Unrep is a struct containing data relevant to an underway replenishment event. DirectionInfo contains data about the course and speed of the ship involved. The last three structs contain information specific to the strike, raid, and ASW events.

One of the most significant omissions in BGLCSS 1.0 was the way that an underway replenishment event and its associated stationing events were handled in the event list. When a user added an underway replenishment event, depending on the tactic used, several stationing events were also added to the list. If a user wanted to delete the underway replenishment, it was necessary for the user to know which stationing events to delete as well. This problem was corrected by using an additional doubly linked list for related events as shown in Figure 32.

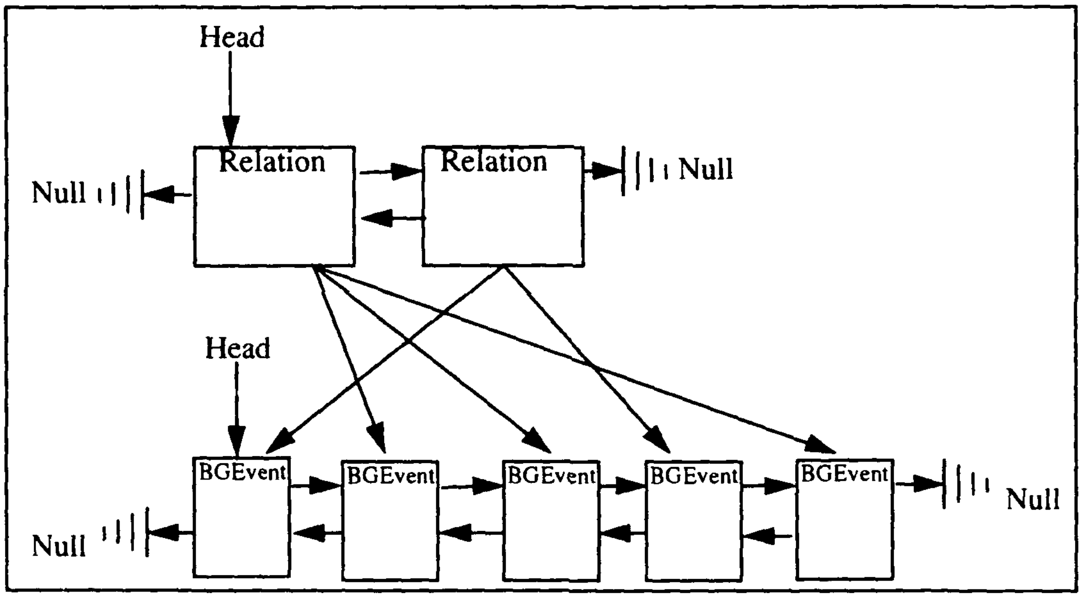

Figure 32: Battle Group Related Event and Battle Group Event List

When an underway replenishment or a consol event is created, a relation event, shown in Figure 33, is also created and inserted into a doubly linked list. The relation struct is created based on the system creation time stamp of the underway replenishment event, also known as a *parent* event. Each event associated with the underway replenishment parent event also contains the same creation time. The relation event pointers point to all of the *child* events, i.e. stationing events, in the BGEvent list. When the deletion of a underway replenishment is to be performed, the related relation event is found and all of the BGEvents to which it points are deleted along with itself. An *orphan* is an event that is not related to any other event, such as a battle group course and speed or a ship stationing event.

```
struct RelationType {
    struct RelationType        *Prev,
                               *Next;
      int                       CreateTime;
      BGEVENT                  *Child1,
                               *Child2,
                               *Child3,
                               *Child4,
                               *Child5;
 };
 typedef struct RelationType RELATION;
```

Figure 33: Relation Type Definition

The last doubly linked list contains the components of a string to represent an event to the user in English (see Figure 34).

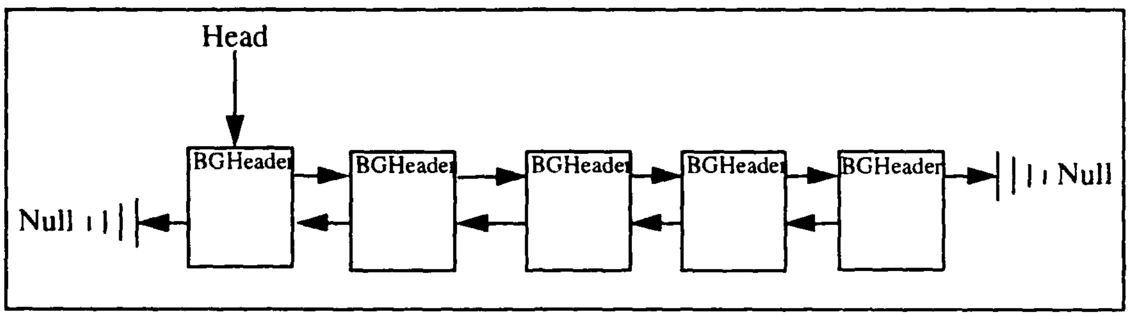

Figure 34: Battle Group Header List

It is used as a convenient way to show the string representation of the event list. For example, the event list is displayed to the user as shown in Figure 35. This list is made up of structs containing the string equivalent of the BGEvent list. It is used instead of traversing the BGEvent list and generating string equivalents for each event each time that the event list would be displayed on a panel.

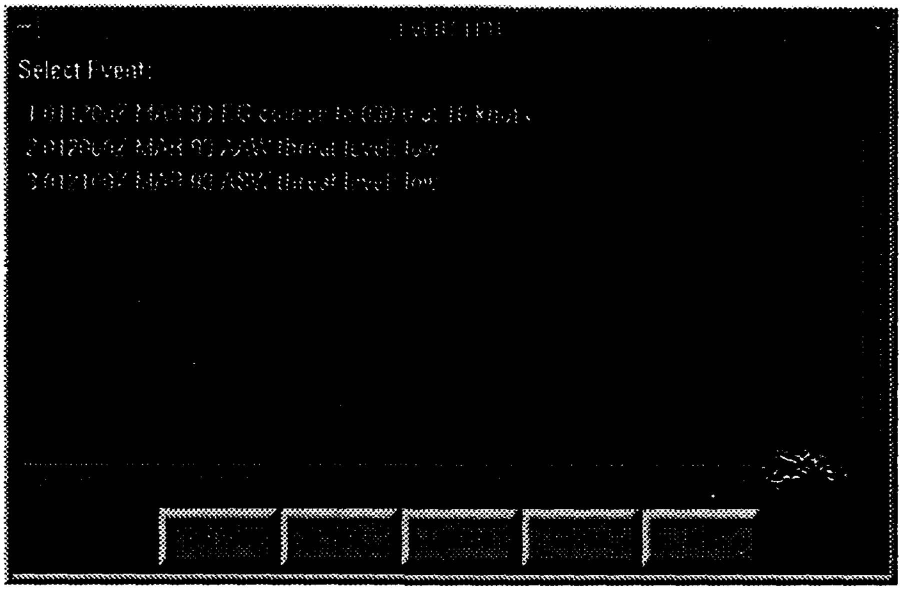

Figure 35: BGLCSS 2.0 Event List Panel

Figure 36 shows the BGHeader Type definition for the header structs in the header list.

```
struct BGHeader{
        struct BGHeader  *Prev,
                         *Next;
        BGEventType      EType;
        int              Index;
        int              DTG;
        char             Date[DTGLENGTH];
        char             Title[MAXLENGTH];
        float            Course;
        float            Speed;
};

typedef struct BGHeader BGHEADER;
```

Figure 36: Battle Group Header Type Definition

E. BGLCSS 2.0 LIBRARY DESIGN

The application library design refers to the functions written to operate on the data structures in the BGLCSS application. There are three groups of functions based on the kind of data involved: battle groups, ships, and events.

Table 3 shows the function name and purpose for each function that operates on battle groups. These functions are contained in bg.c in Appendix B.

Table 3: BATTLE GROUP FUNCTIONS

Function	Purpose
CancelBG	Clears out data from the text keyin items on BG panels BGData and New BG panels.
DeleteBG	Deletes a battle group from the array by replacing its name with blank spaces.
GetBG	Using the string representation of the name of a battle group and the index to the appropriate battle group in the array, return the index to the battle group.
GetBGs	Given the battle group array, this function gets the battle group data from the battle group data ASCII text file. Returns the next available battle group index. Given the battle group array, this function gets the battle group data
GetShip	Using the string of a ship name and the index to the appropriate battle group, this function returns the ship index for the appropriate ship.
MakeBG	Makes a new battle group, using information provided by user to New BG panel. The index for the next available battle group in the array is returned.
SaveBGs	Saves all battle groups to the ASCII text file containing BG data.
SaveNewBG	Saves the new battle group using the TAE Vm target object from the user input panel. First, GetBGs is called, returning the available index to the array. Next, Make BG is called and then SaveBGS saves all battle groups.
ShowBG	Shows the battle group's data given battle group array, index and panel name.
ShowBGs	Shows the list of battle groups from an ASCII text file to an item in a panel.

Table 4 shows the function name and purpose for each function that operates on ships. These functions are also contained in bg.c in Appendix B.

Table 4: SHIP FUNCTIONS

Function	Purpose
AddShip	This function adds a ship and its data to a battle group. The ship list presented to the user contains both the hull number and the ship name. The hull number is required to get the ship type for the appropriate F76 information. The ship name is returned.
ConvertTypeCombatant	Converts an integer to the appropriate string representation of TypeCombatant. C stores the value of an enumerated type on an ASCII file as an integer. To display this value in a panel, it must be converted to a string.
GetF76Table	Gets the F76 information by ship type from the ASCII text file into memory.
GetShip	Using the string of a ship name and the index to the appropriate battle group, GetShip returns the ship index for the appropriate ship.
SaveBGShips	Using the BG array and the index to the specific battle group, this function saves the battle group's ships' data to an ASCII text file.
ShowBGShips	Shows the list of battle group ships, given battle group index, panel name, and selection list item on panel.
ShowF76	Uses TAE Wpt and Vm functions to display values to the Ship panel.
ShowNavyShips	Shows the list of navy ships from ASCII text file to item in panel.
ShowShip	Uses TAE Wpt and Vm functions to display values to the Ship panel.
TypeShip	Checks the first two characters in a ship's hull number and returns an integer that equates to an enumerated ship type.

Table 5 shows the function name and purpose for each function that operates on the three doubly linked lists involved with BGLCSS events. These functions are contained in `BGEventsLib.c` in Appendix B.

Table 5: BATTLE GROUP EVENT FUNCTIONS

Function	Purpose
DeleteBGEvent	Given a pointer to the head of the battle group event list and the event to be deleted, this function deletes the event. Before calling this function with the Parent Event node pointer, need to call the DeleteChildren function to delete the associated children.
DeleteChildren	This function makes repeated calls to DeleteBGEvent in order to delete all of the children of the Parent event. Returns the head of the battle group event list.
GetBGEvents	This function reads the battle group event list data from the appropriate ASCII text file given the index to the battle group array. It returns a pointer to the head of the battle group event list.
GetParent	This functions finds the Parent event with its unique time stamp. If the parent doesn't exist, then it finds the orphan event and returns a pointer to the event found.
GetRelation	This functions finds the Parent event with its unique time stamp. If the parent doesn't exist, then it finds the orphan event and returns a pointer to the event found.
InsertBGEvent	This function takes a pointer to the head of the battle group event list and a pointer to the newly created battle group event and inserts the new event into the list based on chronological date time group of the events. Returns a pointer to the head of the battle group event list.
InsertRelation	This function's basic algorithm is virtually the same to InsertBGEvent except for the final if-statement assignments and the data type involved.
MakeBGEvent	Given the information from an event panel, this function makes a battle event node and returns a pointer to it. This function is currently designed to handle only a battle group course and speed change event.
MakeChild	This function makes a child event by first calling MakeBGEvent and attaching the child to the appropriate relation node. After a call to this function is made, need to call, for instance, UnrepCalculations and make the appropriate assignments to the event node. Function returns a pointer to the newly made child.
MakeRelation	This functions makes a related-event node used to connect related events together such as an unrep with its associated stationing events. The parameter passed is the integer value of the creation time for the parent event (such as the unrep event). No more than 5 associated events are allowed by this function. Returns a pointer to the newly created relation node.

Table 5: BATTLE GROUP EVENT FUNCTIONS

Function	Purpose
MakeUnrep	Given a pointer to the newly made event node and the values passed from the Unrep panel, return a completed unrep event node to be added to the event list.
SaveBGEvents	This function saves the battle group events list given an index to the appropriate battle group in the array and a pointer to the head of the battle group event list. It returns the pointer to the head of the list.
MakeBGHeader	This function creates the header to be displayed in the event list panel to the user. Given the event parameters, return a header node.
GetBGHeaders	Given the appropriate index to the battle group array, this function gets the battle group header information from the appropriate ASCII text file and returns a pointer to the head of the battle group header list. Similar in algorithm to GetBGEvents.
InsertBGHeader	This function inserts the newly created BGHeader into the Header list given a pointer to the head of the header list and a pointer to the newly created BGHeader. It returns a pointer to the head of the header list.
SaveBGHeaders	Given an index to the battle group array and a pointer to the head of the battle group header list, this function saves the header list data to the appropriate ASCII text file. Returns a pointer to the head of the header list.

F. PROGRAM INTEGRATION

There are two ways to connect a call to an application library function to the interface code. If there is only one line of code, i.e. a call to a single function, this line of code can be inserted at the WorkBench level using the Connection Specification Panel shown in the previous chapter. This is the best method, because if an item on a panel is changed from an event-generating item to a non-event-generating item, the source code is overwritten. The original file is copied to a backup file with a `.bak` suffix. However, if several calls need to be made to handle a given TAE event, then the insertion must be made by hand by the applications programmer. To maximize integration at the WorkBench level, once a sequence of application function calls within a TAE Plus event handler function are debugged, a higher-level application function can be written to make the sequence of function calls itself. Thus, a higher-level function call can be integrated at the WorkBench level.

The TAE Plus documentation suggests that a symbol be used for each alteration to TAE Plus generated source code so that in the event of code regeneration, a relatively easy cut and paste operation can be performed. The /*BERN*/ symbol was used in this application.

TAE Plus uses an `Imakefile` to create an application specific `Makefile` which then correctly compiles and links object files with the appropriate libraries. The `Imakefile` can be edited to include the application library code for compilation. In this case, the APP flags in the `Imakefile` for the BGSetup module to be `bg.c`. See the `Imakefile` in Appendix B.

Each panel file must include the pan_name.h files for associated panels as well as connected panels. Connected panels are already handled by the TAE code generator. Associations that fall outside of the WorkBench domain must be done by hand. A panel may be associated with another if, for instance, data from one panel is required to perform operations initiated by another panel. Although TAE automatically will insert a #include `pan_*.h` for connected panels, the application writer must still explicitly include header files for panels requiring information from a previous panel.

Any application functions used in the `pan_*.c` files must be declared as external ahead of the event handling function definition for the particular panel.

G. STRUCTURED DESIGN PROBLEMS

Despite the top-down functional decomposition approach to this application, there are several problems with this design. The separation of the code and the data is evident in the separation of the struct and type definitions in `bg.h` and `BGEventsLib.h` and the function libraries in `bg.c` and `BGEventsLib.c`. At any point in the main program, a variable such as a BGInfo array could be declared and filled with bad data and then saved to the ASCII text file. This vulnerability of the battle group data is a clear weakness of this design.

Second, while this application does define numerous user-defined types such as the enumerated types and structs, these definitions amount only to labels that improve program clarity. They do not prevent illogical operations from being performed such as assigning bad values to the enumerated types. Furthermore, as previously mentioned, date time groups and latitudes and longitudes are reduced to built-in data types and poorly represent real world entities. This data type deficiency led to the GOTS library's long list of specialized functions as well as to the redundant BGLCSS library functions such as the InsertBGEvent, InsertBGHeader, etc.

Third, the distributed functionality of this design will make subsequent program modifications and extensions difficult. For instance, to add another logistics event type would require changes to be made to every event-related function that contained a case statement specific to event type.

V. BGLCSS 2.0 OBJECT-ORIENTED DESIGN

A. CLASSES

This chapter provides a high-level view of how BGLCSS 2.0 could be designed using an object-oriented paradigm in C++. We intend to provide examples of how the use of an object-oriented paradigm, when correctly applied, leads to code reuse, ease of program maintenance and extension. Since the arguments presented here are not dependent on low-level definition details, most class member functions are presented only as prototypes.

Object-oriented design is based on classes and one useful approach to object-oriented design consists of the following methodology [COPLIEN 92]:

(1) Identify the entities in the application domain.

(2) Identify the behaviors of the entities.

(3) Identify the relationships between entities.

(4) Create a C++ design structure from the entities.

According to step (1), the entities in the application domain are battle groups, ships, and logistics events. Step (2), the identification of the behaviors of the entities is listed in Table 6. The behaviors of the battle group entity are limited to getting, setting, adding, and deleting subcomponents. The ship entity, in addition to these behaviors, consumes and fills up with F-76, F-44, and ordnance commodities. Events, shown as a high-level abstraction of all twelve events, includes a computation behavior called ProcessEvents which performs calculations of the F-76, F-44, and ordnance states of ships in a battle group.

Table 6: SAMPLE BEHAVIORS OF BGLCSS ENTITIES

Battle Groups	Ships	Events
GetBGName	GetShipName	GetEventList
GetBGIndex	GetShipIndex	SaveEventList
GetBG	GetShipData	AddEvent
SaveBG	SetShipData	DeleteEvent
GetBGShips	GetShipLocation	GetEvent
SaveBGShips	SetShipLocation	ProcessEvents
AddBGShip	GetShipF76	
DeleteBGShip	SetShipF76	
GetBGLocation	GetShipF44	
SetBGLocation	SetShipF44	
GetBGResults	GetShipOrdnance	
SetBGResults	SetShipOrdnance	
	GetShipAcft	
	SetShipAcft	
	ConsumeF76	
	ConsumeF44	
	ConsumeOrdnance	
	FillF76	
	FillF44	
	FillOrdnance	

Step (3), identifying the relationships between entities, is probably the most crucial step of the design process. First, the battle group exhibits a *has a* relationship with its ships, settings, location, and results. This relationship can be modeled with composition.Therefore, an array of ShipInfo class objects, a SettingsInfo class object, a Location Info class object, and an array of CapacityInfo class objects can be contained within a BGInfo class object. Second, the ShipInfo class can be modeled based on an *is a* relationship with all of the special cases of a ship: destroyer, frigate, aircraft carrier, etc. Finally, events, at this level of abstraction, involve a *uses a* relationship because, when the ProcessEvents behavior is performed, it requires the use of ShipInfo objects. At the level of specific events as in the case of Unrep and Consol events, both of these events involve a *creates a* relationship with SetStation events.

Step (4), a C++ design structure for each of these entities, is presented in detail.

1. Battle Group Class

A real world Navy battle group is characterized by its name and designation, a set of ships, and, for the purposes of the BGLCSS simulation program, a collection of trip-wire settings. The decision to design the battle group class using *composition* is based on the *has a* relationship that describes the real world battle group entity. The battle group *has a* set of settings, a location, a set of ships and a set of calculation results. With composition, all of the data and functions of the first class are reused in the second class. For example, all of the LocationInfo class data and function members are used in the BGInfo class as well as reused again in the ShipInfo class discussed later. This reuse is similar to the nesting of the LocationInfo struct within the BGInfo and ShipInfo structs in the structured design described in chapter IV except that the functions, as well as the data, are included.

While the structure of the BGInfo class is similar to its corresponding structured paradigm struct, its data members can be hidden from outside the class. By declaring them to be private, they can only be accessed by member functions of the same class. The array of ShipInfo objects and the SettingsInfo, LocationInfo, and ResultsInfo C++ classes compose the BGInfo class as shown in Figure 37.

```
class BGInfo {
   private:
        char                  Name[MAXNAME];
        char                  Designation[MAXNAME];
        ShipInfo              Ships[MAXBGSHIPS];
        SettingsInfo          Settings;
        LocationInfo          Location;
        CapacityInfo          Results[MAXBGSHIPS];

   public:
        BGInfo();
        BGInfo(char* N, char* D, SettingsInfo* S)
        ~BGInfo();

        void                  SaveBGShips(ShipInfo S);
        friend BGInfo&        CalculateF76(BGEvent&, int i);
        void                  ShowBGShips(int i);
        void                  ShowBG(int t);
        int                   SaveNewBG(BGInfo B);
        void                  CancelBG(Id I);
        int                   GetBG(int i);
        int                   AddBGShip(int i, ShipInfo* S );
        int                   DeleteBGShip(int i, ShipInfo* S);
        LocationInfo*         GetBGLocation(int i);
        int                   SetBGLocation(LocationInfo* L);
        CapacityInfo          GetBGResults(int i);
        int                   SetBGResults(CapacityInfo C);
};
```

Figure 37: Battle Group Data and Function Members

The choice to keep the array as the data structure to contain the ships and battle groups was made for simplicity. The array of battle groups, i.e. the array of BGInfo objects is shown in Figure 38.

```
class BGArray {
   private:
        BGInfo                BG[MAXBGS];
   public:
        BGArray();
        ~BGArray();
        void                  SaveBGs(BGInfo B);
        void                  ShowBGs(BGInfo B);
        BGInfo                GetBGs();
};
```

Figure 38: Battle Group Array Data and Function Members

2. Ship Class Hierarchy

Contained within the battle group is the array of ships. The object-oriented ship class is shown in Figure 39. It is a classic example of a single inheritance class hierarchy. There are five main classes derived from the ship base class: Destroyer, Frigate, Cruiser, Aircraft Carrier, and CLF Ship. CLF Ship is further divided into Ammunition ship, Fleet Oiler and Combat Support Ship. Finally, at the ends of the ship class tree are the specific ship types such as DD963, FFG7, etc. This hierarchy is based on the *Navy's notion of ship class* which is based on ship architecture and ship mission. BGLCSS, however, is driven by the ship type differences in F-76 and F-44 fuel, and ordnance attributes. For instance, the F-76 fuel capacity and F-76 ship fuel burn rate is dependent on the Navy ship class such as the DD963 destroyer class. As far as BGLCSS 2.0 is concerned, this makes all of the calculations of commodity use far simpler to modify and extend.

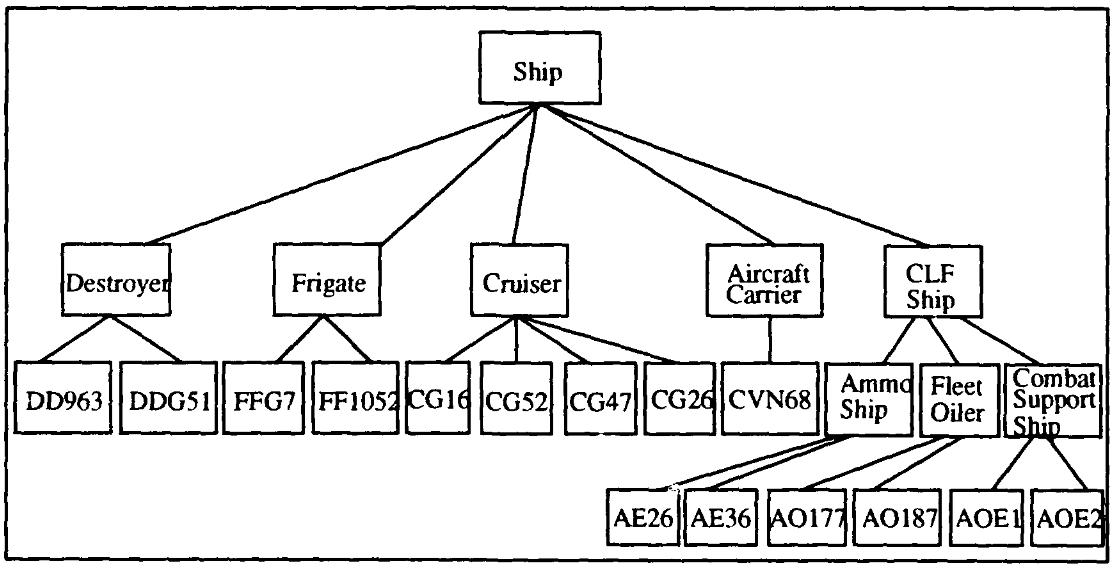

Figure 39: Ship Class Hierarchy

The base class, ShipInfo class, as shown in Figure 40, contains protected data members and public member functions, some of which have been declared virtual. Base class access determines how the derived class receives inherited members.

```
class ShipInfo {
   protected:
        enum CLFType {Air, Combatant, Station,Shuttle};
        enum ShipType{Destroyer, Frigate,Cruiser, AcftCarrier,
        CLFShip};
        char                 Name[40];
        char                 Hull[10];
        CLFType              TypeCombatant;
        LocationInfo         Location;
        F76Info              F76;
        F44Info              F44;
        int                  Approach,
                             BreakAway;
        OrdInfo              Ord;
        AcftInfo             Acft;
   public:
        ShipInfo(); //constructor
        virtual              ~ShipInfo();//destructor
        int                  GetShip(int t, char* c);
        virtual int          GetShipType(ShipInfo s, int i)·
        virtual int          SetShipType(ShipInfo s, int i);
        virtual int          SetCLFType(ShipInfo s, int i);
        virtual int          GetCLFType(ShipInfo s, int i);
        virtual int          SetF76Info(ShipInfo s, int i);
        virtual int          GetF76Info(ShipInfo s, int i);
        char*                GetShipName(int i);
};
class DestroyerInfo : public ShipInfo {
   protected:
        enum DestroyerType{DD963, DD51};

        DestroyerType        DType;

   public:
        int                  GetShipType(DestroyerInfo s, int i);
        int                  SetShipType(DestroyerInfo s, int i);
        int                  SetCLFType(DestroyerInfo s, int i);
        int                  GetCLFType(DestroyerInfo s, int i);

        };
class DD962Info : public DestroyerInfo {
   public:
        int                  GetF76Info(DD963Info d, int i);
        int                  SetF76Info(DD963Info d, int i);
                             //other commodities are similar
        int                  ConsumeF76(int i, F76Info f);
        int                  FillF76(int i, F76Info f);
                             //other commodities are similar
};
```

Figure 40: Ship Class Data and Function Members

Class access is public for the DestroyerInfo class. This means that the base class's protected members remain protected (inheritable, but still hidden from the rest of the program) and the public members remain public. The same is true for the class access of the DD963Info class. DestroyerInfo class access is public for the DD963Info class. The member functions in this last class are the appropriate place for the DD962 specific commodity values to be initialized. These member functions were made virtual in the ShipInfo base class so that they could be tailored for each bottom-level ship class such as DD963. The ship-specific enumerated types were encapsulated within the specific class that they are relevant. By contrast, in the structured programming design, the enumerated types are global to the program.

3. Logistics Events Class Hierarchy

The logistics event classes are the most challenging to design of the BGLCSS classes. When the entities were identified in the BGLCSS application domain, the event list, events in general, and the twelve specific types of logistics events were discussed in general terms. At this point in the analysis, it is possible to make several class designs. We will discuss two specific designs, the first of which is shown in Figure 41.

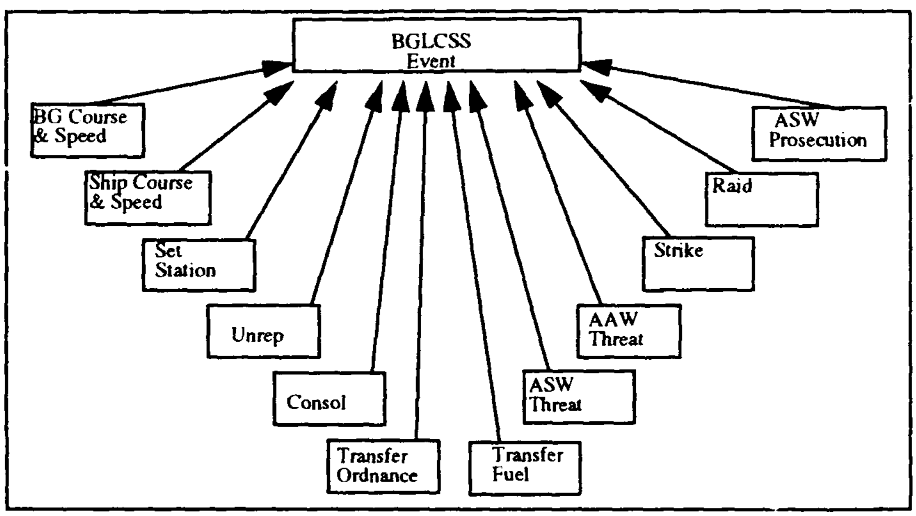

Figure 41: BGLCSS Event Class Hierarchy With Twelve Derived Classes

First, the twelve BGLCSS logistics events could placed in a class hierarchy where there is one abstract base class from which all twelve logistics event classes are derived.

Although an end user might visualize the twelve BGLCSS events in this way, a class hierarchy of this design does little to support code reuse. This is because, when describing the entity behaviors or functions in detail as outlined in step (2) of the object-oriented programming paradigm methodology, the patterns shown in Table 9 become clear.

Table 7: BGLCSS EVENT COMMON FUNCTIONS

	Affects F-76 Level	Affects F-44 Level	Affects Ordnance Level
BGCourseSpeed	Yes		
SetShipStation	Yes		
ShipCourseSpeed	Yes		
FuelTransfer	Yes	Yes	
AAWThreatLevel		Yes	
Strike		Yes	
Unrep	Yes	Yes	Yes
Consol	Yes	Yes	Yes
ASWThreatLevel		Yes	Yes
Raid		Yes	Yes
OrdnanceTransfer			Yes
ASW Prosecution			Yes

Each of these events affects a different combination of the F-76, F-44, and ordnance commodities and would require duplicated code regarding the calculations of commodity levels.

At this point, it is useful to draw a distinction between the *problem domain* and the *program domain*. The *problem domain* refers to the real world problem that the software is intended to solve. In contrast to the structured programming paradigm, the object-oriented programming paradigm focuses on closely mapping the entities in the real world, the *problem domain*, to entities in software. The *program domain* differs from the *problem domain* in that it represents the programming language, operating system, and programming paradigm.Writing a program consists of building a solution within the program domain to solve a problem in the problem domain.

The most straightforward approach is to first solve the problem within the problem domain, then construct a model of the problem domain within the program domain and map the solution over. The more explicit the model, the more obvious the mapping and the easier it becomes to write and understand the resulting program. [DAVIS 92]. Since the problem domain of BGLCSS is to generate percent capacity states for F-76, F-44, and ordnance for ships and battle groups, this distinction was used to determine whether clusterings of similar event behaviors was present. Table 7 clearly shows that there are common behaviors/functions among the events.

The solution to designing a class hierarchy that promotes code reuse involve multiple inheritance. Multiple inheritance permits a class to be derived from two or more base classes. With this kind of construction, class relationships become much more involved than with single inheritance. Under single inheritance, the inheritance hierarchy is a tree; under multiple inheritance, the hierarchy is a directed acyclic graph. Cargill makes a distinction between *synthetic* and *natural* classes. *Synthetic* classes do not correspond to abstractions found in the application problem domain. *Synthetic* classes emerge during design and coding of a system in response to internal, synthetic needs of the software. This is in contrast to *natural* classes, those that correspond to abstractions from the problem domain and typically arise either during analysis or early design. A simple criterion is to ask end users if they recognize the abstraction. Because a natural class comes from the problem domain, an end user will understand its purpose; a synthetic class arises only from software implementation considerations, so the end user will not appreciate the need for it [CARGILL 92].

The BGLCSS 2.0 event classes are more complex than the ship classes because, in order to maximize reuse of class member functions, multiple inheritance is necessary. Instead of a tree structure as in the ship classes, the event class is a directed acyclic graph, where the classes lower on the tree inherit from the classes connected above them. This design decision is based on the common denominators of the logistics events, in other words, these events either affect F-76, F-44, or ordnance, or a combination of these. There are no *is a* relationships. The class hierarchy is just a convenience to reuse code, particularly the member functions of the commodity affecting classes. There is no domain specific relationship.

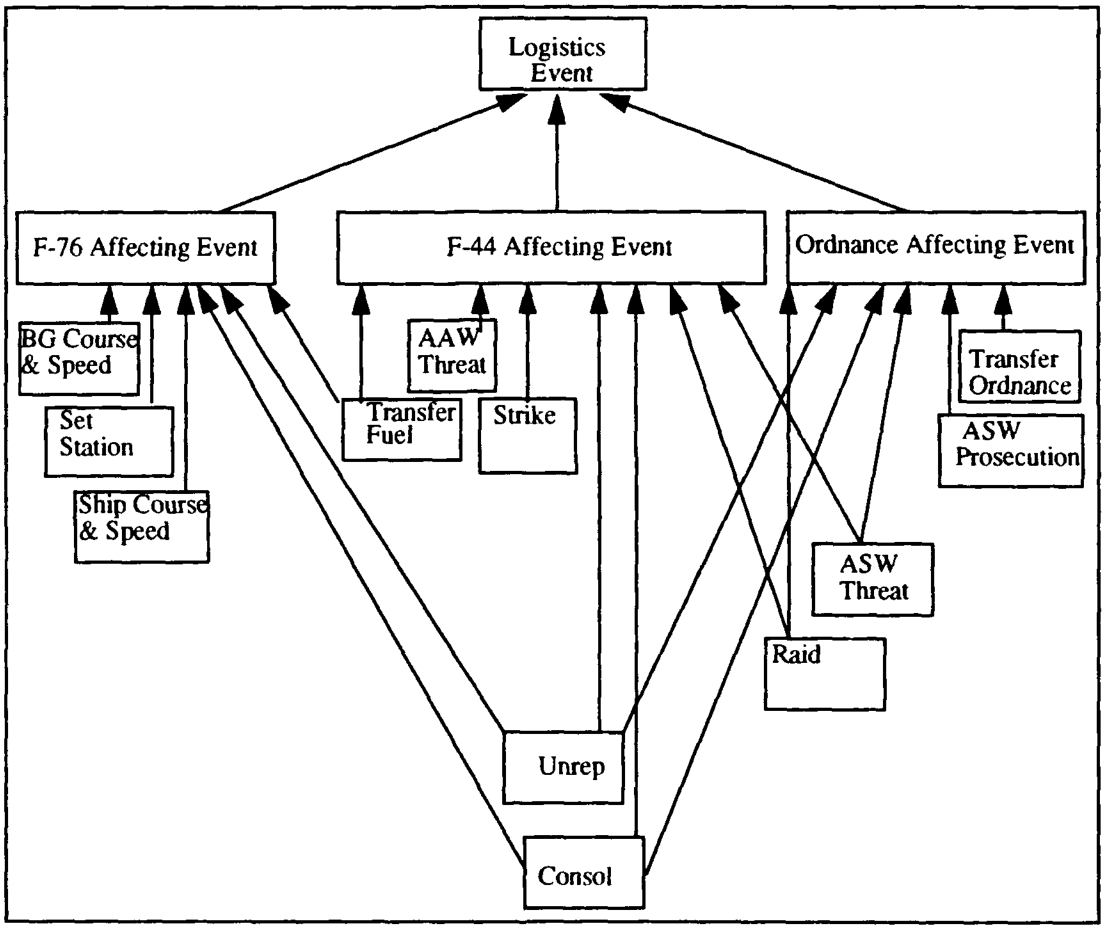

Figure 42: Logistics Event Class Hierarchy With Three Synthetic Derived Classes

The C++ class definition for the logistics event class is provided in Figure 43. The enumerated types specific to the events are encapsulated within the abstract base class BGEvent.

```
class BGEvent {
   protected:
        enum BGEventType {BGCourseSpeed, ASWLevel, AAWLevel,
        SetStation, ShipCourseSpeed, Unrep, Consol,
        FuelTransfer, OrdTransfer, RaidEvent, StrikeEvent,
        ASWProsecute, ResumeBGCourseSpeed, Other};

        enum PredictType {Orphan, Child, Parent,Interval};

        enum ThreatType {Low, Med, High, Raid, Strike, Asw};

        enum CalcType {Ord, F76, F44, BothFuel, All};

        BGEvent             *Prev,
                            *Next;
        int                  DTG,
                             Index,
                             Created,
                             PredictInterval;
        BGEventType          EType;
        PredictType          PType;
        CalcType             CType;
        DirectionInfo        Direction;
   public:
        BGEvent();
        ~BGEvent();
};

class F76Event : public BGEvent {
   public:
        friend BGInfo& BGInfo::CalculateF76(BGEvent&, int i);
};

class Unrep : public F76Event {
   public:

        UnrepInfo          UnrepData;
};
```

Figure 43: BGEvent Class Data and Function

Whereas in the BGInfo struct in the structured programming design described in chapter IV contained the UnrepInfo, ThreatType, StrikeInfo, RaidInfo, and ASWInfo structs, they are omitted from the BGInfo abstract base class. Instead, these objects are contained only in the appropriate bottom level class objects. For instance, the UnrepInfo object would be contained in the Unrep class only. This design is more representative of the real world entities and makes future modifications and extensions easier to perform because all of the underway replenishment data and functions are localized to the one class where this information is relevent.

When a function manipulates objects of two distinct classes, the function can be made a friend function to both classes. This is what is done with the CalculateF76 function which illustrates the uses a relationship between the BGEvent class and the BGInfo class. This function was made a friend to both classes.

As far as code reuse is concerned, with C++ version 2.1, the events classes discussed so far provide only limited code reuse when considering that the BGLCSS application has three doubly linked lists, each performing insertion, deletion, search, etc. Using this event hierarchy as is would involve creating three classes for each of the doubly linked lists. Each of these data structures could only handle specific objects. To capture maximum code reuse for the logistics events and list operations in BGLCSS, a feature supported in C++ 3.0 is needed. Templates provide a solution to this code duplication problem. Template classes model generic objects that provide similar operations for different data types. By using a template class, as shown in Figure 44, a generic double linked list can be instantiated for pre-defined and user-defined types.

```
template <class T>
class List {
  protected:
     struct Node {
           T Data;
           Node* Prev;
           Node* Next;
     };
     Node *Nodeptr;
     Node *Headptr;
  public:
     List();
     ~List();
      virtual InsertNode(T);
      virtual DeleteNode(T& node);
      virtual SearchNode(T& node);
};
```

Figure 44: BGLCSS Template List Class

In BGLCSS, we could instantiate three List classes using the three different objects: BGEvents, BGHeader, and Relation. Then, from the BGEvents abstract base class, the *synthetic* commodity affecting classes would be derived. Finally, the twelve logistics events would be derived from the appropriate set or sets of *synthetic* commodity affecting classes

B. SYMBOLIC CONSTANTS

Instead of using the preprocessor directive **`#define`** to define program constants and string literals, C++ and ANSI C provide **`const`** to reserve storage for data that is read-only as shown in Figure 45.The drawback to using **`#define`** is that is does no type checking. Any value can be given to **`#define`** without regard to proper type checking. The lack of proper type checking is one of C's weaknesses that can pose enormous problems for the programmer when trying to trace bugs in code.

```
const int MAXNAME          = 25;
const int MAXF76COEF       = 3;
const int MAXSHIPTYPES     = 8;
const int MAXORD           = 100;
const int MAXACFT          = 20;
const int MAXTHREATLEVELS = 2;
const int MAXINTERVALS     = 3;
const int MAXENGAGEMENTS   = 2;
const int HOURSINDAY       = 23;
const int MAXUSETYPES      = 5;
const int MAXBGSHIPS       = 30;
const int MAXBGS           = 10;
const int MAXORD           = 100;
const int MAXSHIPS         = 100;
const int DTGLENGTH        = 15;
const int MAXLENGTH        = 25;
const char F76DATA[]       = "/h/bglcss/scripts/data/F76.dat";
const char BGDATA[]        = "/h/bglcss/scripts/data/BGData.dat";
const char BGSHIPS[]       = "/h/bglcss/scripts/data/Ships";
const char NAVYSHIPS[]     = "/h/bglcss/scripts/data/NavyShips.dat";
const char EVENTSDATA[]    = "/h/bglcss/scripts/data/Events";
```

Figure 45: Symbolic Constants for Battle Groups and Ships

C. OBJECT-ORIENTED DESIGN BENEFITS

This description of object-oriented mechanisms when applied to the BGLCSS application domain, provided examples of ease of modification , extension, and code reuse. Ease of modification and extension are the natural by-products of a class structure where data and function are encapsulated. The program is easier to modify because the data and functions are not separate and instead would work together. The clutter of the similarly named functions such as InsertBGEvent, InsertBGHeader, and InsertRelation would be replaced by the use of polynorphism where the respective object is sent a message such as Insert.

The ShipInfo class hierarchy that derived specialized classes for each Navy ship class such as DD-963 provided a better way to perform Navy class-specific commodity information.

Code reuse was to a limited extent accomplished with the use of synthetic classes in the BGEvent class hierarchy. A more substantial degree of code reuse could be achieved by using a list template class for all three of the inked list structures.

VI. CONCLUSION AND RECOMMENDATIONS

The structured and object-oriented designs of the same program, BGLCSS 2.0, were presented and the merits of the application of each paradigm were discussed. It is clear that there are numerous technical benefits to using an object-oriented programming paradigm instead of a structured programming paradigm for systems expected to evolve over time. There is little doubt that Command, Control, and Communications (C^3) applications such as the BGLCSS tactical decision aid will be refined and extended as battle group coordinators use the system and identify additional components to be added or existing ones to be changed. In fact, one thesis currently being developed by an Operations Research Department student at the Naval Postgraduate School involves a modified version of the underway replenishment event within BGLCSS.

The initial drawback to moving to an object-oriented programming paradigm can be characterized as the trade-off between long-term planning and design versus short-term production gains. At the beginning of a move to an object-oriented approach, a substantial amount of time is required to study the paradigm and produce an overall design for the classes and their hierarchies in the application. By contrast, accepting the status quo and remaining within a structured programming paradigm requires no extra effort. In a world of time constraints, decisions are frequently made to quickly produce a software application prototype and delay concern about modification and extension until a later time. While this reasoning has dominated many software development projects, it is recommended that organizations such as the Navy Space and Warfare Command which is in charge of managing large software systems pursue moving towards adopting an object-oriented paradigm in the future. The long-term benefits outweigh the short-term benefits. It is also recommended that NTCS-A applications programmers consider using TAE Plus to build their graphical user interfaces instead of using low-level Motif functions. A high-level tool such as TAE Plus greatly reduced BGLCSS graphical user interface development time.

APPENDIX A. BGLCSS 2.0 GRAPHICAL USER INTERFACE PANELS

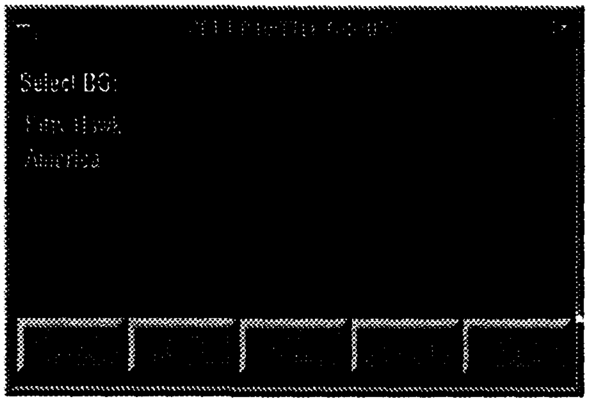

Figure 46: BGLCSS 2.0 Set Up Battle Groups Initial Panel

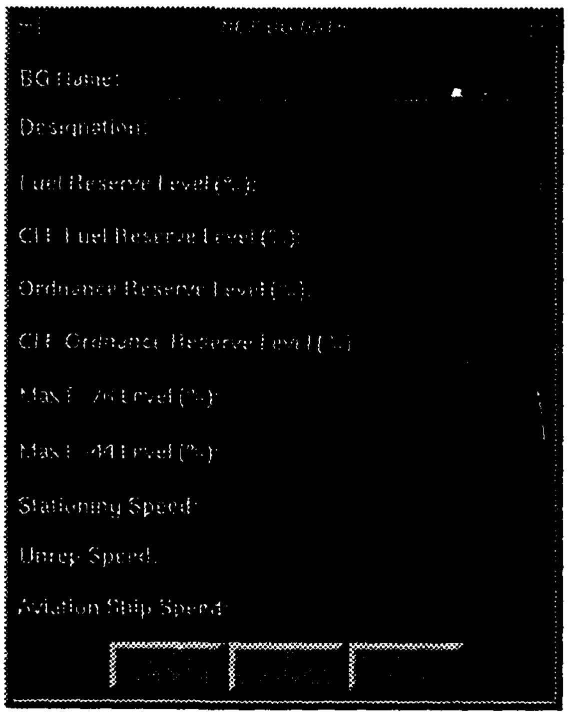

Figure 47: BGLCSS 2.0 New Battle Group Data Panel

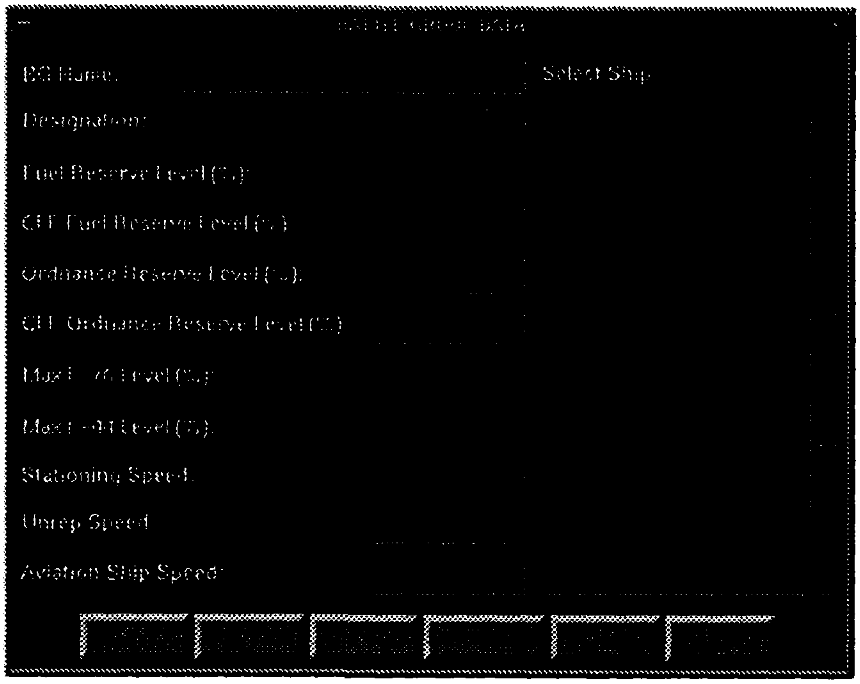

Figure 48: BGLCSS 2.0 Battle Group Data Panel

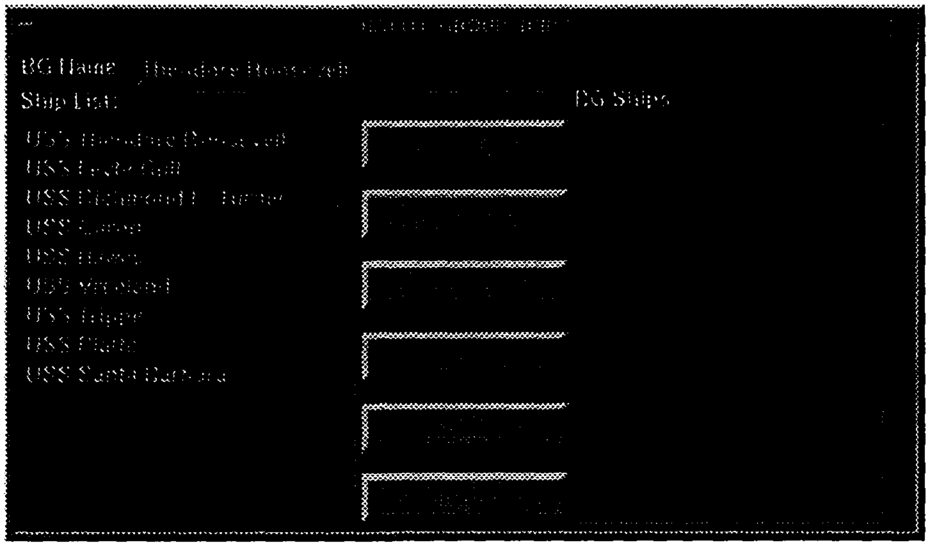

Figure 49: BGLCSS 2.0 Battle Group Ships Panel

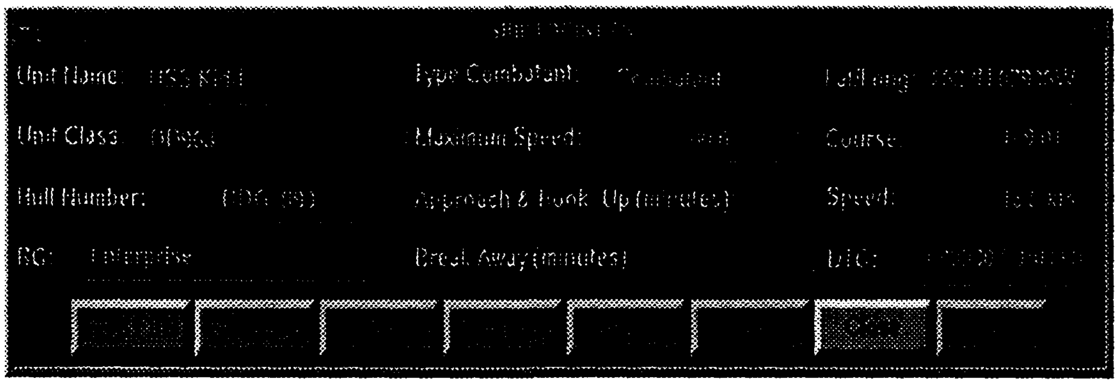

Figure 50: BGLCSS 2.0 Ship Logistics Panel

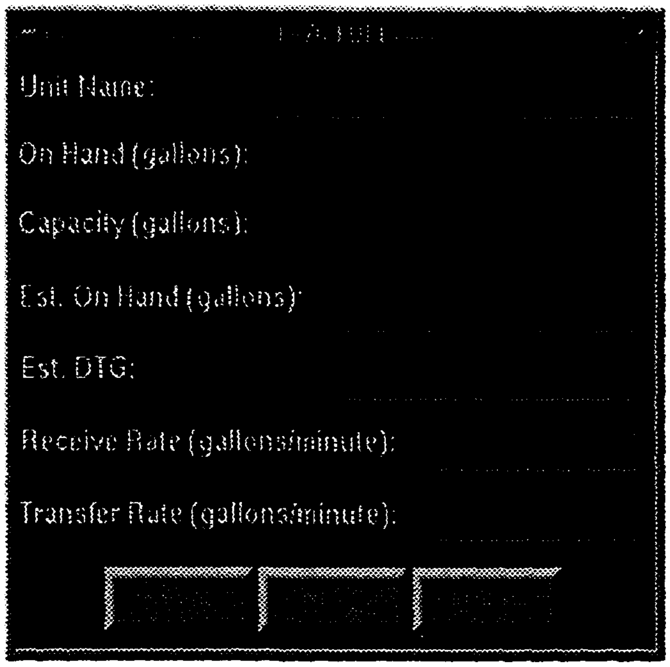

Figure 51: BGLCSS 2.0 Ship F-76 Fuel Panel

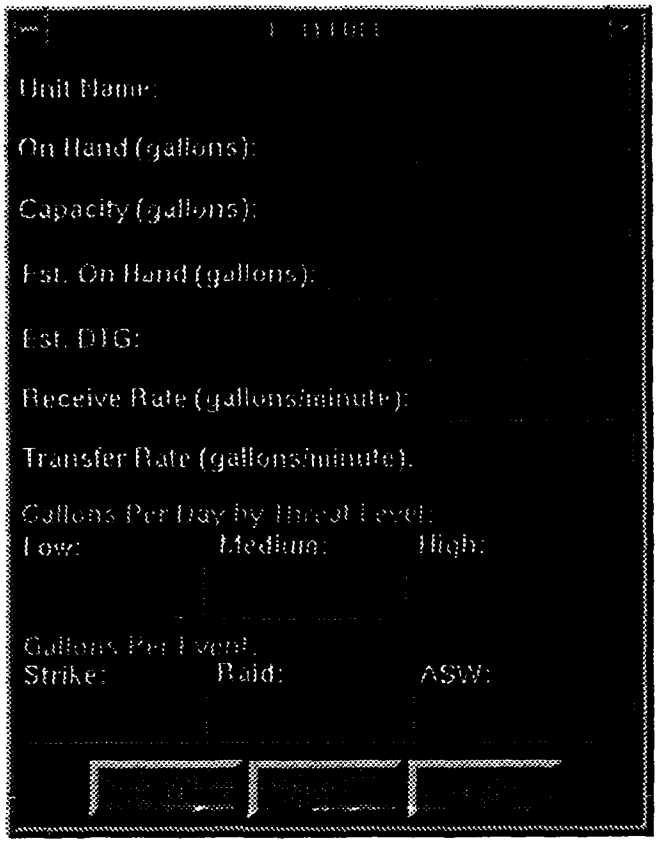

Figure 52: BGLCSS 2.0 Ship F-44 Fuel Panel

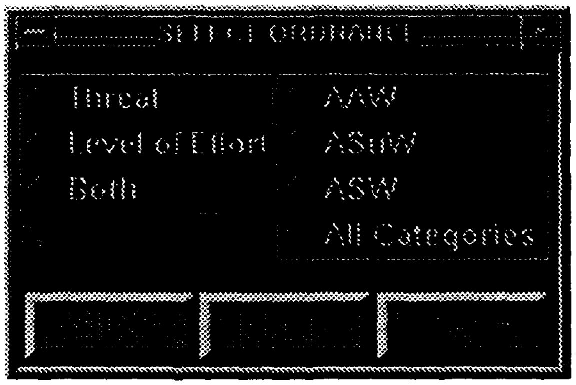

Figure 53: BGLCSS 2.0 Select Ordnance Panel

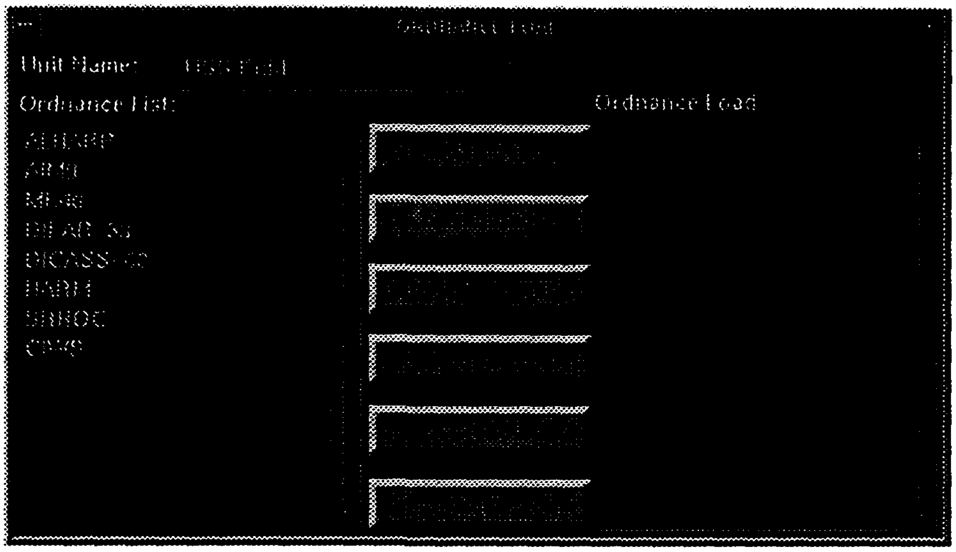

Figure 54: BGLCSS 2.0 Ordnance Load Panel

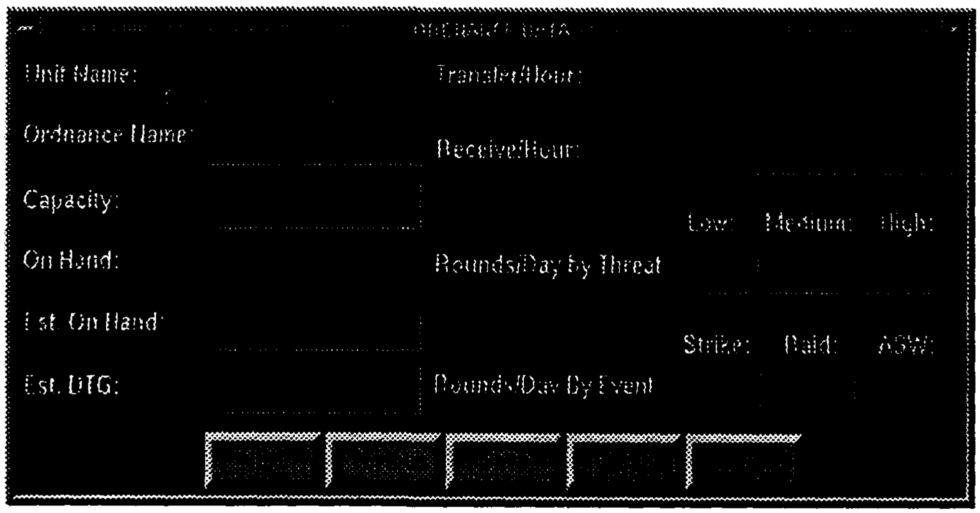

Figure 55: BGLCSS 2.0 Ordnance Data Panel

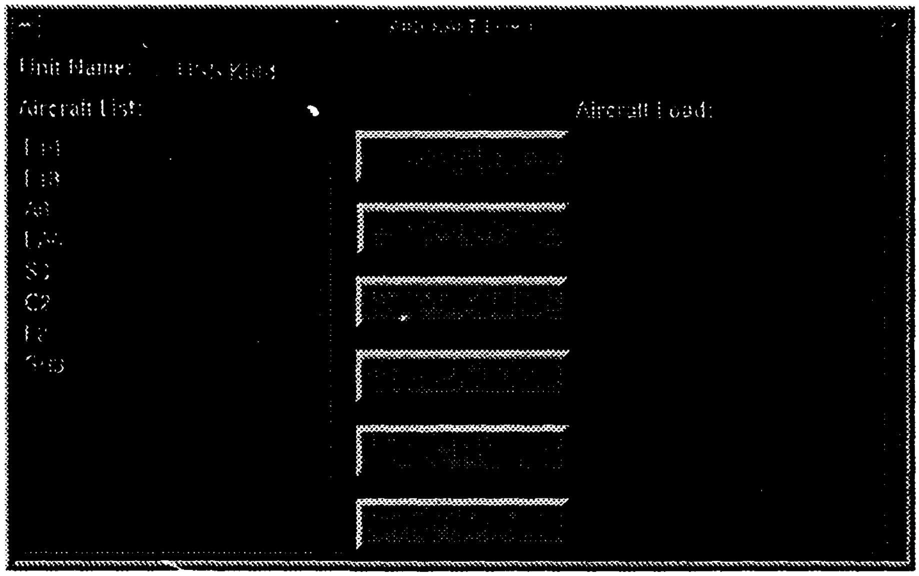

Figure 56: BGLCSS 2.0 Aircraft Load Panel

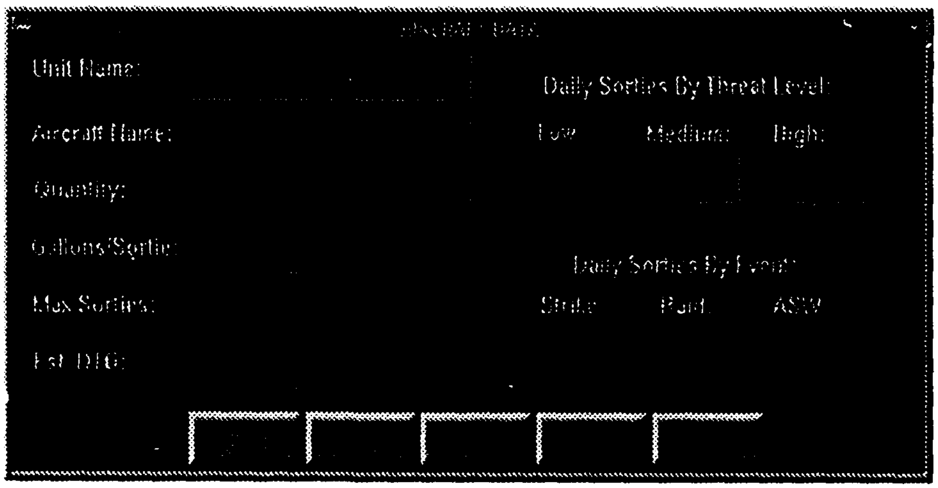

Figure 57: BGLCSS 2.0 Aircraft Data Panel

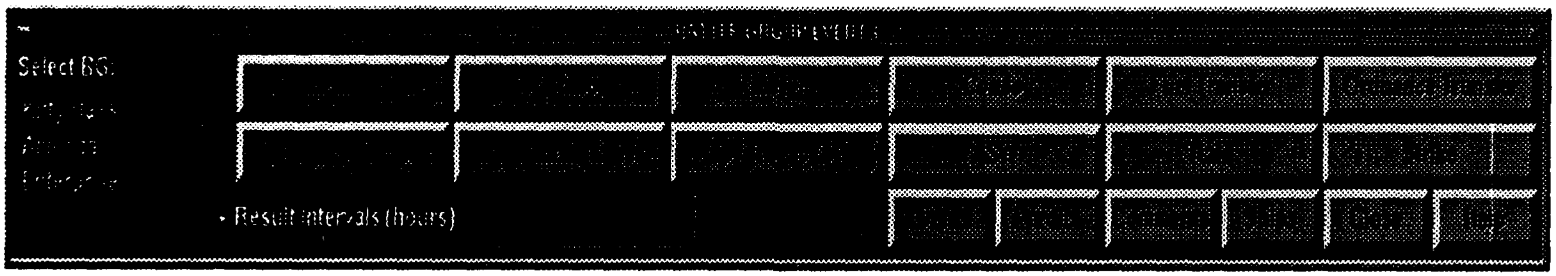

Figure 58: BGLCSS 2.0 Battle Group Events Initial Panel

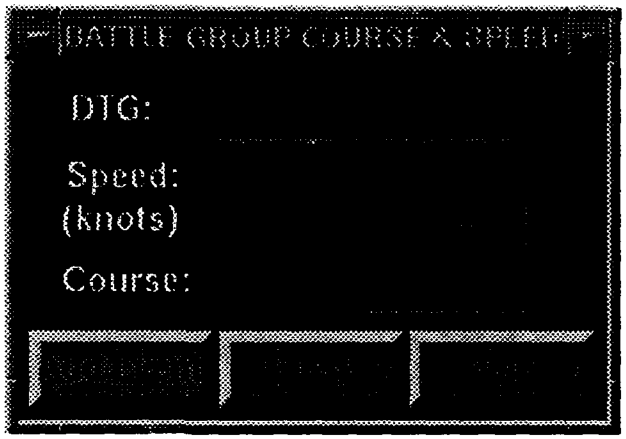

Figure 59: BGLCSS 2.0 Battle Group Course and Speed Panel

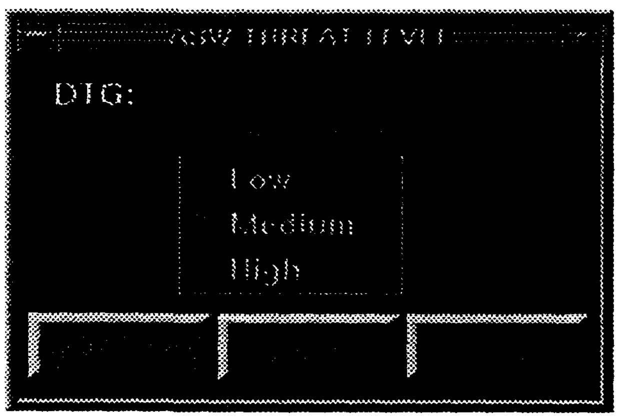

Figure 60: BGLCSS 2.0 ASW Threat Level Panel

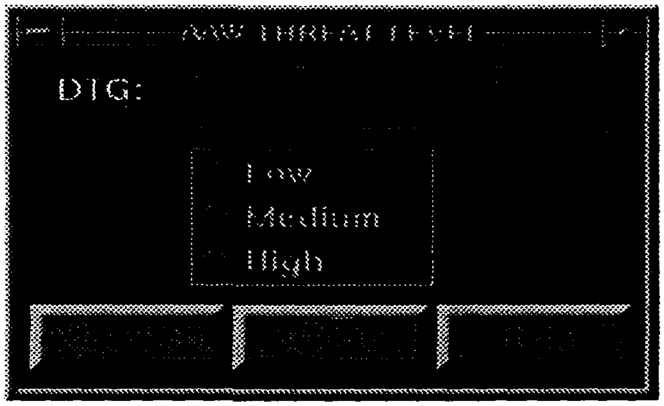

Figure 61: BGLCSS 2.0 AAW Threat Level Panel

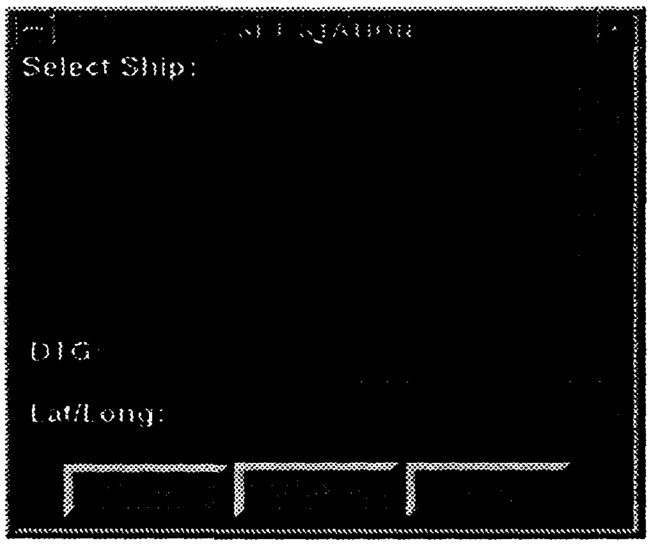

Figure 62: BGLCSS 2.0 Set Station Panel

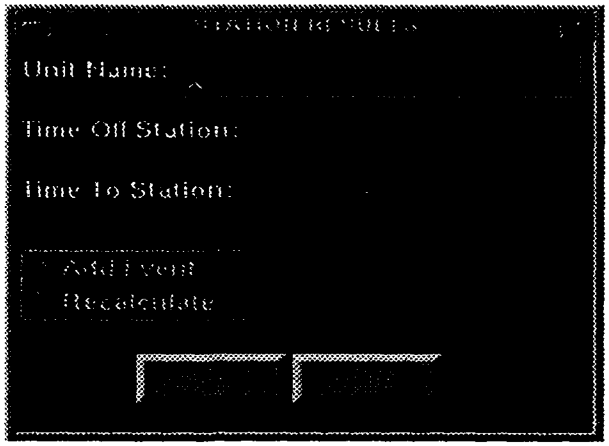

Figure 63: BGLCSS 2.0 Station Results Panel

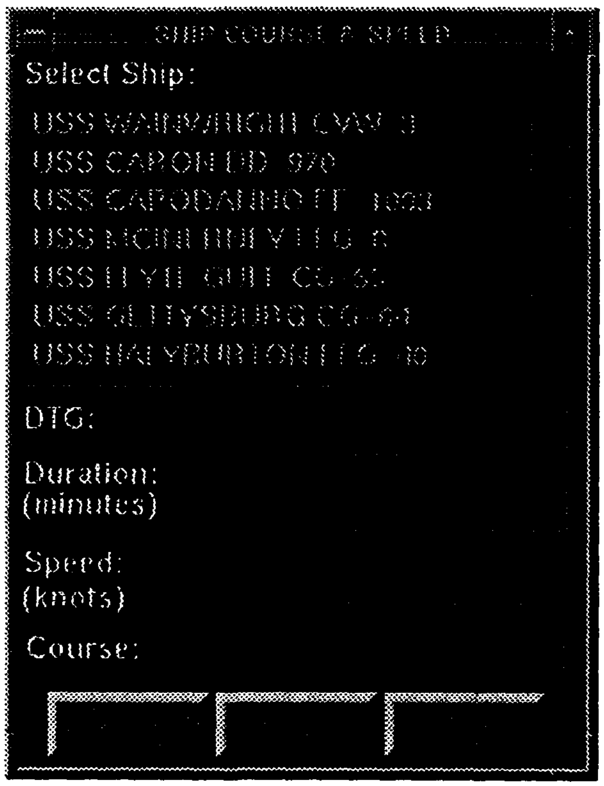

Figure 64: BGLCSS 2.0 Ship Course and Speed Panel

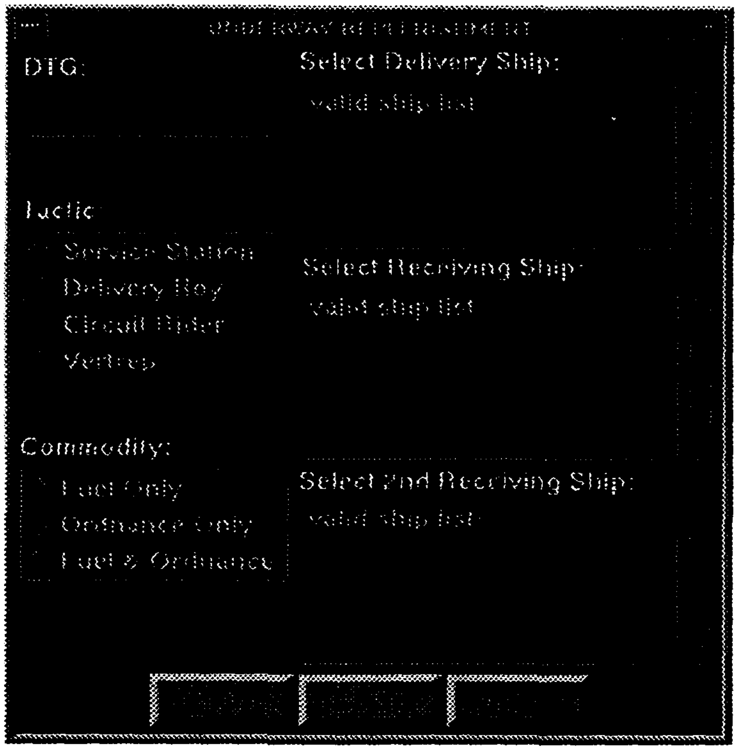

Figure 65: BGLCSS 2.0 Underway Replenishment Panel

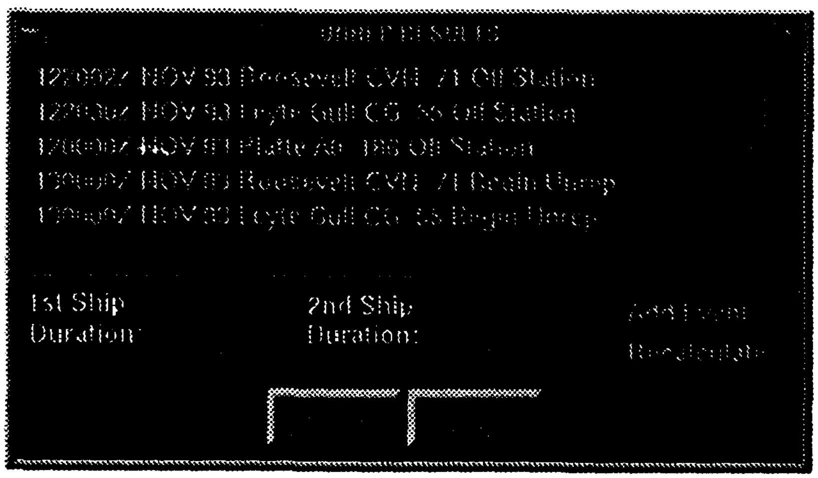

Figure 66: BGLCSS 2.0 Underway Replenishment Results Panel

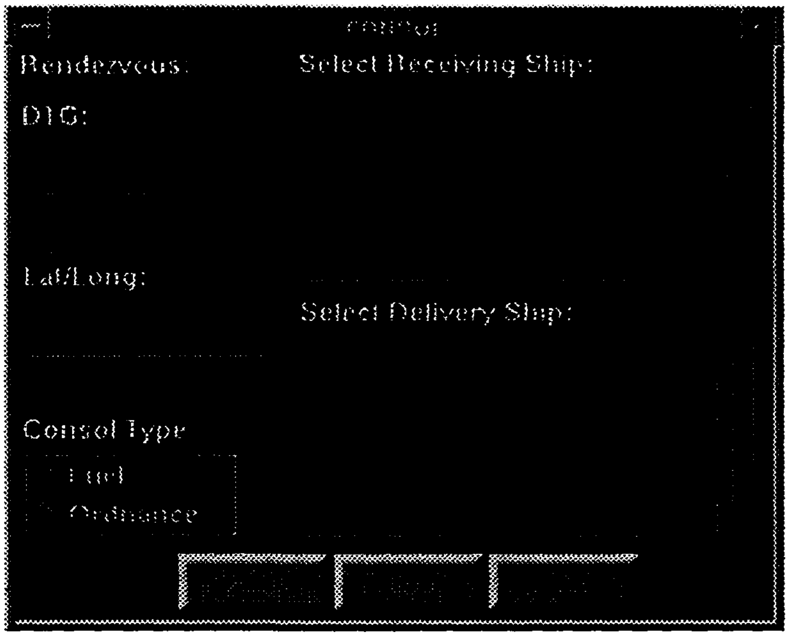

Figure 67: BGLCSS 2.0 Consol Panel

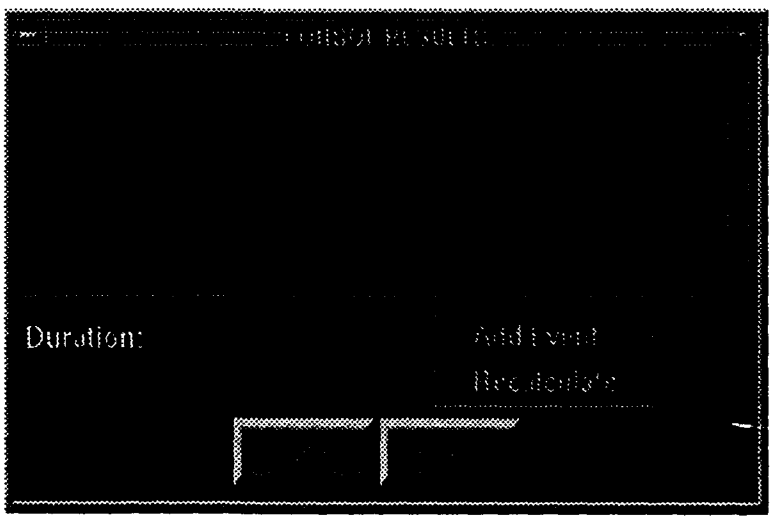

Figure 68: BGLCSS 2.0 Consol Results Panel

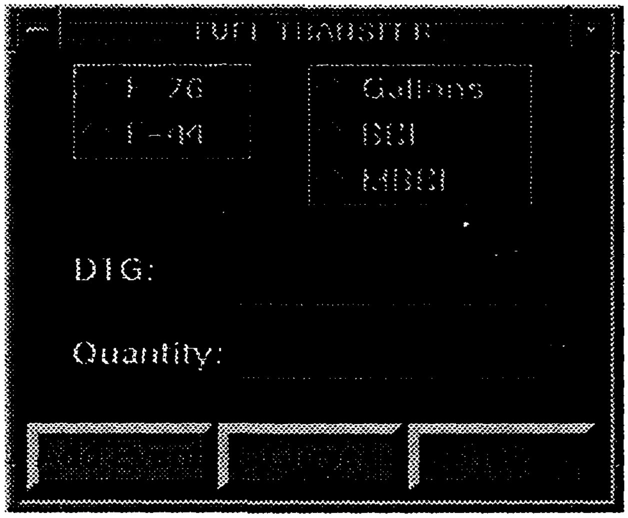

Figure 69: BGLCSS 2.0 Fuel Transfer Panel

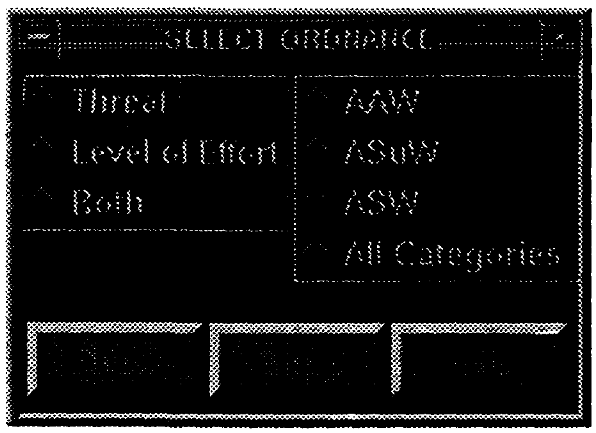

Figure 70: BGLCSS 2.0 Select Ordnance Panel

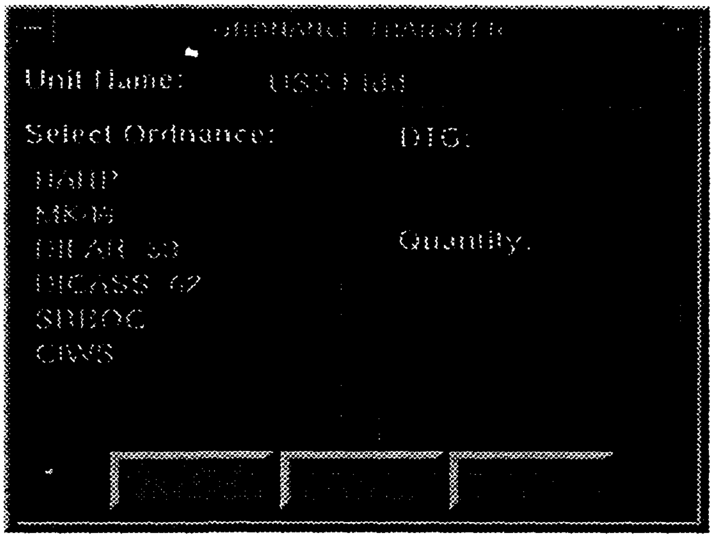

Figure 71: BGLCSS 2.0 Ordnance Transfer Panel

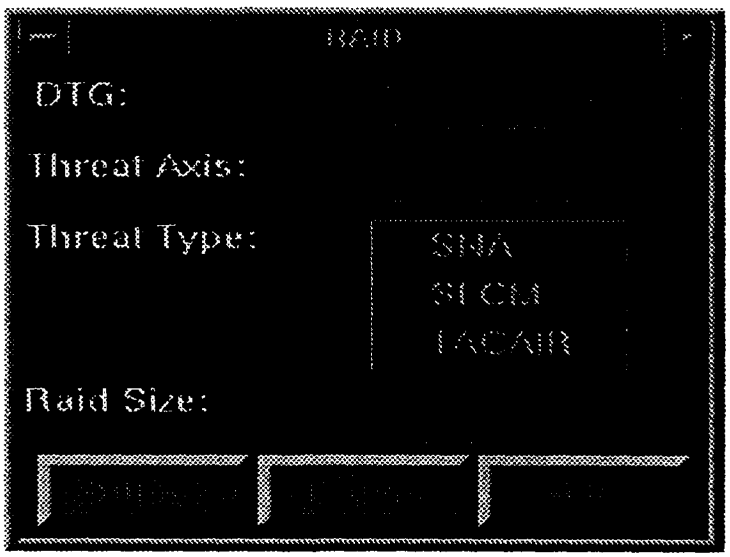

Figure 72: BGLCSS 2.0 Raid Panel

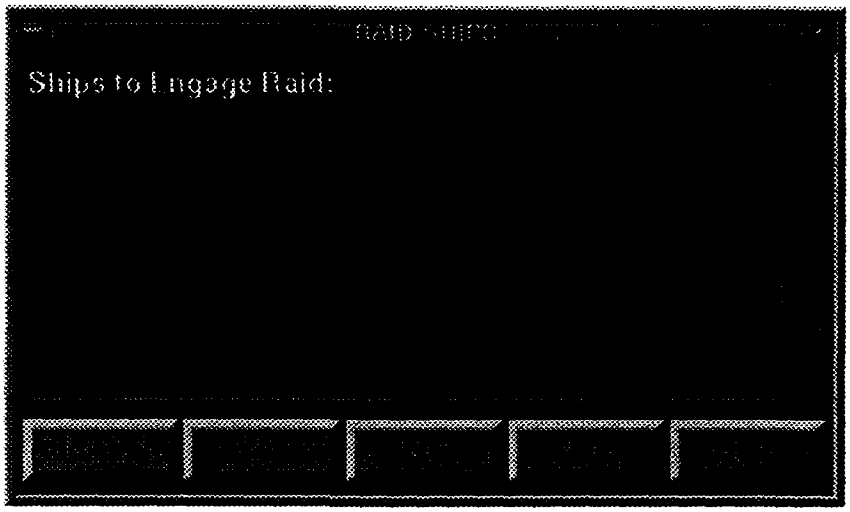

Figure 73: BGLCSS 2.0 Raid Ships Panel

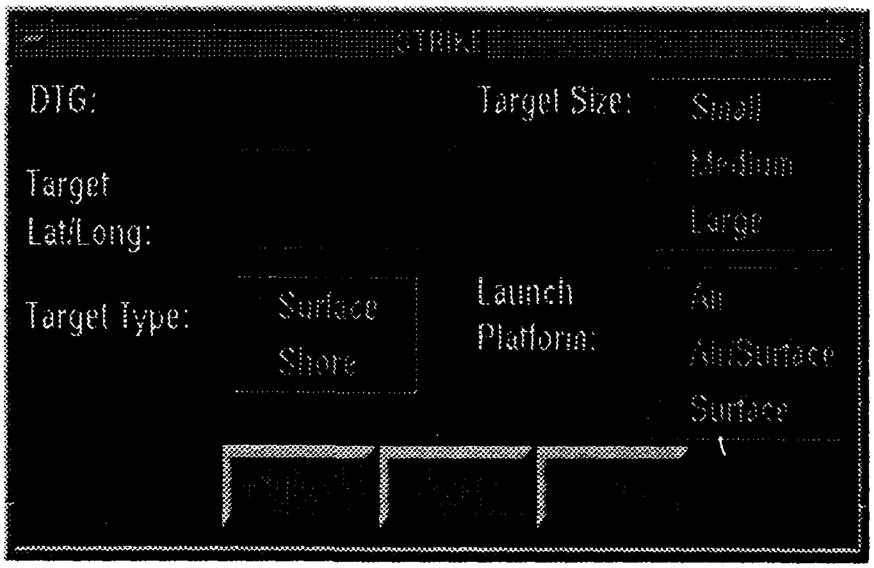

Figure 74: BGLCSS 2.0 Strike Panel

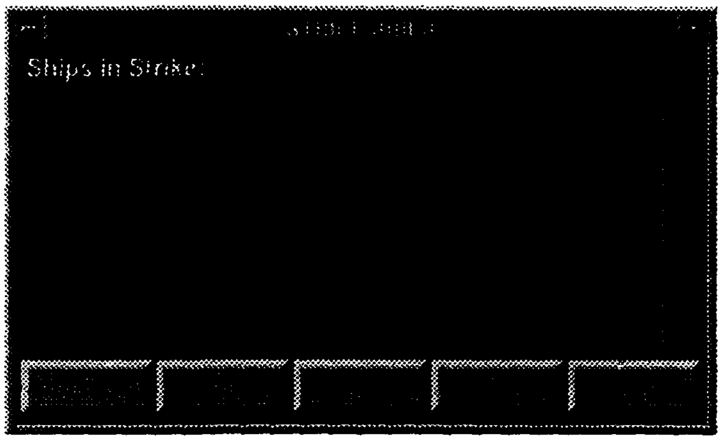

Figure 75: BGLCSS 2.0 Strike Ships Panel

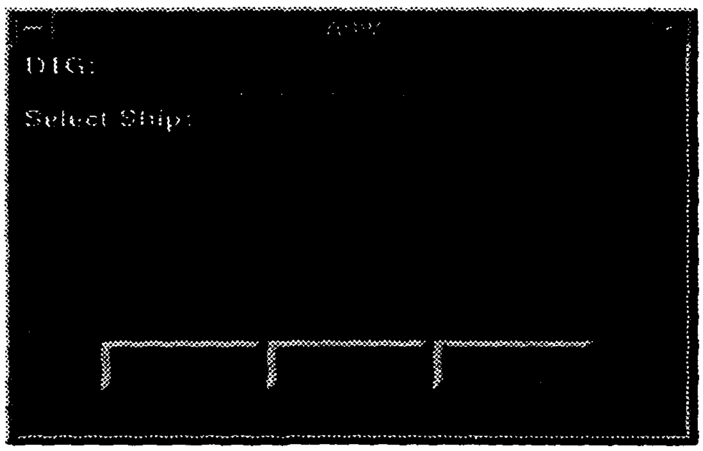

Figure 76: BGLCSS 2.0 ASW Panel

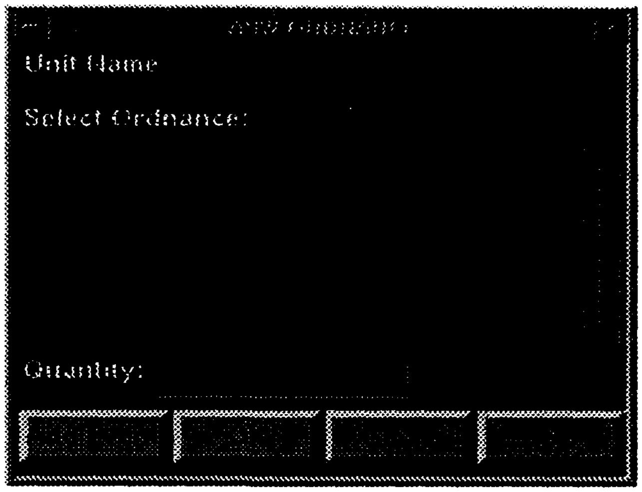

Figure 77: BGLCSS 2.0 ASW Ordnance Panel

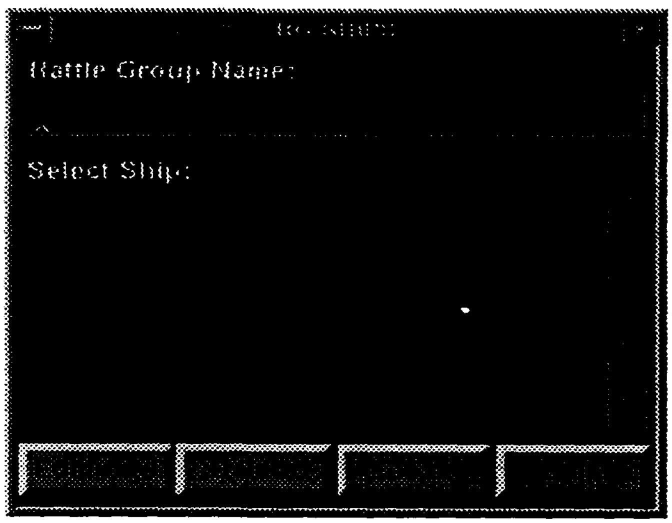

Figure 78: BGLCSS 2.0 Select BG Ship Panel

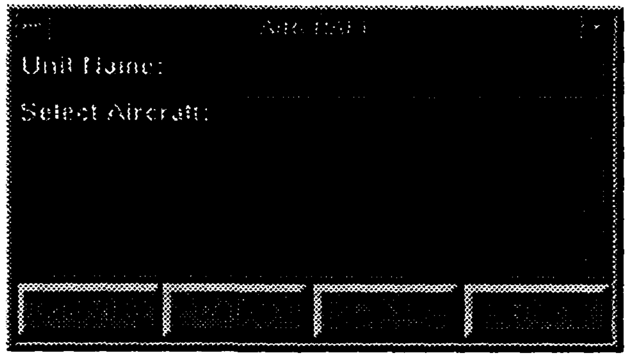

Figure 79: BGLCSS 2.0 Select Ship Aircraft Panel

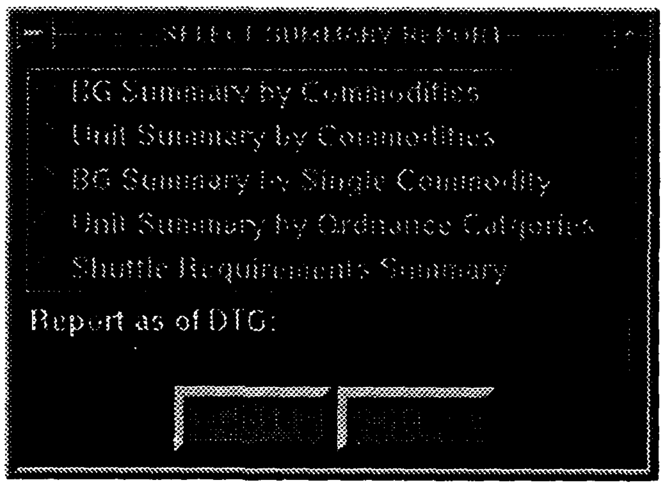

Figure 80: BGLCSS 2.0 Select Summary Report Panel

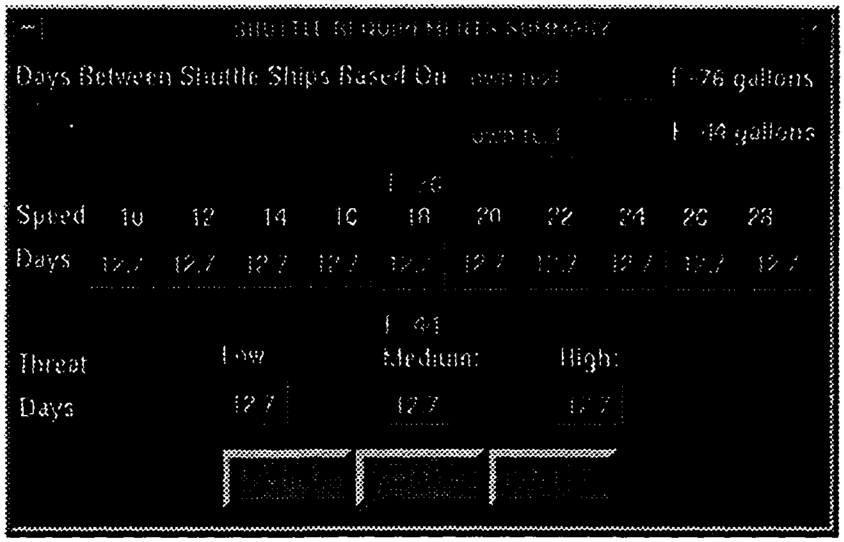

Figure 81: BGLCSS 2.0 Battle Group Shuttle Requirements Report Panel

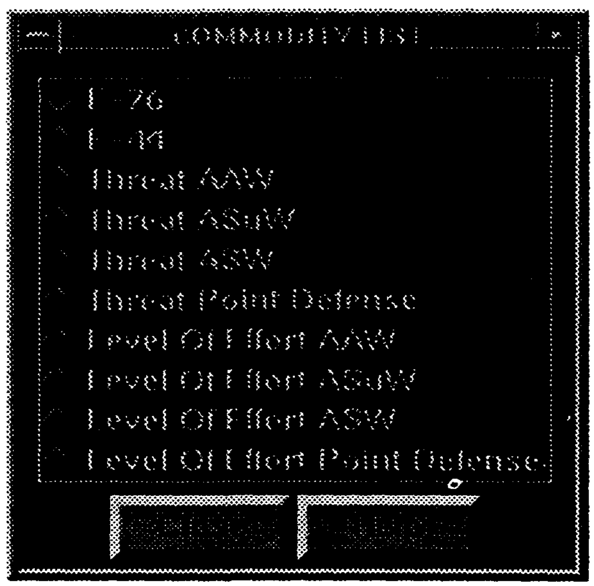

Figure 82: BGLCSS 2.0 Commodity List Panel

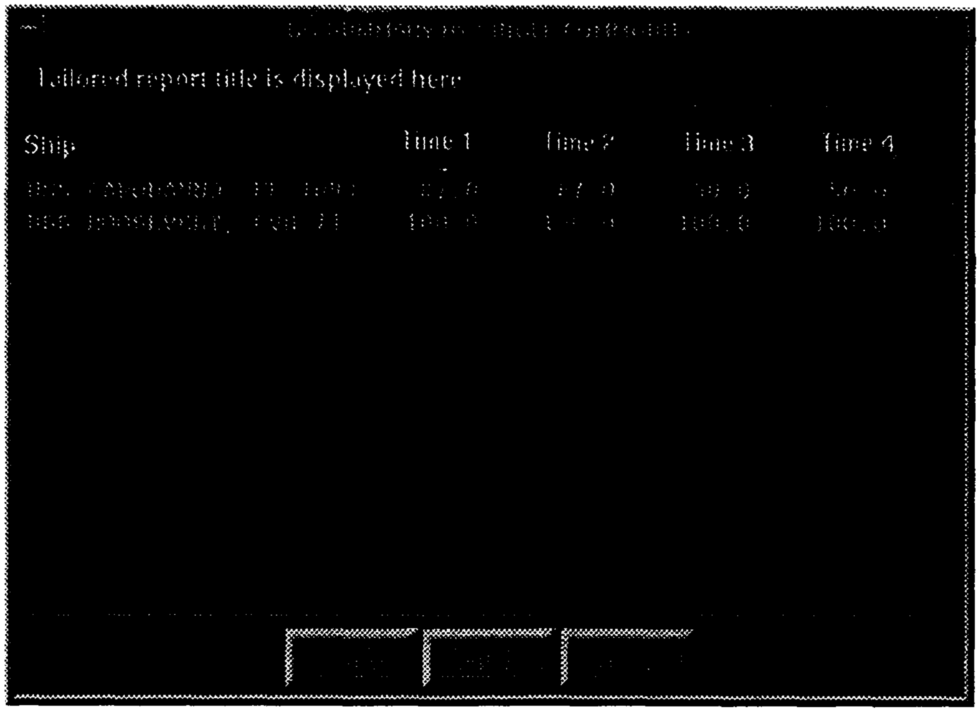

Figure 83: BGLCSS 2.0 BG Summary By Single Commodity Panel

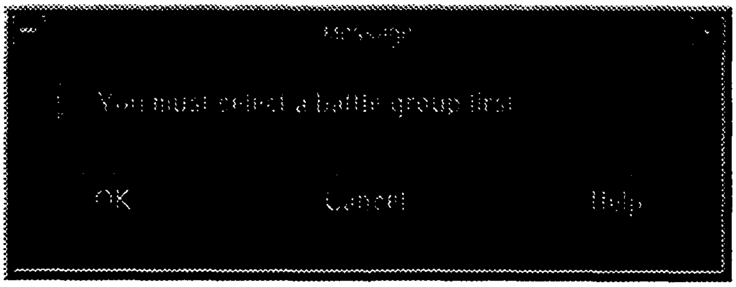

Figure 84: BGLCSS 2.0 Battle Group Selection Message Panel

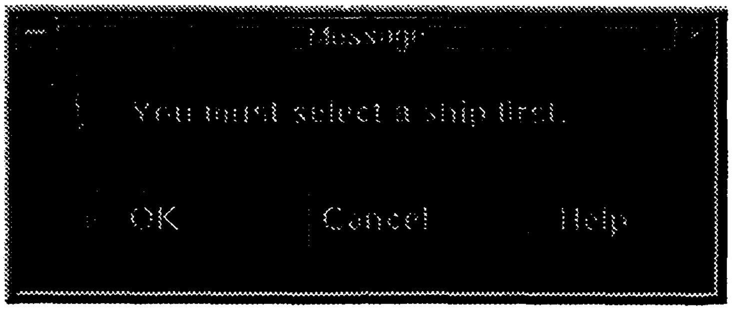

Figure 85: BGLCSS 2.0 Ship Selection Message Panel

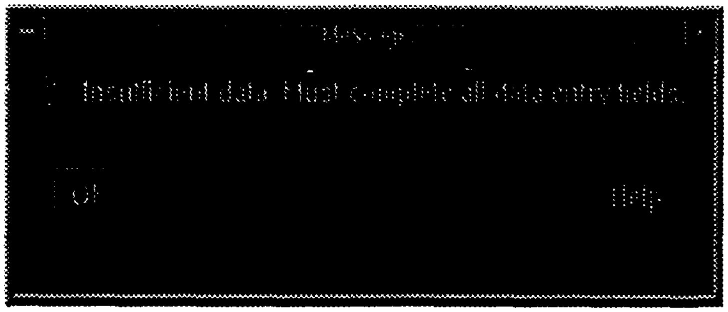

Figure 86: BGLCSS 2.0 Insufficient Data Message Panel

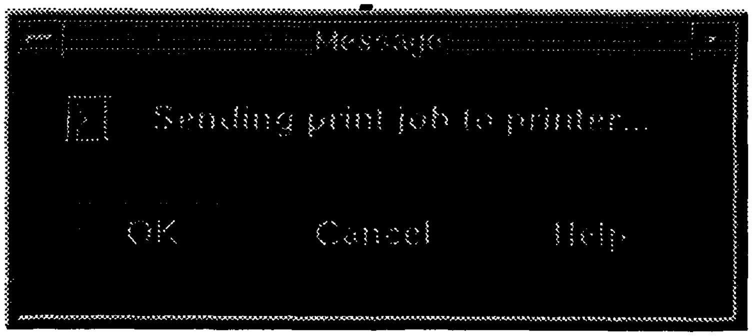

Figure 87: BGLCSS 2.0 Print Job Message Panel

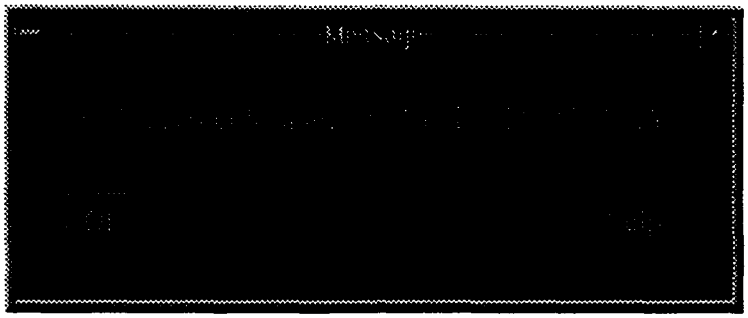

Figure 88: BGLCSS 2.0 Incorrect DTG Format Message Panel

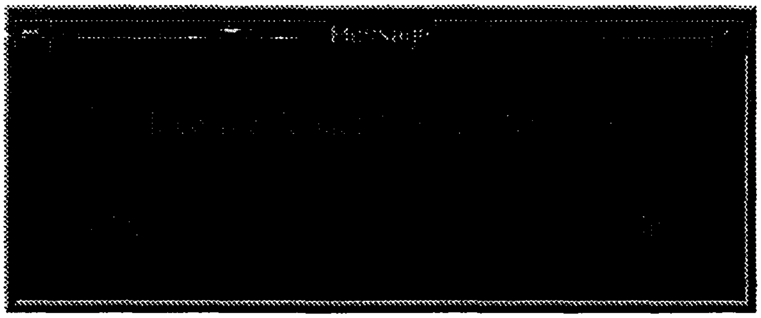

Figure 89: BGLCSS 2.0 Incorrect Lat/Long Format Message Panel

Figure 90: BGLCSS 2.0 Close All Events Panels Message Panel

Figure 91: BGLCSS 2.0 New BG Data Saved Message Panel

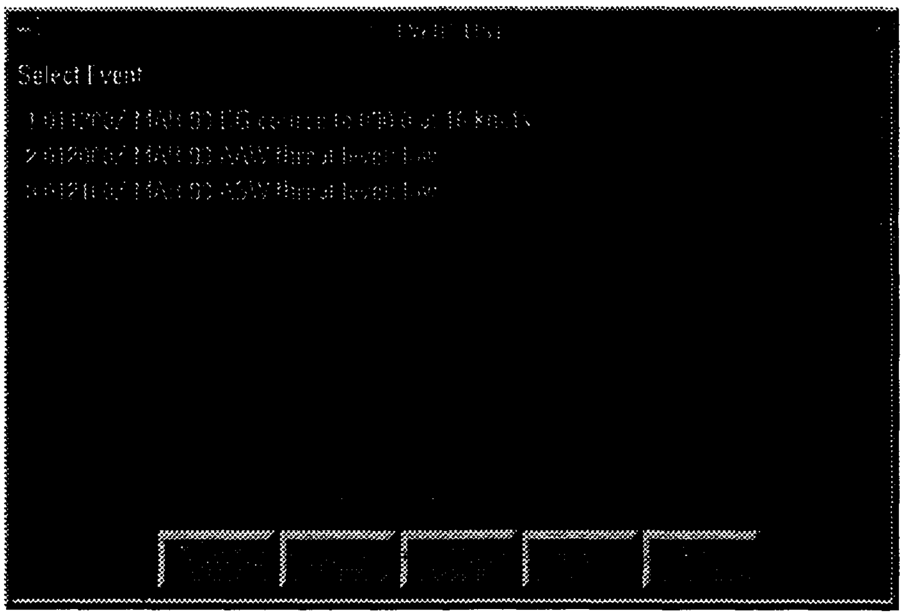

Figure 92: BGLCSS 2.0 Event List Panel

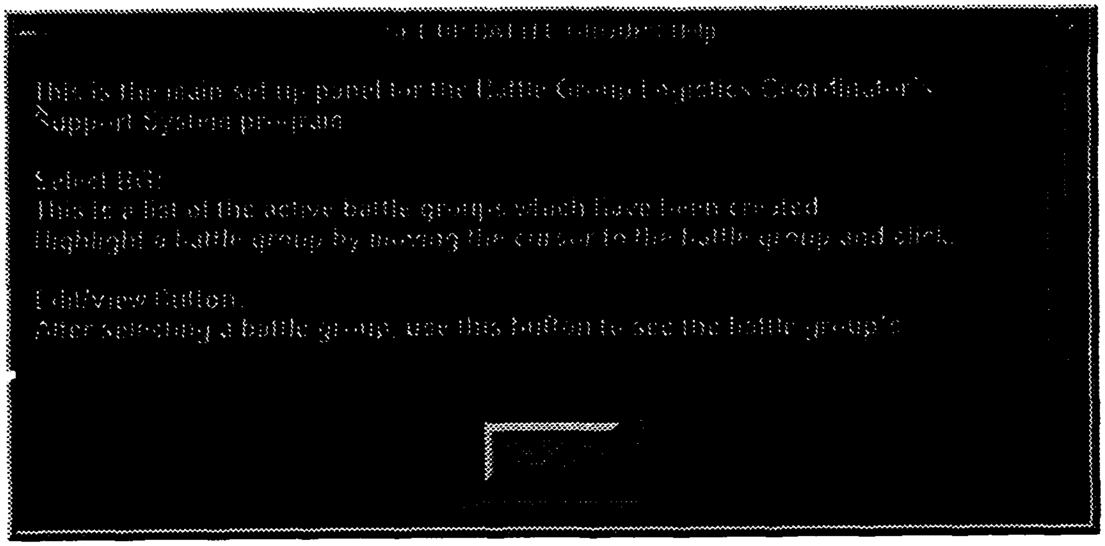

Figure 93: BGLCSS 2.0 Sample Help Panel

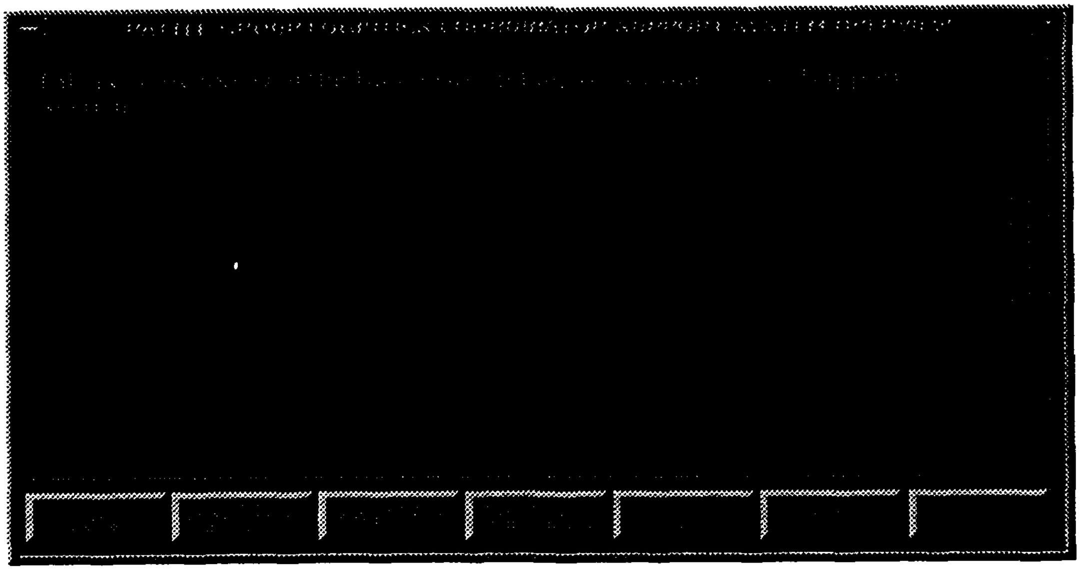

Figure 94: BGLCSS 2.0 Overview Initial Panel

APPENDIX B. BGLCSS 2.0 C PROGRAM LISTING

Files Common to BGSetup and BGEvents modules:

bg.h
bg.c
BGEventsLib.h
BGEventsLib.c
global.h
pan_WptHelp.c
pan_WptHelp.h
wpthelp.c
wpthelp_creat_init.c
wpthelp_init_pan.c

Files Specific to BGSetup:

BGSetup.c
BGSetup_creat_init.c
BGSetup_init_pan.c
Imakefile
pan_BGData.c
pan_BGData.h
pan_BGShips.c
pan_BGShips.h
pan_CloseAll.c
pan_CloseAll.h
pan_DelBG.c
pan_DelBG.h
pan_Dtg.c
pan_Dtg.h
pan_LackData.c
pan_LackData.h
pan_NewBG.c
pan_NewBG.h
pan_SaveNewB.c
pan_SaveNewB.h
pan_SelBG.c
pan_SelBG.h
pan_SetUpBGs.c
pan_SetUpBGs.h
pan_Ship.c
pan_Ship.h

Files Specific to BGEvents module:

BGEvents.c
BGEvents_creat_init.c
BGEvents_init_pan.c
Imakefile
pan_BGCrsSpd.c
pan_BGCrsSpd.h
pan_BGEvents.c
pan_BGEvents.h

Files Specific to Overview module:

Overview.c
Overview_creat_init.c
Overview_init_pan.c
pan_Overview.c
Pan_Overview.h

```
/*****************************************************************
 *Author    :   Bernadette C. Brooks
 *Office    :   Computer Science Department
 *              Naval Postgraduate School
 *              Monterey, CA 93943
 *              Phone: (408) 656-2180
 *Project
 *Advisor   :   Dr. C. Thomas Wu
 *              Computer Science Department
 *              Naval Postgraduate School
 *              Monterey, CA 93943
 *              Phone: (408) 656-3391
 *Filename  :   bg.h
 *Date      :   27 Feb 93
 *Content   :   C manifests, data type definitions, and data
 *              structure definitions for all of BGLCSS 2.0
 *Note      :   "global.h" TAE-generated file includes bg.h
 *****************************************************************
 */

#include <stdio.h>
#include <stdlib.h>

#define MAXNAME             25
#define MAXF76COEF          3
#define MAXSHIPTYPES        8
#define MAXORD              100
#define MAXACFT             20
#define MAXTHREATLEVELS     2       /* Low, Medium, High */
#define MAXENGAGEMENTS      2       /* Raid, Strike, ASW  */
#define MAXINTERVALS        3
#define HOURSINDAY          23
#define MAXUSETYPES         5
#define MAXBGSHIPS          30
#define MAXBGS              10

#define MAXORD              100
#define MAXSHIPS            100
#define DTGLENGTH           15
#define MAXLENGTH           25

#define F76DATA             "/h/bglcss/scripts/data/F76.dat"
#define BGDATA              "/h/bglcss/scripts/data/BGData.dat"
#define BGSHIPS             "/h/bglcss/scripts/data/Ships"
#define NAVYSHIPS           "/h/bglcss/scripts/data/NavyShips.dat"
#define EVENTSDATA          "/h/bglcss/scripts/data/Events"
#define HEADERSDATA         "/h/bglcss/scripts/data/Headers"

/*****************************************************************/

enum CLFType {
        Air,
        Combatant,
        Station,
        Shuttle
};

typedef enum CLFType CLFType;
```

```
/*****************************************************************/

enum AcftType {
        F14,
        FA18,
        A6,
        EA6B,
        E2
};

typedef enum AcftType AcftType;


/*******************************************************************/

typedef struct {
        int     Capacity,
                ReceiveRate,
                TransferRate,
                OnHand,
                EstOnHand,
                Dtg;
        float   Coef[MAXF76COEF];
}F76Info;

/*******************************************************************/

typedef struct {
        int     Capacity,
                ReceiveRate,
                TransferRate,
                OnHand,
                EstOnHand,
                Dtg;
}F44Info;

/*******************************************************************/

typedef struct {
        int             Dtg;
        float           Speed,
                        MaxSpeed;
        double          Latitude,
                        Longitude,
                        Course;
}LocationInfo;

/*******************************************************************/

typedef struct {
        int     Quantity[MAXUSETYPES];
}OrdUse;

/*******************************************************************/

typedef struct {
        char    Name[MAXNAME];
}OrdName;
```

```
/***********************************************************************/

typedef struct {
        OrdName         Name[MAXORD];
        int             TotalNumber,
                        Capacity[MAXORD],
                        Range[MAXORD],
                        TransferRate[MAXORD],
                        OnHand[MAXORD],
                        EstOnHand[MAXORD],
                        OnHandDtg[MAXORD],
                        EstOnHandDtg[MAXORD];
        OrdUse          UseRate[MAXORD];
}OrdInfo;

/***********************************************************************/

typedef struct {
        AcftType        AType;
        int             NumberAcft,
                        FuelBurnedSortie,
                        MaxSortiesDay;
        int             SortieRate[MAXTHREATLEVELS];
        int             NumSorties[MAXENGAGEMENTS];
}AcftRecord;

/***********************************************************************/

typedef struct {
        int             SortieFlown[HOURSINDAY];
        AcftRecord      Wing[MAXACFT];
}AcftInfo;

/***********************************************************************/

typedef struct {
        char            Name[40];
        char            Hull[10];
        CLFType         TypeCombatant;
        LocationInfo    Location;
        F76Info         F76;
        F44Info         F44;
        int             Approach,
                        BreakAway;
        OrdInfo         Ord;
        AcftInfo        Acft;
}ShipInfo;

/***********************************************************************/

typedef struct {
        char    ShipType[10];
        int     F76Capacity,
                F76Receive,
                F76Transfer;
        float   Coef[MAXF76COEF];
}F76ShipTypeInfo;
```

```
/***********************************************************************/

typedef struct {
        float   FuelRes,
                CLFFuelRes,
                OrdRes,
                CLFOrdRes,
                MaxF76,
                 MaxF44,
                StationSpeed,
                UnrepSpeed,
                AcftShipSpeed;
        int     PredictStart;
        int     PredictHours[MAXINTERVALS];
}SettingsInfo;

/***************************************************************************/

typedef struct{
        int     TotalNumber;
}OrdCapInfo;

/***************************************************************************/

typedef struct{
        float           F76Capacity[MAXINTERVALS],
                        F44Capacity[MAXINTERVALS];
        OrdCapInfo      OrdCapacity[MAXINTERVALS];
}CapacityInfo;

/***************************************************************************/

typedef struct {
        char            Name[MAXNAME];
        char            Designation[MAXNAME];
        SettingsInfo    Settings;
        LocationInfo    Location;
        ShipInfo        Ships[MAXBGSHIPS];
        CapacityInfo    Results[MAXBGSHIPS];
}BGInfo;

/***************************************************************************/
```

```
/*****************************************************************
 *Author    :   Bernadette C. Brooks
 *Office    :   Computer Science Department
 *              Naval Postgraduate School
 *              Monterey, CA 93943
 *              Phone: (408) 656-2180
 *Project
 *Advisor   :   Dr. C. Thomas Wu
 *              Computer Science Department
 *              Naval Postgraduate School
 *              Monterey, CA 93943
 *              Phone: (408) 656-3391
 *Filename  :   bg.c
 *Date      :   27 Feb 93
 *Content   :   Bodies of user-defined functions to represent battle
 *              groups and ships. C manifests contained in bg.h
 *Note      :   "global.h" TAE-generated file includes bg.h
 *****************************************************************
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

/*TAE system include files*/
#include        "taeconf.inp"
#include        "wptinc.inp"
#include        "symtab.inc"
#include        "parblk.inc"
#include        "terminc.inc"
#include        "global.h"

/*****************************************************
 * Convert an integer to the appropriate string representation of
 * TypeCombatant. C stores the value of an enumerated type in an
 * ASCII file as an integer. To display this value in a panel, it must be
 * converted to a string.
 *****************************************************
 */

char* ConvertTypeCombatant (Integer)
int Integer;
{
        char    TempString[MAXNAME];

        switch (Integer) {

        case 0:
                strcpy(TempString, "Aviation");
                break;
        case 1:
                strcpy(TempString, "Combatant");
                break;
        case 2:
                strcpy(TempString, "Station CLF");
                break;
        case 3:
                strcpy(TempString, "Shuttle");
                break;
        default:
```

```
                strcpy(TempString, "unknown");
        }
        return(TempString);
}

/*************************************************************
 * Convert an integer to its appropriate string representation.
 * UNIX K & R C library does not contain this kind of function.
 * DOS C does. It is needed for I/O to the ASCII text files containing
 * ship, battle group, and event information.
 *************************************************************
 *

char* IntToString(i)
int i;
{
        switch(i){
        case 0:
                return("0"); break;
        case 1:
                return("1"); break;
        case 2:
                return("2"); break;
        case 3:
                return("3"); break;
        case 4:
                return("4"); break;
        case 5:
                return("5"); break;
        case 6:
                return("6"); break;
        case 7:
                return("7"); break;
        case 8:
                return("8"); break;
        case 9:
                return("9"); break;
        case 10:
                return("10"); break;
        default:
                return("11");
        }
}

/*************************************************************
 * Using the string of a ship name and the index to the appropriate battle
 * group, GetShip returns the ship index for the appropriate ship.
 *************************************************************
 */

int GetShip(Name, i)
char    Name[MAXNAME];
int     i;
{
        char    FileName[80];
        FILE*   DataFile;
        char    Suffix[MAXNAME];
        TEXT*   temp[MAXBGS];
        char    Cmd[MAXNAME];
        int     s;
        char    buf[10];
```

```
        int     Found;

        strcpy(FileName, BGSHIPS);

        strcpy(Suffix,IntToString(i));

        strcat(Suffix, ".dat");

        strcat(FileName, Suffix);

        DataFile = fopen(FileName, "r");

        while (!feof(DataFile)){

                fscanf(DataFile,"%[^\n]\n", Cmd);
                s = 0;

                /*skip over all ship info*/
                for (s = 0; s < 13; s++){
                        fscanf(DataFile,"%s\n", buf);
                }

                if (strcmp(Name,Cmd) == 0){
                        Found = 1;
                } else{
                        i++;
                }
        }

        fclose(DataFile);

        if (Found == 0){
                return(-1);
        }else{
                return(i);
        }
}

/***********************************************************
 * This function uses TAE functions to display values to the Ship panel.
 ***********************************************************
 */

void ShowShip(PanelView, PanelId, BGs, i, s)
Id      PanelView,
        PanelId;
BGInfo  BGs[MAXBGS];
int     i,
        s;
{
        TEXT*  name[1];
        TEXT*  hull[1];

        strcpy(name[0],BGs[i].Ships[s].Name);
        Vm_SetString(PanelView,"ShipName.textstrs",1,name,P_UPDATE);
        Wpt_ViewUpdate(PanelId,"ShipName", PanelView,"ShipName");

        strcpy(hull[0],BGs[i].Ships[s].Hull);
        Vm_SetString(PanelView,"Hull.textstrs",1,hull,P_UPDATE);
        Wpt_ViewUpdate(PanelId,"Hull", PanelView,"Hull");
```

```
        strcpy(name[0],BGs[i].Name);
        Vm_SetString(PanelView,"BGName.textstrs",1,name,P_UPDATE);
        Wpt_ViewUpdate(PanelId,"BGName", PanelView,"BGName");

        strcpy(name[0],
        ConvertTypeCombatant(BGs[i].Ships[s].TypeCombatant));


        Vm_SetString
        (PanelView,"TypeCombatant.textstrs",1,name,P_UPDATE);
        Wpt_ViewUpdate(PanelId,"TypeCombatant",
        PanelView,"TypeCombatant");

        strcpy(name[0],dtg_to_a(BGs[i].Ships[s].Location.Dtg));
        Vm_SetString(PanelView,"Dtg.textstrs",1,name,P_UPDATE);
        Wpt_ViewUpdate(PanelId,"Dtg", PanelView,"Dtg");

        Wpt_SetIntg(PanelId, "Approach", 20);
        Wpt_SetIntg(PanelId, "BreakAway",10);

}

/**********************************************************
 * This function uses TAE functions to display values to the Ship panel
 **********************************************************
 */

void ShowF76 (BGs, i, s, PanelId, PanelView)
BGInfo  BGs[MAXBGS];
int     i;
int     s;
Id      PanelId,
        PanelView;
{
        TEXT* name[1];
        TEXT* estonhand[1];
        TEXT* estdtg[1];

        strcpy(name[0], BGs[i].Ships[s].Name);

        Vm_SetString(PanelView, "Name.textstrs", 1, name, P_UPDATE);
        Wpt_ViewUpdate(PanelId, "Name", PanelView, "Name");

        Wpt_SetIntg(PanelId, "OnHand",  BGs[i].Ships[s].F76.OnHand);
        Wpt_SetIntg(PanelId, "Capacity", BGs[i].Ships[s].F76.Capacity);

        /*not implemented yet*/
        /*Vm_SetString(PanelView, "EstOnHand.textstrs", 1,
        estonhand, P_UPDATE);
        Wpt_ViewUpdate(PanelId, "EstOnHand", PanelView, "EstOnHand");*/

        /*need to convert date integer to string representations first*/
        /*not implemented yet*/
        /*Vm_SetString(PanelView, "EstDtg.textstrs", 1, estdtg, P_UPDATE);
        Wpt_ViewUpdate(PanelId, "EstDtg", PanelView, "EstDtg");
        */

        /*not implemented yet*/
        /*Wpt_SetIntg(PanelId, "ReceiveRate",
        BGs[i].Ships[s].F76.ReceiveRate);
        Wpt_SetIntg(PanelId,
```

```
        "TransferRate",BGs[i].Ships[s].F76.TransferRate);
        */
}

/*********************************************************
 * Using the BG array and the index to the specific battle group, this
 * function saves the battle group's ships' data to an ASCII text file.
 *********************************************************
 */

void SaveBGShips(BGs,i)
BGInfo  BGs[MAXBGS];
int     i;
{
        char    FileName[100];
        char    Suffix[MAXNAME];
        static  blank[MAXNAME];
        FILE*   DataFile;
        int     s;
        int     Index;
        static  char Cmd[100];

        s = 0;

        /*Based on BG Index, create appropriate file name for BG ships. Ex:
        For ships in BGs[0], filename is /h/bglcss/scripts/data/Ships0.dat*/

        strcpy(FileName, BGSHIPS);
        strcpy(Suffix,IntToString(i));
        strcat(Suffix, ".dat");
        strcat(FileName, Suffix);

        /*use system call to remove previous file*/
        strcpy(Cmd, "rm ");
        strcat(Cmd, FileName);
        system(Cmd);

        DataFile = fopen (FileName, "w");
        /*Fill in F76 Table for ships by type*/

        /*Read in ShipInfo for ships in this BG from appropriate file*/
        while(s < MAXBGS){
                if(strcmp(BGs[i].Ships[s].Name,blank) != 0){
                        fprintf(DataFile, "%s\n", BGs[i].Ships[s].Name);
                        fprintf(DataFile, "%s\n", BGs[i].Ships[s].Hull);
                        fprintf(DataFile, "%d\n",
                        BGs[i].Ships[s].TypeCombatant);

                        fprintf(DataFile, "%d\n",
                        BGs[i].Ships[s].Location.Dtg);

                        fprintf(DataFile, "%.3f\n",
                        BGs[i].Ships[s].Location.Speed);

                        fprintf(DataFile, "%.3f\n",
                        BGs[i].Ships[s].Location.MaxSpeed);

                        fprintf(DataFile, "%.3lf\n",
                        BGs[i].Ships[s].Location.Latitude);
```

```
                        fprintf(DataFile, "%.3lf\n",
                        BGs[i].Ships[s].Location.Longitude);

                        fprintf(DataFile, "%.3lf\n",
                        BGs[i].Ships[s].Location.Course);

                        fprintf(DataFile, "%d\n",
                        BGs[i].Ships[s].F76.OnHand);

                        fprintf(DataFile, "%d\n",
                        BGs[i].Ships[s].F76.EstOnHand);

                        fprintf(DataFile, "%d\n", BGs[i].Ships[s].F76.Dtg);

                        /*F44Info here; not implemented yet*/

                        fprintf(DataFile, "%d\n", BGs[i].Ships[s].Approach);
                        fprintf(DataFile, "%d\n", BGs[i].Ships[s].BreakAway);

                        /*OrdInfo here; not implemented yet*/

                        /*AcftInfo here; not implemented yet*/
                }
                s++;
        }
        fclose(DataFile);
}

/*********************************************************
 * Get the F76 information by ship type from the ASCII text file into
 * memory.
 *********************************************************
 */

void GetF76Table(Table)
F76ShipTypeInfo Table[MAXSHIPTYPES];
{
    FILE*       DataFile;
    int         i = 0;

    DataFile = fopen(F76DATA, "r");

    while (!feof(DataFile)) {
        fscanf(DataFile, "%s",    Table[i].ShipType);
        fscanf(DataFile, "%d",  &Table[i].F76Capacity);
        fscanf(DataFile, "%d",  &Table[i].F76Receive);
        fscanf(DataFile, "%d",  &Table[i].F76Transfer);
        fscanf(DataFile, "%f",  &Table[i].Coef[0]);
        fscanf(DataFile, "%f",  &Table[i].Coef[1]);
        fscanf(DataFile, "%f\n",&Table[i].Coef[2]);
        i++;
    }
            fclose(DataFile);
}
```

```
/*****************************************************************
 * Get the battle group ships from the ship data ASCII text file and the
 * F76 information by ship type from memory. The next ship index available
 * is returned as an integer.
 *****************************************************************
 */

int GetBGShips(BGs,i, F76Table)
BGInfo                    BGs[MAXBGS];
int                       i;
F76ShipTypeInfo           F76Table[MAXSHIPTYPES];
{
        char        FileName[100];
        char        Suffix[MAXNAME];
        static      char blank[MAXNAME];
        FILE*       DataFile;
        int         s;
        int         Index;

        s = 0;

        /*Based on BG Index, create appropriate file name for BG ships. Ex:
        For ships in BGs[0], filename is /h/bglcss/scripts/data/Ships0.dat*/

        strcpy(FileName, BGSHIPS);
        strcpy(Suffix,IntToString(i));
        strcat(Suffix, ".dat");
        strcat(FileName, Suffix);
        DataFile = fopen (FileName, "r");

        /*Read in ShipInfo for ships in this BG from appropriate file*/
        while(!feof(DataFile)){

                fscanf(DataFile, "%[^\n]\n",BGs[i].Ships[s].Name);
                fscanf(DataFile, "%s\n",  BGs[i].Ships[s].Hull);

                Index = TypeShip(BGs[i].Ships[s].Hull);

                BGs[i].Ships[s].F76.Capacity    =F76Table[Index].F76Capacity;
                BGs[i].Ships[s].F76.ReceiveRate=
                F76Table[Index].F76Receive;
                BGs[i].Ships[s].F76.TransferRate=
                F76Table[Index].F76Transfer;

                BGs[i].Ships[s].F76.Coef[0]   = F76Table[Index].Coef[0];
                BGs[i].Ships[s].F76.Coef[1]   = F76Table[Index].Coef[1];
                BGs[i].Ships[s].F76.Coef[2]   = F76Table[Index].Coef[2];

                fscanf(DataFile, "%d\n",  &BGs[i].Ships[s].TypeCombatant);
                fscanf(DataFile, "%d\n",  &BGs[i].Ships[s].Location.Dtg);
                fscanf(DataFile, "%f\n",  &BGs[i].Ships[s].Location.Speed);
                fscanf(DataFile, "%f\n",  &BGs[i].Ships[s].Location.MaxSpeed);
                fscanf(DataFile, "%lf\n", &BGs[i].Ships[s].Location.Latitude);
                fscanf(DataFile, "%lf\n", &BGs[i].Ships[s].Location.Longitude);
                fscanf(DataFile, "%lf\n", &BGs[i].Ships[s].Location.Course);

                /*Read in last current F76 states*/
                fscanf(DataFile, "%d\n", &BGs[i].Ships[s].F76.OnHand);
                fscanf(DataFile, "%d\n", &BGs[i].Ships[s].F76.EstOnHand);
                fscanf(DataFile, "%d\n", &BGs[i].Ships[s].F76.Dtg);
```

```
                /*F44Info here; not implemented yet*/

                fscanf(DataFile, "%d\n", &BGs[i].Ships[s].Approach);
                fscanf(DataFile, "%d\n", &BGs[i].Ships[s].BreakAway);

                /*OrdInfo here; not implemented yet*/

                /*AcftInfo here; not implemented yet*/
                s++;
        }
        fclose(DataFile);
        return(s);
}

/*******************************************************************
 * Make a new battle group, using information provided by user to New BG
 * Panel. The index for the next available battle group in the array is
 * returned.
 *******************************************************************
 */

int MakeBG(BGs,i,Name,Desg,FRes,CRes,ORes,CORes,F76,F44,SSpeed,
           USpeed,ASSpeed)

BGInfo          BGs[MAXBGS];
int             i;
char            Name[MAXNAME];
char            Desg[MAXNAME];
float           FRes, CRes, ORes, CORes,
                F76, F44, SSpeed, USpeed, ASSpeed;
{
        if (Name && Desg && FRes && CRes && ORes && CORes
          && F76 && F44 && SSpeed && USpeed && ASSpeed) {

                strcpy(BGs[i].Name, Name);
                strcpy(BGs[i].Designation, Desg);

                BGs[i].Settings.FuelRes                 = FRes;

                BGs[i].Settings.CLFFuelRes              = CRes;
                BGs[i].Settings.OrdRes                  = ORes;
                BGs[i].Settings.CLFOrdRes               = CORes;
                BGs[i].Settings.MaxF76                  = F76;
                BGs[i].Settings.MaxF44                  = F44;
                BGs[i].Settings.StationSpeed            = SSpeed;
                BGs[i].Settings.UnrepSpeed              = SSpeed;
                BGs[i].Settings.AcftShipSpeed           = ASSpeed;
                i++;
                return (1);
        } else {
                return (0);
        }
}
```

```
/*************************************************************************
 * Save all battle groups to the ASCII text file containing BG data.
 *************************************************************************
 */

void SaveBGs(BGs)
BGInfo BGs[MAXBGS];
{
	FILE*		DataFile;
	int		i = 0;
	static char	blank[MAXNAME];

	DataFile = fopen(BGDATA, "w");

	while (i < MAXBGS){
		if (strcmp(BGs[i].Name, blank) != 0){
			fprintf(DataFile,"%s\n", BGs[i].Name);
			fprintf(DataFile,"%s\n", BGs[i].Designation);
			fprintf(DataFile,"%.1f\n",BGs[i].Settings.FuelRes);
			fprintf(DataFile,"%.1f\n",BGs[i].Settings.CLFFuelRes);
			fprintf(DataFile,"%.1f\n",BGs[i].Settings.OrdRes);
			fprintf(DataFile,"%.1f\n",BGs[i].Settings.CLFOrdRes);
			fprintf(DataFile,"%.1f\n",BGs[i].Settings.MaxF76);
			fprintf(DataFile,"%.1f\n",BGs[i].Settings.MaxF44);
			fprintf(DataFile,"%.1f\n",BGs[i].Settings.StationSpeed);
			fprintf(DataFile,"%.1f\n",BGs[i].Settings.UnrepSpeed);
			fprintf(DataFile,"%.1f\n",
			BGs[i].Settings.AcftShipSpeed);
		}
		i++;
	}
	fclose(DataFile);
}

/*************************************************************
 * Show list of battle group ships, given battle group index, panel name,
 * and selection list item on panel.
 *************************************************************
 */

void ShowBGShips(i, Panel, ItemName)
int		i;
Id		Panel;
char		ItemName[15];
{
	char		FileName[80];
	FILE*		DataFile;
	char		Suffix[MAXNAME];
	TEXT*		temp[MAXBGS];
	char		Cmd[MAXNAME];
	char		buff[10];
	TAEINT		a, z;
	static char	blank[MAXNAME];

	a = 0;

	/*Based on BG Index, get the appropriate file name for BG ships. Ex:
	For ships in BGs[0], filename is /h/bglcss/scripts/data/Ships0.dat*/
	strcpy(FileName, BGSHIPS);
	strcpy(Suffix, IntToString(i));
	strcat(Suffix, ".dat");
```

```
        strcat(FileName, Suffix);

        DataFile = fopen(FileName, "r");

        while (!feof(DataFile)){
                fscanf(DataFile,"%[^\n]\n",  &Cmd[0]);
                z = 0;

                /*skip over data until reach ship name*/
                for (z = 0; z < 13; z++){
                        fscanf(DataFile,"%[^\n]\n", &buff[0]);
                }
                temp[a]=(TEXT *) malloc(strlen(Cmd)+1);
                strcpy(temp[a], Cmd);
                a++;
        }

        fclose(DataFile);

        Wpt_SetStringConstraints(Panel,ItemName,a,temp);
}

/**************************************************************
 * Show battle group data given battle group array, index,and panel name.
 **************************************************************
 */

void ShowBG(BGs, i, Panel)
BGInfo          BGs[MAXBGS];
int             i;
Id              Panel;
{
    Wpt_SetString(Panel, "Name",                 BGs[i].Name);
    Wpt_SetString(Panel, "Designation",          BGs[i].Designation);

    Wpt_SetReal(Panel, "FuelRes",                BGs[i].Settings.FuelRes);
    Wpt_SetReal(Panel, "CLFFuelRes",             BGs[i].Settings.CLFFuelRes);
    Wpt_SetReal(Panel, "OrdRes",                 BGs[i].Settings.OrdRes);
    Wpt_SetReal(Panel, "CLFOrdRes",              BGs[i].Settings.CLFOrdRes);
    Wpt_SetReal(Panel, "MaxF76",                 BGs[i].Settings.MaxF76);
    Wpt_SetReal(Panel, "MaxF44",                 BGs[i].Settings.MaxF44);
    Wpt_SetReal(Panel, "StationSpeed",           BGs[i].Settings.StationSpeed);
    Wpt_SetReal(Panel, "UnrepSpeed",             BGs[i].Settings.UnrepSpeed);
    Wpt_SetReal(Panel, "AcftShipSpeed",          BGs[i].Settings.AcftShipSpeed);

}

/***********************************************************
 * Show list of navy ships from ASCII text file to item in panel.
 ***********************************************************
 */

void ShowNavyShips(Panel, ItemName)
Id      Panel;
char    ItemName[15];
{
        TEXT*           temp[MAXBGS];
        FILE*           DataFile;
        char            Cmd[MAXNAME];
        TAEINT          a;
```

```
        a = 0;

        DataFile = fopen(NAVYSHIPS, "r");

        while (!feof(DataFile)){

                fscanf(DataFile,"%[^\n]\n",  &Cmd[0]);

                temp[a]=(TEXT *) malloc(strlen(Cmd)+1);

                strcpy(temp[a], Cmd);

                a++;
        }
        Wpt_SetStringConstraints(Panel, ItemName,a,temp);

        fclose(DataFile);
}

/************************************************************
 * Show list of battle groups from ASCII text file to item in panel.
 ************************************************************
 */

void ShowBGs(Panel,ItemName)
Id      Panel;
char    ItemName[15];
{
        TEXT*           temp[MAXBGS];
        FILE*           DataFile;
        char            Cmd[MAXNAME];
        char            buff[10];
        TAEINT          a, i;
        static  char    blank[MAXNAME];

        a = 0;

        DataFile = fopen(BGDATA, "r");

        while (!feof(DataFile)){
                fscanf(DataFile,"%[^\n]\n",  &Cmd[0]);
                i = 0;
                for (i = 0; i < 10; i++){
                        fscanf(DataFile,"%[^\n]\n", &buff[0]);
                }
                if (strcmp(Cmd,blank) != 0){
                        temp[a]=(TEXT *) malloc(strlen(Cmd)+1);
                        strcpy(temp[a], Cmd);
                        a++;
                }
        }
        Wpt_SetStringConstraints(Panel, ItemName,a, temp);
        fclose(DataFile);
}
```

```
/***********************************************************
 * Save new battle group using Id Target from user input panel. First,
 * GetBGs is called, returning the available index to the array. Next,
 * Make BG is called and then SaveBGS saves all battle groups.
************************************************************
 */

int SaveNewBG(Target)
Id Target;
{
    BGInfo BGs[MAXBGS];
    int    BGIndex;

    BGIndex = GetBGs(BGs);

    if (MakeBG(BGs,BGIndex,StringParm(Target,"Name"),
        StringParm(Target,"Designation"),
                RealParm (Target,"FuelRes"),
                RealParm (Target,"CLFFuelRes"),
                RealParm (Target,"OrdRes"),
                RealParm (Target,"CLFOrdRes"),
                RealParm (Target,"MaxF76"),
                RealParm (Target,"MaxF44"),
                RealParm (Target,"StationSpeed"),
                RealParm (Target,"UnrepSpeed"),
                RealParm (Target,"AcftShipSpeed"))) {

                SaveBGs(BGs);
                return(1);
        } else {
                return(0);
        }
}

/***********************************************************
 * This function merely wipes out the contents of a battle group panel.
************************************************************
 */

void CancelBG(Panel)
Id Panel;
{
        Wpt_SetNoValue(Panel,"Name");
        Wpt_SetNoValue(Panel,"Designation");
        Wpt_SetNoValue(Panel,"FuelRes");
        Wpt_SetNoValue(Panel,"CLFFuelRes");
        Wpt_SetNoValue(Panel,"OrdRes");
        Wpt_SetNoValue(Panel,"CLFOrdRes");
        Wpt_SetNoValue(Panel,"MaxF76");
        Wpt_SetNoValue(Panel,"MaxF44");
        Wpt_SetNoValue(Panel,"StationSpeed");
        Wpt_SetNoValue(Panel,"UnrepSpeed");
        Wpt_SetNoValue(Panel,"AcftShipSpeed");
        Wpt_SetNoValue(Panel,"BGShips");
}
```

```
/************************************************************
 * Using the string representation of the name of a battle group and the
 * index to the appropriate battle group in the array, return the index to
 * the battle group.
 ************************************************************
 */

int GetBG(Name, BGIndex)
char        Name[MAXNAME];
int         BGIndex;
{
            FILE*   DataFile;
            int     i,s;
            char    Cmd[MAXNAME];
            char    buf[MAXNAME];
            int     Found;

            Found = 0;

            i = 0;

            DataFile = fopen(BGDATA, "r");
            while ((!feof(DataFile)) && (Found == 0)){

                    fscanf(DataFile,"%[^\n]\n", &Cmd[0]);

                    s = 0;
                    for (s = 0; s < 10; s++){
                            fscanf(DataFile,"%s\n", &buf[0]);
                    }

                    if (strcmp(Name,Cmd) != 0){
                            i++;
                    } else{
                            Found = 1;
                    }

            }
            fclose(DataFile);

            if (Found == 0){
                    return(BGIndex);
            } else {
                    return(i);
            }
}

/************************************************************
 * This deletes the battle group from the array by removing its name.
 * The name is replaced by blank spaces.
 ************************************************************
 */

void DeleteBG(BGs, i)
BGInfo   BGs[MAXBGS];
int         i;
{
            static char blank[MAXNAME];

            if (i != -1){
                    strcpy(BGs[i].Name,blank);
```

```
	}
}

/******************************************************
 * This function checks the first two characters in a ship's hull number
 * and returns an integer that equates to an enumerated ship type.
 ******************************************************
 */

int TypeShip(String)
char	String[40];
{
	int	s;
	char	Slice[40];

	strcpy(Slice, String);

	Slice[2] = 0;

	if (strcmp(Slice, "DD") == 0) {
		return (3);
	} else if (strcmp(Slice, "AO") == 0) {
		return (6);
	}
}

/************************************************************
 * This function adds a ship and its data to a battle group.
 * The ship list presented to the user contains both the hull number
 * and the ship name. The hull number is required to get the ship type for
 * the appropriate F76 information. The ship name is returned.
 ************************************************************
 */

char* AddShip (BGs, i, s, ShipString)
BGInfo	BGs[MAXBGS];
int	i, s;
char*	ShipString;
{
	int			Index;
	F76ShipTypeInfo		F76Table[MAXSHIPTYPES];
	char*			MyString;
	char*			HullString;
	char*			NameString;

	MyString = strdup(ShipString);

	HullString = strtok(ShipString," ");

	NameString  = strstr(MyString,"USS");

	strcpy(BGs[i].Ships[s].Name, NameString);

	strcpy(BGs[i].Ships[s].Hull, HullString);

	Index = TypeShip(HullString);

	if (Index == 3){
		BGs[i].Ships[s].TypeCombatant = 1;
	}
```

```
        GetF76Table(F76Table);

        BGs[i].Ships[s].F76.Capacity            = F76Table[Index].F76Capacity;
        BGs[i].Ships[s].F76.OnHand              = BGs[i].Ships[s].F76.Capacity;
        BGs[i].Ships[s].F76.EstOnHand           = BGs[i].Ships[s].F76.Capacity;
        BGs[i].Ships[s].F76.Dtg                 = current_time();

        BGs[i].Ships[s].F76.ReceiveRate         = F76Table[Index].F76Receive;
        BGs[i].Ships[s].F76.TransferRate        = F76Table[Index].F76Transfer;

        BGs[i].Ships[s].F76.Coef[0]             = F76Table[Index].Coef[0];
        BGs[i].Ships[s].F76.Coef[1]             = F76Table[Index].Coef[1];
        BGs[i].Ships[s].F76.Coef[2]             = F76Table[Index].Coef[2];

        BGs[i].Ships[s].Location.Dtg            = current_time();
        BGs[i].Ships[s].Location.Speed          = 0.0;
        BGs[i].Ships[s].Location.MaxSpeed       = 0.0;
        BGs[i].Ships[s].Location.Latitude       = 0.0;
        BGs[i].Ships[s].Location.Longitude      = 0.0;

        return((char*)ShipString);

}

/************************************************************
 * Given the battle group array, this function gets the battle group data
 * from the battle group data ASCII text file. Returns the next available
 * battle group index.
 ************************************************************
 */

int GetBGs(BGs)
BGInfo BGs[MAXBGS];
{
        FILE* DataFile;
        int   i = 0;

        system   ("cp $BGLCSS/data/BGData.dat $BGLCSS/data/BGData.dat.bak");

        DataFile = fopen(BGDATA, "r");

        while (!feof(DataFile)) {
                fscanf(DataFile, "%[^\n]\n",     BGs[i].Name);
                fscanf(DataFile, "%[^\n]\n",     BGs[i].Designation);
                fscanf(DataFile, "%f\n",        &BGs[i].Settings.FuelRes);
                fscanf(DataFile, "%f\n",        &BGs[i].Settings.CLFFuelRes);
                fscanf(DataFile, "%f\n",        &BGs[i].Settings.OrdRes);
                fscanf(DataFile, "%f\n",        &BGs[i].Settings.CLFOrdRes);
                fscanf(DataFile, "%f\n",        &BGs[i].Settings.MaxF76);
                fscanf(DataFile, "%f\n",        &BGs[i].Settings.MaxF44);
                fscanf(DataFile, "%f\n",        &BGs[i].Settings.StationSpeed);
                fscanf(DataFile, "%f\n",        &BGs[i].Settings.UnrepSpeed);
                fscanf(DataFile, "%f\n",        &BGs[i].Settings.AcftShipSpeed);
                i++;
        }
        fclose(DataFile);
        return(i);
}

/**********************************************************************/
```

```
/*******************************************************************
 *Author   :    Bernadette C. Brooks
 *Office   :    Computer Science Department
 *              Naval Postgraduate School
 *              Monterey, CA 93943
 *              Phone: (408) 656-2180
 *Project
 *Advisor  :    Dr. C. Thomas Wu
 *              Computer Science Department
 *              Naval Postgraduate School
 *              Monterey, CA 93943
 *              Phone: (408) 656-3391
 *Filename :    BGEventsLib.h
 *Date     :    27 Feb 93
 *Content  :    C manifests, data type definitions, and data
 *              structure definitions
 *Note     :    "global.h" TAE-generated file includes bg.h
 *******************************************************************
 */

#include <stdio.h>
#include <stdlib.h>


/***********************************************************************/

enum ThreatType {
        Low,
        Med,
        High,
        Raid,
        Strike,
        Asw,
        NoThreat

};

typedef enum ThreatType ThreatType;

/***********************************************************************/

enum CalcType {
        Ord,
        F76,
        F44,
        BothFuel,
        All
};

typedef enum CalcType CalcType;
```

```
/************************************************************************/

enum TargetSize {
        Small,
        Medium,
        Large
};

typedef enum TargetSize TargetSize;

/************************************************************************/

enum StrikeType {
        SAirOnly,
        SAirSurface,
        SSurfaceOnly,
        LAirOnly,
        LAirSurface,
        LSurfaceOnly
};

typedef enum StrikeType StrikeType;

/************************************************************************/

enum RaidType {
        SNA,
        SLCM,
        TACAIR
};

typedef enum RaidType RaidType;

/************************************************************************/

enum TacticType {
        ServiceStation,
        DeliveryBoy,
        CircuitRider,
        Vertrep
};

typedef enum TacticType TacticType;

/************************************************************************/

typedef struct {
        OrdName         Name;
        int             Quantity;
}OrdAmounts;

/************************************************************************/

typedef struct {
        int             NumberofWepaons;
        OrdAmounts      WeaponsUse[MAXORD];
}RaidOrd;
```

```
/**********************************************************************/

typedef struct {
        int             DelShip,
                        RecShip1,
                        RecShip2;
        TacticType      Tactic;
        double          Latitude,
                        Longitude;
}UnrepInfo;

/**************************************************************************/

typedef struct {
        int             ShipInvolved;
        double          Course:
        double          Speed;
}DirectionInfo;

/**********************************************************************/

typedef struct {
        int             ShipInvolved;
        int             TotalNumber;
        TargetSize      Size;
        StrikeType      AttackProfile;
        double          Latitude,
                        Longitude;
        OrdAmounts      StrikeOrdUse[MAXORD];
}StrikeInfo;

/**********************************************************************/

typedef struct {
        RaidType        AttackProfile;
        int             Size,
                        ThreatAxis,
                        NumberofShips;
        int             ShipsInvolved[MAXSHIPS];
        RaidOrd         ShipUse[MAXSHIPS];
}RaidInfo;

/**********************************************************************/

typedef struct {
        int             ShipInvolved;
        OrdAmounts      WeaponsData[MAXORD];
}ASWInfo;

/**********************************************************************/

typedef struct {
        OrdName         Name[MAXORD];
        int             Range[MAXORD];
}WeaponRangeInfo;
```

```
/*****************************************************************/

enum BGEventType {
        BGCourseSpeed,
        ASWLevel,
        AAWLevel,
        SetStation,
        ShipCourseSpeed,
        Unrep,
        Consol,
        FuelTransfer,
        OrdTransfer,
        RaidEvent,
        StrikeEvent,
        ASWProsecute,
        ResumeBGCourseSpeed,
        Other
};

typedef enum BGEventType BGEventType;

/*******************************************************************/

enum PredictType {
        Orphan,
        Child,
        Parent,
        Interval
};

typedef enum PredictType PredictType;

/*******************************************************************/

typedef struct {
        float     F76PercentCap[3],
                  F44PercentCap[3];
}PercentCapInfo;

/*******************************************************************/

typedef struct {
        PercentCapInfo Ships[MAXBGSHIPS];
}BGResultInfo;

/*******************************************************************/
struct BGHeader{
        struct BGHeader        *Prev,
                               *Next;
        BGEventType            EType;
        int                    Index;
        int                    DTG;
        char                   Date[DTGLENGTH];
        char                   Title[MAXLENGTH];
        float                  Course;
        float                  Speed;
};

typedef struct BGHeader BGHEADER;
```

```
/*****************************************************************************/

struct BGEvent {
        struct BGEvent          *Prev,
                                *Next;
        int                     DTG,
                                Index,
                                Created,
                                PredictInterval;
        BGEventType             EType;
        PredictType             PType;
        CalcType                CType;
        ThreatType              TType;
        UnrepInfo               Unrep;
        DirectionInfo           Direction;
        StrikeInfo              Strike;
        RaidInfo                Raid;
        ASWInfo                 ASW;
};

typedef struct BGEvent BGEVENT;

/*****************************************************************************/

struct RelationType {
        struct RelationType     *Prev,
                                *Next;
        int                     Created;
        BGEVENT                 *Child1,
                                *Child2,
                                *Child3,
                                *Child4,
                                *Child5;
};

typedef struct RelationType RELATION;

/*****************************************************************************/
```

```
/*****************************************************************
 *Author    :     Bernadette C. Brooks
 *Office    :     Computer Science Department
 *                Naval Postgraduate School
 *                Monterey, CA 93943
 *                Phone: (408) 656-2180
 *Project
 *Advisor   :     Dr. C. Thomas Wu
 *                Computer Science Department
 *                Naval Postgraduate School
 *                Monterey, CA 93943
 *                Phone: (408) 656-3391
 *Filename  :     BGEventsLib.c
 *Date      :     27 Feb 93
 *Content   :     Bodies of user-defined functions to represent battle
 *                groups and ships. C manifests contained in BGEventsLib.h
 *Note      :     "global.h" TAE-generated file includes bg.h
 *****************************************************************
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/*TAE system include files*/
#include     "taeconf.inp"
#include     "wptinc.inp"
#include     "symtab.inc"
#include     "parblk.inc"
#include     "terminc.inc"
#include     "global.h"

#define YES     1
#define NO      0

/*********************************************************
 * This function saves the battle group events list given an index to the
 * appropriate battle group in the array and a pointer to the head of the
 * battle group event list. It returns the pointer to the head of the
 * list.
 *********************************************************
 */

BGEVENT* SaveBGEvents(BGIndex, BGEventList)
int            BGIndex;
BGEVENT*       BGEventList;
{

        char            FileName[100];
        char            Suffix[MAXNAME];
        FILE*           DataFile;
        BGEVENT*        Current; /*pointer used to traverse the doubly-linked list*/

    /*Based on BG Index, create appropriate file name for BG events. Ex:
    For events in BGs[0], filename is /h/bglcss/scripts/data/Events0.dat*/

    strcpy(FileName, EVENTSDATA);
    strcpy(Suffix,IntToString(BGIndex));
    strcat(Suffix, ".dat");
    strcat(FileName, Suffix);
```

```
    DataFile = fopen (FileName, "w");

    Current =  BGEventList;
    /*assign the head pointer to Current to save the original position of the
     head pointer*/

    while (Current != NULL){
        fprintf(DataFile, "%d\n", Current->DTG);
        fprintf(DataFile, "%d\n", Current->Created);
        fprintf(DataFile, "%d\n", Current->EType);
        fprintf(DataFile, "%d\n", Current->CType);
        fprintf(DataFile, "%d\n", Current->TType);
            if (Current->EType == BGCourseSpeed){
                        fprintf(DataFile, "%lf\n", Current->Direction.Course);
                        fprintf(DataFile, "%lf\n", Current->Direction.Speed);
            }
        if (Current->EType == Unrep){
                        fprintf(DataFile,"%d\n", Current->Unrep.DelShip);
                        fprintf(DataFile,"%d\n", Current->Unrep.RecShip1);
                        fprintf(DataFile,"%d\n", Current->Unrep.RecShip2);
                        fprintf(DataFile,"%d\n", Current->Unrep.Tactic);
                        fprintf(DataFile,"%lf\n", Current->Unrep.Latitude);
                        fprintf(DataFile,"%lf\n", Current->Unrep.Longitude);
                        fprintf(DataFile,"%d\n", Current->Direction.ShipInvolved);
                        fprintf(DataFile,"%lf\n",Current->Direction.Course);
                        fprintf(DataFile,"%lf\n",Current->Direction.Speed);
        }
        Current = Current->Next;
    }
    fclose(DataFile);
    return(BGEventList);
}

/*********************************************************
 * This function reads the battle group event list data from the appropri-
 * ate ASCII text file given the index to the battle group array. It
 * returns a pointer to the head of the battle group event list.
 *********************************************************
 */

BGEVENT* GetBGEvents(BGIndex)
int     BGIndex;
{
        char            FileName[100];
        char            Suffix[MAXNAME];
        FILE*           DataFile;
        BGEVENT*        Current;
        BGEVENT*        Head = NULL;
        BGEVENT*        Temp;

        /*allocate memory for the Head event struct*/
        Head = (BGEVENT* ) malloc(sizeof (struct BGEvent ));

        /*make sure these data are correctly initialized*/
        Head->Next = NULL;
        Head->Prev = NULL;
        Head->DTG = 0;

        /*Based on BG Index, create appropriate file name for BG events. Ex:
        For events in BGs[0], filename is /h/bglcss/scripts/data/Events0.dat*/
```

```
strcpy(FileName, EVENTSDATA);
strcpy(Suffix,IntToString(BGIndex));
strcat(Suffix, ".dat");
strcat(FileName, Suffix);

if ((DataFile = fopen (FileName, "r")) != NULL){

 /*get the Head data first*/
 fscanf(DataFile, "%d\n", &Head->DTG);
 fscanf(DataFile, "%d\n", &Head->Created);
 fscanf(DataFile, "%d\n", &Head->EType);
 fscanf(DataFile, "%d\n", &Head->CType);
 fscanf(DataFile, "%d\n", &Head->TType);
 if (!feof(DataFile)){
        if (Head->EType == BGCourseSpeed){
                fscanf(DataFile, "%lf\n", &Head->Direction.Course);
                fscanf(DataFile, "%lf\n", &Head->Direction.Speed);
        }
        if (Head->EType == Unrep){
                fscanf(DataFile,"%d\n", &Head->Unrep.DelShip);
                fscanf(DataFile,"%d\n", &Head->Unrep.RecShip1);
                fscanf(DataFile,"%d\n", &Head->Unrep.RecShip2);
                fscanf(DataFile,"%d\n", &Head->Unrep.Tactic);
                fscanf(DataFile,"%lf\n",&Head->Unrep.Latitude);
                fscanf(DataFile,"%lf\n",&Head->Unrep.Longitude);
                fscanf(DataFile,"%d\n",&Head->Direction.ShipInvolved);
                fscanf(DataFile,"%lf\n",&Head->Direction.Course);
                fscanf(DataFile,"%lf\n",&Head->Direction.Speed);
        }
        /*other event cases to be implemented*/
 }

 /*assign Head to Current to save original Head pointer position*/
 Current = Head;

 while (!feof(DataFile)){

        /*make a new event node for each new data read*/
        Temp = (BGEVENT* ) malloc(sizeof (struct BGEvent ));
        Temp->Next = NULL;

        /*attach new node to Current*/
        Temp->Prev = Current;
        Current->Next = Temp;

        /*move to the new node*/
        Current = Current->Next;

        fscanf(DataFile, "%d\n", &Current->DTG);
        fscanf(DataFile, "%d\n", &Current->Created);
        fscanf(DataFile, "%d\n", &Current->EType);
        fscanf(DataFile, "%d\n", &Current->CType);
        fscanf(DataFile, "%d\n", &Current->TType);
        if (!feof(DataFile)){
                if (Current->EType == BGCourseSpeed){
                        fscanf(DataFile,"%lf\n",&Current->Direction.Course);
                        fscanf(DataFile,"%lf\n",&Current->Direction.Speed);
                }
                if (Current->EType == Unrep){
                        fscanf(DataFile,"%d\n", &Current->Unrep.DelShip);
                        fscanf(DataFile,"%d\n", &Current->Unrep.RecShip1);
```

```
                        fscanf(DataFile,"%d\n", &Current->Unrep.RecShip2);
                        fscanf(DataFile,"%d\n", &Current->Unrep.Tactic);
                        fscanf(DataFile,"%lf\n", &Current->Unrep.Latitude);
                        fscanf(DataFile,"%lf\n", &Current->Unrep.Longitude);
                    fscanf(DataFile,"%d\n",
                    &Current->Direction.ShipInvolved);
                    fscanf(DataFile,"%lf\n",&Current->Direction.Course);
                    fscanf(DataFile,"%lf\n",&Current->Direction.Speed);
                    }
                    /*other event cases to be implemented*/
                }
            }
            fclose(DataFile);
        }
        return(Head);
}

/*************************************************************
 * Given a pointer to the newly made event node and values passed from
 * the Unrep panel, return a completed unrep event node to be added to
 * the event list. After calling this function, it is necessary to make
 * the following calls to these functions (yet to be implemented):
 * GetRelations,
 * MakeRelation,
 * InsertRelation,
 * UnrepCalculations,
 * MakeChild,
 * InsertBGEvent,
 * SaveRelations,
 * SaveBGEvents.
 *************************************************************
 */

BGEVENT* MakeUnrep(BGEvent, Delivery, Rec1, Rec2, Tactic, Lat, Long)
BGEVENT*        BGEvent;
int             Delivery, Rec1, Rec2, Tactic;
double          Lat, Long;
{
        if (BGEvent->EType == 5){
                BGEvent->Unrep.DelShip          = Delivery;
                BGEvent->Unrep.RecShip1         = Rec1;
                BGEvent->Unrep.RecShip2         = Rec2;
                BGEvent->Unrep.Tactic           = Tactic;
                BGEvent->Unrep.Latitude         = Lat;
                BGEvent->Unrep.Longitude        = Long;
        } else {
                ;
        }
        return(BGEvent);
}
```

```
/**********************************************************
 * This function creates the header to be displayed in the event list
 * panel to the user. Given the event parameters, return a header node.
 **********************************************************
 */

BGHEADER* MakeBGHeader(EventType, EventTime, EventCourse, EventSpeed)

/*Function is incomplete. It is designed to handle past of the battle course and
  speed change event. Need to add remaining parameters to make function
  generic to all events.*/

BGEventType             EventType;
char                    EventTime[DTGLENGTH];
float                   EventCourse;
float                   EventSpeed;
{
        BGHEADER*BGHeader;

        BGHeader = (BGHEADER* ) malloc(sizeof (struct BGHeader ));

        BGHeader->Prev = NULL;

        BGHeader->Next = NULL;

        if (EventType == BGCourseSpeed){
                strcpy(BGHeader->Date, EventTime);
                strcpy(BGHeader->Title, "BG course to");

                BGHeader->Course = EventCourse;
                BGHeader->Speed  = EventSpeed;
                BGHeader->EType  = EventType;
        }
        /*other events to be implemented*/

        return(BGHeader);
}

/**********************************************************
 * Given the appropriate index to the battle group array, this function
 * gets the battle group header information from the appropriate ASCII
 * text file and returns a pointer to the head of the battle group header
 * list. Similar in algorithm to GetBGEvents.
 **********************************************************
 */

BGHEADER* GetBGHeaders(BGIndex)
int     BGIndex;
{
        int             i = 0;
        FILE*           DataFile;
        char            Suffix[MAXNAME];
        char            FileName[100];
        BGHEADER*       Current;
        BGHEADER*       Head = NULL;
        BGHEADER*       Temp;

        Head            = (BGHEADER* ) malloc(sizeof (struct BGHeader ));
        Head->Next      = NULL;
        Head->Prev      = NULL;
        Head->DTG       = 0;
```

```
        /*Based on BG Index, create appropriate file name for BG events. Ex:
        For events in BGs[0], filename is /h/bglcss/scripts/data/Events0.dat*/

        strcpy(FileName, HEADERSDATA);
        strcpy(Suffix,IntToString(BGIndex));
        strcat(Suffix, ".dat");
        strcat(FileName, Suffix);

        if ((DataFile = fopen (FileName, "r")) != NULL){
                Head->Index = i;
                fscanf(DataFile, "%d\n", &Head->EType);
                fscanf(DataFile, "%d\n", &Head->DTG);
                fscanf(DataFile, "%s\n",   Head->Date);
                fscanf(DataFile, "%s\n",   Head->Title);
                if (Head->EType ==  BGCourseSpeed){
                        fscanf(DataFile,"lf\n", &Head->Course);
                        fscanf(DataFile,"lf\n", &Head->Speed);
                }

                /*other events to be implemented*/

                Current = Head;

                while (!feof(DataFile)){
                  Temp = (BGHEADER* ) malloc(sizeof (struct BGHeader ));
                  Temp->Next              = NULL;
                  Temp->Prev              = Current;
                  Current->Next           = Temp;
                  Current                 = Current->Next;

                  fscanf(DataFile, "%d\n", &Head->EType);
                  fscanf(DataFile, "%d\n", &Current->DTG);
                  fscanf(DataFile, "%s\n",  Current->Date);
                  fscanf(DataFile, "%s\n",  Current->Title);

                  if (Current->EType ==  BGCourseSpeed){

                        fscanf(DataFile,"lf\n", &Current->Course);
                        fscanf(DataFile,"lf\n", &Current->Speed);
                  }

                  /*other events to be implemented*/
          }
          fclose(DataFile);
        }
        return(Head);
}
```

```
/*************************************************************
 * This function inserts the newly created BGHeader into the Header list
 * given a pointer to the head of the header list and a pointer to the
 * newly created BGHeader. It returns a pointer to the head of the header
 * list.
 *************************************************************
 */

BGHEADER* InsertBGHeader(Head, BGHeader)
BGHEADER* Head;
BGHEADER* BGHeader;
{
    int              SpotFound;
    BGHEADER*        Current;

    SpotFound        = NO;
    Current          = Head;

    if (Head->DTG == 0) {
        Head = BGHeader;

    } else if ((BGHeader->DTG > Current->DTG)
    && (Current->Next == NULL)){

        Current->Next    = BGHeader;
        BGHeader->Prev = Current;

    } else if ((BGHeader->DTG > Current->DTG) && (Current->Next != NULL)){

        while ((Current->Next != NULL) && (SpotFound == NO)) {

            if (BGHeader->DTG <= Current->DTG) {
                SpotFound = YES;
            } else {
                Current = Current->Next;
            }
        }

        if (Current->Next == NULL && BGHeader->DTG >= Current->DTG){
            Current->Next            = BGHeader;
            BGHeader->Prev           = Current;
        } else {
            BGHeader->Next                    = Current;
            BGHeader->Prev                    = Current->Prev;
            Current->Prev                     = BGHeader;
            BGHeader->Prev->Next              = BGHeader;
        }

    } else if (BGHeader->DTG <= Current->DTG) {
        Current->Prev    = BGHeader;
        BGHeader->Next = Current;
        Head              = BGHeader;
    }

    return (Head);
}
```

```
/*****************************************************************
 * Given an index to the battle group array and a pointer to head of
 * the battle group header list, this function saves the header list data
 * to the appropriate ASCII text file. Returns a pointer to the head of
 * the header list.
 *****************************************************************
 */

BGHEADER* SaveBGHeaders(BGIndex, BGHeaderList)
int             BGIndex;
BGHEADER*       BGHeaderList;
{
    char                FileName[100];
    char                Suffix[MAXNAME];
    FILE*               DataFile;
    BGHEADER*           Current;

    /*Based on BG Index, create appropriate file name for BG events. Ex:
    For events in BGs[0], filename is /h/bglcss/scripts/data/Events0.dat*/

    strcpy(FileName, HEADERSDATA);

    strcpy(Suffix,IntToString(BGIndex));

    strcat(Suffix, ".dat");

    strcat(FileName, Suffix);

    DataFile = fopen (FileName, "w");

    Current = BGHeaderList;

    while (Current != NULL){
                fprintf(DataFile, "%d\n", Current->EType);
                fprintf(DataFile, "%d\n", Current->DTG);
                fprintf(DataFile, "%s\n", Current->Date);
                fprintf(DataFile, "%s\n", Current->Title);

                if (Current->EType == BGCourseSpeed){
                        fprintf(DataFile,"%lf\n", Current->Course);
                        fprintf(DataFile,"%lf\n", Current->Speed);
                }

                /*other events to be implemented*/

                Current = Current->Next;
    }

    fclose(DataFile);
    return(BGHeaderList);
}
```

```
/*********************************************************
 * Given the information from an event panel, this function makesa battle
 * event node and returns a pointer to it. This function is currently
 * designed to handle only a battle group course and speed change event.
 * It needs to be extended to handle the remaining events.
 *********************************************************
 */

BGEVENT* MakeBGEvent(EventCreated,  EventDTG, EventType,  EventPredictType,
                     EventCalc,     EventThreat,          EventShip,
                     EventCourse,   EventSpeed)

int             EventCreated,
                EventDTG,
                EventShip;
BGEventType     EventType;
PredictType     EventPredictType;
CalcType        EventCalc;
ThreatType      EventThreat;
float           EventCourse;
float           EventSpeed;
{
        BGEVENT*        BGEvent;

        BGEvent = (BGEVENT* ) malloc(sizeof (struct BGEvent ));

        BGEvent->Created                = EventCreated;
        BGEvent->Prev                   = NULL;
        BGEvent->Next                   = NULL;
        BGEvent->DTG                    = EventDTG;
        BGEvent->EType                  = EventType;
        BGEvent->PType                  = EventPredictType;
        BGEvent->CType                  = EventCalc;
        BGEvent->TType                  = EventThreat;

        if (BGEvent->EType == BGCourseSpeed){
                BGEvent->Direction.Course       = EventCourse;
                BGEvent->Direction.Speed        = EventSpeed;
        }

        /*other events to be implmented*/

        return (BGEvent);
}
```

```
/*****************************************************************
 * This functions makes a related-event node used to connect related
 * events together such as as unrep with its associated stationing events.
 * The parameter passed is the integer value of the creation time for the
 * parent event (such as the unrep event). No more than 5 associated
 * events are allowed by this function. Returns a pointer to the newly
 * created relation node.
 *****************************************************************
 */

RELATION* MakeRelation(RelationCreated)
int         RelationCreated;
{
        RELATION*     Relation;

        Relation = (RELATION* ) malloc(sizeof (struct RelationType ));

        Relation->Created               = RelationCreated;
        Relation->Prev                  = NULL;
        Relation->Next                  = NULL;
        Relation->Child1                = NULL;
        Relation->Child2                = NULL;
        Relation->Child3                = NULL;
        Relation->Child4                = NULL;
        Relation->Child5                = NULL;

        return (Relation);
}

/*****************************************************************
 * This function makes a child event by first calling MakeBGEvent and
 * attaching the child to the appropriate relation node.  After a call
 * to this function is made, need to call, for instance, UnrepCalculations
 * and make the appropriate assignments to the event node. Function
 * returns a pointer to the newly made child.
 *****************************************************************
 */

BGEVENT* MakeChild(Relation, ParentCreationTime)
RELATION*       Relation;
int             ParentCreationTime;
{
        BGEVENT*            Child;
        int                 Now;

        /*now should be assigned the integer value of current system time*/

        Child = MakeBGEvent
        (Now,ParentCreationTime,Other,Child,NoThreat,100,0.0,0.0);

        if (Relation->Child1 == NULL) {
                Relation->Child1 = Child;

        } else if (Relation->Child2 == NULL) {
                Relation->Child2 = Child;

        } else if (Relation->Child3 == NULL) {
                Relation->Child3 = Child;

        } else if (Relation->Child4 == NULL) {
                Relation->Child4 = Child;
```

```
        } else if (Relation->Child5 == NULL) {
                Relation->Child5 = Child;
        }

        return (Child);
}

/*********************************************************
 * This function takes a pointer to the head of the battle group event
 * list and a pointer to the newly created battle group event and inserts
 * the new event into the list based on chronological dat time group of
 * the events. Returns a pointer to the head of the batlle group event
 * list.
 *********************************************************
 */

BGEVENT* InsertBGEvent(Head, BGEvent)
BGEVENT* Head;
BGEVENT* BGEvent;
{
        int             SpotFound;
        BGEVENT*        Current;

        SpotFound       = NO;
        Current         = Head;

        if (Head->DTG == 0) {
                Head = BGEvent;

        } else if ((BGEvent->DTG > Current->DTG)
        && (Current->Next == NULL)) {
                Current->Next           = BGEvent;
                BGEvent->Prev           = Current;

        } else if ((BGEvent->DTG > Current->DTG)
        && (Current->Next != NULL)) {

                while ((Current->Next != NULL) && (SpotFound == NO)) {

                        if (BGEvent->DTG <= Current->DTG) {
                                SpotFound = YES;
                        } else {
                                Current = Current->Next;
                        }
                }

                if (Current->Next == NULL
                && BGEvent->DTG >= Current->DTG) {
                        Current->Next           = BGEvent;
                        BGEvent->Prev           = Current;
                } else {
                        BGEvent->Next           = Current;
                        BGEvent->Prev           = Current->Prev;
                        Current->Prev           = BGEvent;
                        BGEvent->Prev->Next     = BGEvent;
                }

        } else if (BGEvent->DTG <= Current->DTG) {
                Current->Prev           = BGEvent;
                BGEvent->Next           = Current;
```

```
		Head				= BGEvent;
	}

	/* if BGEvent is actually an interval node, copy course and speed
	 * from Prev */
	if ((BGEvent->EType == Other) && (BGEvent->Prev != NULL)) {
		BGEvent->Direction.Course = BGEvent->Prev->Direction.Course;
		BGEvent->Direction.Speed  = BGEvent->Prev->Direction.Speed;
	}

	return (Head);
}

/***********************************************************
 * This function's basic algorithm is virutally the same to InsertBGEvent
 * except for the final if-statement assignments and the data type
 * involved.
 ***********************************************************
 */

RELATION* InsertRelation(Head, Relation)
RELATION*	Head;
RELATION*	Relation;
{
	int		SpotFound;
	RELATION*	Current;

	SpotFound	= NO;
	Current		= Head;

	if (Head == NULL) {
		Head = Relation;

	}else if((Relation->Created > Current->Created)
	&& (Current->Next == NULL)) {

		Current->Next  = Relation;
		Relation->Prev = Current;

	} else if ((Relation->Created > Current->Created)
	&& (Current->Next != NULL)) {

		while ((Current->Next != NULL) && (SpotFound == NO)) {
			if (Relation->Created <= Current->Created) {
				SpotFound = YES;
			} else {
				Current = Current->Next;
			}
		}
		if (Current->Next == NULL && Relation->Created >=
		   Current->Created) {
			Current->Next  = Relation;
			Relation->Prev = Current;
		} else {
			Relation->Next			= Current;
			Relation->Prev			= Current->Prev;
			Current->Prev			= Relation;
			Relation->Prev->Next		= Relation;
		}
	} else if (Relation->Created <= Current->Created) {
		Current->Prev			= Relation;
```

```
                Relation->Next                  = Current;
                Head                            = Relation;
        }
        return (Head);
}

/************************************************************
 * This functions finds the Parent event with its unique time stamp. If
 * If the parent doesn't exist, then it finds the orphan event and returns
 * a pointer to the event found.
 ************************************************************
 */

BGEVENT* GetParent(Head, Creation)
BGEVENT*        Head;
int             Creation;
{
        BGEVENT*        Current;
        BGEVENT*        OrphanEvent;
        int             SpotFound;

        SpotFound               = NO;
        OrphanEvent             = NULL;
        Current                 = Head;

        /* if Head of list is orphan and there are events to scan */
        if (Current->Next != NULL) {

                while ((Current->Next != NULL) && (SpotFound ==
                  NO)) {

                        if (Current->Created != Creation) {
                                Current = Current->Next;
                        } else if ((Current->Created == Creation) &&
                          (Current->EType == 2)) {
                                Current = Current->Next;
                        } else {
                                SpotFound = YES;
                        }
                }

                return (Current);

        } else if (Current->Next == NULL) {
                return (Current);

        } else if (Current == NULL) {
                return (OrphanEvent);
        }
}

/************************************************************
 * This functions finds the Parent event with its unique time stamp. If
 * If the parent doesn't exist, then it finds the orphan event and returns
 * a pointer to the event found.
 ************************************************************
 */

RELATION* GetRelation(Head, Creation)
RELATION*       Head;
int             Creation;
```

```
{
        RELATION*       Current;
        RELATION*       OrphanEvent;
        int             SpotFound;

        OrphanEvent             = NULL;
        Current                 = Head;
        SpotFound               = NO;

        /* if Head of list is orphan and there are events to scan */
        if (Current->Next != NULL) {

                while ((Current->Next != NULL) && (SpotFound == NO)) {

                        if (Current->Created != Creation) {

                                Current = Current->Next;
                        } else {
                                SpotFound = YES;
                        }
                }

                return (Current);

        } else if (Current->Next == NULL) {
                return (Current);

        } else if (Current == NULL) {
                return (OrphanEvent);
        }
}

/*************************************************************
 * Given a pointer to the head of the battle group event list and the
 * event to be deleted, this function deletes the event. Before calling
 * this function with the Parent Event node pointer, need to call
 * the DeleteChildren function to delete the associated children.
 *************************************************************
 */

BGEVENT *DeleteBGEvent(Head, BGEvent)
BGEVENT *Head;
BGEVENT *BGEvent;
{
        /* delete tail */
        if ((BGEvent->Next == NULL) && (BGEvent->Prev != NULL)) {
                BGEvent->Prev->Next = NULL;
                BGEvent->Prev = NULL;
                BGEvent->Next = NULL;

        /* delete Head */
        } else if ((BGEvent->Prev == NULL) && (BGEvent->Next != NULL)) {
                Head     = BGEvent->Next;
                Head->Prev = NULL;

        /* delete middle */
        } else if ((BGEvent->Next != NULL) && (BGEvent->Prev != NULL)) {
                BGEvent->Prev->Next = BGEvent->Next;
                BGEvent->Next->Prev = BGEvent->Prev;

        /* delete one-node list */
```

```
	} else {
		Head = NULL;
		BGEvent = NULL;

		free(Head);
	}

	free(BGEvent);

	return (Head);
}

/*************************************************************
 * This function makes repeated calls to DeleteBGEvent in order to delete
 * all of the children of the Parent event. Returns the head of the
 * battle group event list.
 *************************************************************
 */

BGEVENT* DeleteChildren(Head, Parent)
BGEVENT* Head;
RELATION*Parent;
{
	if (Parent->Child1 != NULL) {
		Head = DeleteBGEvent(Head, Parent->Child1);
	}
	if (Parent->Child2 != NULL) {
		Head = DeleteBGEvent(Head, Parent->Child2);
	}
	if (Parent->Child3 != NULL) {
		Head = DeleteBGEvent(Head, Parent->Child3);
	}
	if (Parent->Child4 != NULL) {
		Head = DeleteBGEvent(Head, Parent->Child4);
	}
	if (Parent->Child5 != NULL) {
		Head = DeleteBGEvent(Head, Parent->Child5);
	}
	return (Head);
}

/*************************************************************/
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      global.h *** */
/* *** Generated:  Dec  2 16:01:50 1992 *** */
/* ***********************************************************************
 * PURPOSE:
 * This global header file is automatically "#include"d in each panel
 * file.  You can insert references to global variables here.
 * ***********************************************************************
 */

#ifndef I_GLOBAL                    /* prevent double include */
#define I_GLOBAL       0

/*     macros for access to parameter values
 *
 * These macros obtain parameter values given the name of
 * a Vm object and the name string of the parameter.
 * The Vm objects are created by the Initialize_All_Panels
 * function for a resource file.
 *
 * Reference scalar parameters as follows:
 *
 *     StringParm(myPanelTarget, "s")   -- string pointer
 *     IntParm(myPanelTarget, "i")      -- integer value
 *     RealParm(myPanelTarget, "r")     -- real value
 *
 *  For vector parameters, do the following:
 *
 *     TAEINT  *ival;
 *     ival = &IntParm(myPanelTarget, "i");
 *     printf ("%d %d %d", ival[0], ival[1], ival[2]);
 *
 */

#include "bg.h"

#include "BGEventsLib.h"

struct VARIABLE *Vm_Find ();

#define StringParm(vmId, name)   (SVAL(*Vm_Find(vmId, name),0))
#define IntParm(vmId, name)      (IVAL(*Vm_Find(vmId, name), 0))
#define RealParm(vmId, name)     (RVAL(*Vm_Find(vmId, name), 0))

/*     Dispatch Table typedef        */

typedef VOID (*FUNCTION_PTR) ();
typedef struct DISPATCH
    {
    TEXT            *parmName;
    FUNCTION_PTR       eventFunction;
    } ;

#define EVENT_HANDLER static VOID          /* a flag for documentation */

/*     Display Id for use by direct Xlib calls: */

extern Display  *Default_Display;

#define SET_APPLICATION_DONE \
    { \
```

```
    extern BOOL Application_Done; \
    Application_Done = TRUE; \
    }

#endif
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_WptHelp.c *** */
/* *** Generated:  Jan 19 14:53:14 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * This file encapsulates the TAE Plus panel:  WptHelp
 * These routines enable panel initialization, creation, and destruction.
 * Access to these routines from other files is enabled by inserting
 * '#include "pan_WptHelp.h"'.  For more advanced manipulation of the panel
 * using the TAE routines, the panel's Id, Target, and View are provided.
 *
 * For the panel items:
 *   (NO EVENT GENERATING ITEMS IN THIS PANEL)
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#include      "taeconf.inp"
#include      "wptinc.inp"
#include      "global.h"          /* Application globals */
#include      "pan_WptHelp.h"

/* One "include" for each connected panel */

Id WptHelpTarget, WptHelpView, WptHelpId;
/* WptHelpDispatch is defined at the end of this file */

/* ************************************************************************
 * Initialize the view and target of this panel.
 */
FUNCTION VOID WptHelp_Initialize_Panel (vmCollection)
   Id vmCollection;
   {
   Id  Co_Find ();

   WptHelpView = Co_Find (vmCollection, "WptHelp_v");
   WptHelpTarget = Co_Find (vmCollection, "WptHelp_t");
   }

/* ************************************************************************
 * Create the panel object and display it on the screen.
 */
FUNCTION VOID WptHelp_Create_Panel (relativeWindow, flags)
   Window     relativeWindow;
   COUNT      flags;
   {
   if (WptHelpId)
      printf ("Panel (WptHelp) is already displayed.\n");
   else
      WptHelpId = Wpt_NewPanel (Default_Display, WptHelpTarget, WptHelpView,
                        relativeWindow, WptHelpDispatch, flags);
   }

/* ************************************************************************
 * Erases a panel from the screen and de-allocate the associated panel
 * object.
 */
FUNCTION VOID WptHelp_Destroy_Panel ()
   {
```

```
    Wpt_PanelErase(WptHelpId);
    WptHelpId=0;
    }

/* ***********************************************************************
 * Connect to this panel.  Create it or change it's state.
 */
FUNCTION VOID WptHelp_Connect_Panel (relativeWindow, flags)
    Window     relativeWindow;
    COUNT      flags;
    {
    if (WptHelpId)
        Wpt_SetPanelState (WptHelpId, flags);
    else
        WptHelp_Create_Panel (relativeWindow, flags);
    }

struct DISPATCH WptHelpDispatch[] = {
    {NULL, NULL}                    /* terminator entry */
};
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_WptHelp.h *** */
/* *** Generated:  Jan 19 14:53:14 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Header file for panel:  WptHelp
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *     The panel's name is changed (not title)
 * For panel:
 *   WptHelp
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_PAN_WptHelp                    /* prevent double include */
#define I_PAN_WptHelp     0

/* Vm objects and panel Id.  */
extern Id WptHelpTarget, WptHelpView, WptHelpId;

/* Dispatch table (global for calls to Wpt_NewPanel) */
extern struct DISPATCH WptHelpDispatch[];

/* Initialize WptHelpTarget and WptHelpView */
extern VOID WptHelp_Initialize_Panel ();

/* Create this panel and display it on the screen */
extern VOID WptHelp_Create_Panel ();

/* Destroy this panel and erase it from the screen */
extern VOID WptHelp_Destroy_Panel ();

/* Connect to this panel.  Create it or change it's state */
extern VOID WptHelp_Connect_Panel ();
#endif
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:     wpthelp.c *** */
/* *** Generated:  Jan 19 14:53:14 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * This the main program of an application generated by the TAE Plus Code
 * Generator.
 *
 * REGENERATED:
 * This file is generated only once.
 *
 * NOTES:
 * To turn this into a real application, do the following:
 *
 * 1. Each panel that has event generating parameters is encapsulated by
 * a separate file, named by concatenating the string "pan_" with the
 * panel name (followed by a ".c"). Each parameter that you have defined
 * to be "event-generating", has an event handler procedure in the
 * appropriate panel file. Each handler has a name that is a
 * concatentation of the parameter name and the string "_Event". Add
 * application-dependent logic to each event handler. (As generated by
 * the WorkBench, each event handler simply logs the occurrence of the
 * event.)
 *
 * 2. To build the program, type "make". If the symbols TAEINC, ....
 * are not defined, the TAE shell (source) scripts $TAE/bin/csh/taesetup
 * will define them.
 *
 * ADDITIONAL NOTES:
 * 1. Each event handler has two arguments: (a) the value vector
 * associated with the parameter and (b) the number of components. Note
 * that for scalar values, we pass the value as if it were a vector with
 * count 1.
 *
 * Though it's unlikely that you are interested in the value of a button
 * event parameter, the values are always passed to the event handler for
 * consistency.
 *
 * 2. You gain access to non-event parameters by calling the Vm package
 * using the targetId Vm objects that are created in
 * Initialize_All_Panels. There are macros defined in global.h to assist
 * in accessing values in Vm objects.
 *
 * To access panel Id, target, and view, of other panels, add an
 * "#include" statement for each appropriate panel header file.
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#include    "taeconf.inp"
#include    "wptinc.inp"
#include    "symtab.inc"
#include    "global.h"          /* Application globals */

Display *Default_Display;
BOOL   Application_Done = FALSE;

main (argc, argv)
```

```
FUNINT    argc;
TEXT      *argv[];

{
WptEvent  wptEvent;           /* event data */
CODE      eventType;

COUNT     termLines, termCols;
CODE      termType;

/* PROGRAMMER NOTE:
 * add similar extern's for each resource file in this application
 */
extern VOID wpthelp_Initialize_All_Panels ();
extern VOID wpthelp_Create_Initial_Panels ();

struct DISPATCH          *dp;      /* working dispatch pointer */
IMPORT struct VARIABLE   *Vm_Find();
struct VARIABLE          *parmv;   /* pointer to event VARIABLE */

/* initialize terminal without clearing screen */
t_pinit (&termLines, &termCols, &termType);

/* permit upper/lowercase file names */
f_force_lower (FALSE);

Default_Display = Wpt_Init (NULL);

/* initialize resource file */
/* PROGRAMMER NOTE:
 * For each resource file in this application, calls to the appropriate
 * Initialize_All_Panels and Create_Initial_Panels must be added.
 */
wpthelp_Initialize_All_Panels ("/h/bglcss/scripts/gui/setup/wpthelp.res");
wpthelp_Create_Initial_Panels ();

/* main event loop */
/* PROGRAMMER NOTE:
 * use SET_APPLICATION_DONE in "quit" event handler to exit loop.
 * (SET_APPLICATION_DONE is defined in global.h)
 */
while (!Application_Done)
  {
  eventType = Wpt_NextEvent (&wptEvent); /* get next WPT event */

  switch (eventType)
    {
  case WPT_PARM_EVENT:

    /* Event has occurred from a Panel Parm. Lookup the event
     * in the dispatch table and call the associated event
     * handler function.
     */

    dp = (struct DISPATCH *) wptEvent.p_userContext;
    for (; (*dp).parmName != NULL; dp++)
      if (s_equal ((*dp).parmName, wptEvent.parmName))
        {
        parmv = Vm_Find (wptEvent.p_dataVm, wptEvent.parmName);
        (*(*dp).eventFunction)
          ((*parmv).v_cvp, (*parmv).v_count);
```

```
            break;
            }
        break;

    case WPT_FILE_EVENT:

        /* PROGRAMMER NOTE:
         * Add code here to handle file events.
         * Use Wpt_AddEvent and Wpt_RemoveEvent to register and remove
         * event sources.
         */
        printf ("No EVENT_HANDLER for event from external source.\n");
        break;

    case WPT_WINDOW_EVENT:

        /* PROGRAMMER NOTE:
         * Add code here to handle window events.
         * WPT_WINDOW_EVENT can be caused by windows which you directly
         * create with X (not TAE panels), or by user acknowledgement
         * of a Wpt_PanelMessage (therefore no default print statement
         * is generated here).
         */
        break;

    case WPT_TIMEOUT_EVENT:

        /* PROGRAMMER NOTE:
         * Add code here to handle timeout events.
         * Use Wpt_SetTimeOut to register timeout events.
         */
        printf ("No EVENT_HANDLER for timeout event.\n");
        break;

    default:

        printf("Unknown WPT Event\n");
        break;
        }
    }                           /* end main event loop */

Wpt_Finish();/* close down all display connections */

/* PROGRAMMER NOTE:
 * Application has ended normally.  Add application specific code to
 * close down your application
 */

}                               /* end main */
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:     wpthelp_creat_init.c *** */
/* *** Generated:  Jan 19 14:53:14 1993 *** */
/* **********************************************************************
 * PURPOSE:
 * Displays all panels in the initial panel set of this resource file
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    A panel is added to the initial panel set
 *    A panel is deleted from the initial panel set
 * For the set of initial panels:
 *    WptHelp
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * **********************************************************************
 */
#include      "taeconf.inp"
#include      "wptinc.inp"
#include      "global.h"          /* Application globals */

/* One include for each panel in initial panel set */
#include      "pan_WptHelp.h"


FUNCTION VOID wpthelp_Create_Initial_Panels ()
   {
   /* Show panels */

   WptHelp_Create_Panel (NULL, WPT_PREFERRED);
   }
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      wpthelp_init_pan.c *** */
/* *** Generated:  Jan 19 14:53:14 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Initialize all panels in the resource file.
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *   A panel is deleted
 *   A new panel is added
 *   A panel's name is changed (not title)
 * For the panels:
 *  WptHelp
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#include     "taeconf.inp"
#include     "wptinc.inp"
#include     "symtab.inc"
#include     "global.h"          /* Application globals */

/* One "include" for each panel in resource file */
#include     "pan_WptHelp.h"


FUNCTION VOID wpthelp_Initialize_All_Panels (resfileSpec)
   TEXT      *resfileSpec;
   {
   extern Id  Co_Find ();
   extern Id  Co_New ();
   Id vmCollection ;

   /* read resource file */
   vmCollection = Co_New (P_ABORT);
   Co_ReadFile (vmCollection, resfileSpec, P_ABORT);

   /* initialize view and target Vm objects for each panel */
   WptHelp_Initialize_Panel (vmCollection);
   }
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      BGSetup.c *** */
/* *** Generated:  Dec  2 16:01:50 1992 *** */
/* ************************************************************************
 * PURPOSE:
 * This the main program of an application generated by the TAE Plus Code
 * Generator.
 *
 * REGENERATED:
 * This file is generated only once.
 *
 * NOTES:
 * To turn this into a real application, do the following:
 *
 * 1. Each panel that has event generating parameters is encapsulated by
 * a separate file, named by concatenating the string "pan_" with the
 * panel name (followed by a ".c"). Each parameter that you have defined
 * to be "event-generating", has an event handler procedure in the
 * appropriate panel file. Each handler has a name that is a
 * concatentation of the parameter name and the string "_Event". Add
 * application-dependent logic to each event handler. (As generated by
 * the WorkBench, each event handler simply logs the occurrence of the
 * event.)
 *
 * 2. To build the program, type "make". If the symbols TAEINC, ....
 * are not defined, the TAE shell (source) scripts $TAE/bin/csh/taesetup
 * will define them.
 *
 * ADDITIONAL NOTES:
 * 1. Each event handler has two arguments: (a) the value vector
 * associated with the parameter and (b) the number of components. Note
 * that for scalar values, we pass the value as if it were a vector with
 * count 1.
 *
 * Though it's unlikely that you are interested in the value of a button
 * event parameter, the values are always passed to the event handler for
 * consistency.
 *
 * 2. You gain access to non-event parameters by calling the Vm package
 * using the targetId Vm objects that are created in
 * Initialize_All_Panels. There are macros defined in global.h to assist
 * in accessing values in Vm objects.
 *
 * To access panel Id, target, and view, of other panels, add an
 * "#include" statement for each appropriate panel header file.
 * ************************************************************************
 */

#include    "taeconf.inp"
#include    "wptinc.inp"
#include    "symtab.inc"
#include    "global.h"          /* Application globals */

Display *Default_Display;
BOOL  Application_Done = FALSE;

main (argc, argv)

   FUNINT    argc;
   TEXT     *argv[];
```

```
{
WptEvent   wptEvent;              /* event data */
CODE       eventType;

COUNT      termLines, termCols;
CODE       termType,/*BERN*/ret;

        /*BERN*/

/* PROGRAMMER NOTE:
 * add similar extern's for each resource file in this application
 */
extern VOID BGSetup_Initialize_All_Panels ();
extern VOID BGSetup_Create_Initial_Panels ();

struct DISPATCH          *dp;      /* working dispatch pointer */
IMPORT struct VARIABLE    *Vm_Find();
struct VARIABLE          *parmv;   /* pointer to event VARIABLE */

/* initialize terminal without clearing screen */
t_pinit (&termLines, &termCols, &termType);

/* permit upper/lowercase file names */
f_force_lower (FALSE);

Default_Display = Wpt_Init (NULL);

/* initialize resource file */
/* PROGRAMMER NOTE:
 * For each resource file in this application, calls to the appropriate
 * Initialize_All_Panels and Create_Initial_Panels must be added.
 */
BGSetup_Initialize_All_Panels ("/h/bglcss/scripts/gui/setup/BGSetup.res");
BGSetup_Create_Initial_Panels ();

/* main event loop */
/* PROGRAMMER NOTE:
 * use SET_APPLICATION_DONE in "quit" event handler to exit loop.
 * (SET_APPLICATION_DONE is defined in global.h)
 */

        /*BERN*/
        ret = Wpt_SetHelpStyle ("wpthelp.res");
        if (ret != SUCCESS)
                printf("Couldn't set help style\n");


while (!Application_Done)
  {
  eventType = Wpt_NextEvent (&wptEvent); /* get next WPT event */

  switch (eventType)
    {
  case WPT_PARM_EVENT:

    /* Event has occurred from a Panel Parm. Lookup the event
     * in the dispatch table and call the associated event
     * handler function.
     */

    dp = (struct DISPATCH *) wptEvent.p_userContext;
```

```
    for (; (*dp).parmName != NULL;  dp++)
      if (s_equal ((*dp).parmName, wptEvent.parmName))
        {
        parmv = Vm_Find (wptEvent.p_dataVm, wptEvent.parmName);
        (*(*dp).eventFunction)
          ((*parmv).v_cvp, (*parmv).v_count);
        break;
        }
    break;

  case WPT_FILE_EVENT:

    /* PROGRAMMER NOTE:
     * Add code here to handle file events.
     * Use Wpt_AddEvent and Wpt_RemoveEvent to register and remove
     * event sources.
     */
    printf ("No EVENT_HANDLER for event from external source.\n");
    break;

  case WPT_WINDOW_EVENT:

    /* PROGRAMMER NOTE:
     * Add code here to handle window events.
     * WPT_WINDOW_EVENT can be caused by windows which you directly
     * create with X (not TAE panels), or by user acknowledgement
     * of a Wpt_PanelMessage (therefore no default print statement
     * is generated here).
     */
    break;

  case WPT_TIMEOUT_EVENT:

    /* PROGRAMMER NOTE:
     * Add code here to handle timeout events.
     * Use Wpt_SetTimeOut to register timeout events.
     */
    printf ("No EVENT_HANDLER for timeout event.\n");
    break;

  default:

    printf("Unknown WPT Event\n");
    break;
    }
  }                       /* end main event loop */

Wpt_Finish();/* close down all display connections */

/* PROGRAMMER NOTE:
 * Application has ended normally.  Add application specific code to
 * close down your application
 */

}

                  /* end main */
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:     BGSetup_creat_init.c *** */
/* *** Generated:  Jan 19 11:14:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Displays all panels in the initial panel set of this resource file
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    A panel is added to the initial panel set
 *    A panel is deleted from the initial panel set
 * For the set of initial panels:
 *   SetUpBGs
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */
#include      "taeconf.inp"
#include      "wptinc.inp"
#include      "global.h"             /* Application globals */

/* One include for each panel in initial panel set */
#include      "pan_SetUpBGs.h"


FUNCTION VOID BGSetup_Create_Initial_Panels ()
  {
  /* Show panels */

  SetUpBGs_Create_Panel (NULL, WPT_PREFERRED);
  }
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:21 EDT 1992 *** */
/* *** File:    BGSetup_init_pan.c *** */
/* *** Generated:  Jan 19 11:14:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Initialize all panels in the resource file.
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    A panel is deleted
 *    A new panel is added
 *    A panel's name is changed (not title)
 * For the panels:
 *   AcftLoad, AirData, BGData,  BGShips, CloseAll, DelBG,   DeleteSh,
 *   DelShip, Dtg,    F44Fuel, F76Fuel, LackData, NewBG,   OrdData,
 *   OrdLoad, OrdSel,  PrintJob, SaveNewB, SelBG,   SetUpBGs, Ship,
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#include     "taeconf.inp"
#include     "wptinc.inp"
#include     "symtab.inc"
#include     "global.h"          /* Application globals */

/* One "include" for each panel in resource file */
#include     "pan_AcftLoad.h"
#include     "pan_AirData.h"
#include     "pan_BGData.h"
#include     "pan_BGShips.h"
#include     "pan_CloseAll.h"
#include     "pan_DelBG.h"
#include     "pan_DeleteSh.h"
#include     "pan_DelShip.h"
#include     "pan_Dtg.h"
#include     "pan_F44Fuel.h"
#include     "pan_F76Fuel.h"
#include     "pan_LackData.h"
#include     "pan_NewBG.h"
#include     "pan_OrdData.h"
#include     "pan_OrdLoad.h"
#include     "pan_OrdSel.h"
#include     "pan_PrintJob.h"
#include     "pan_SaveNewB.h"
#include     "pan_SelBG.h"
#include     "pan_SetUpBGs.h"
#include     "pan_Ship.h"


FUNCTION VOID BGSetup_Initialize_All_Panels (resfileSpec)
  TEXT     *resfileSpec;
  {
  extern Id  Co_Find ();
  extern Id  Co_New ();
  Id vmCollection ;

  /* read resource file */
  vmCollection = Co_New (P_ABORT);
  Co_ReadFile (vmCollection, resfileSpec, P_ABORT);
```

```
/* initialize view and target Vm objects for each panel */
AcftLoad_Initialize_Panel (vmCollection);
AirData_Initialize_Panel (vmCollection);
BGData_Initialize_Panel (vmCollection);
BGShips_Initialize_Panel (vmCollection);
CloseAll_Initialize_Panel (vmCollection);
DelBG_Initialize_Panel (vmCollection);
DeleteSh_Initialize_Panel (vmCollection);
DelShip_Initialize_Panel (vmCollection);
Dtg_Initialize_Panel (vmCollection);
F44Fuel_Initialize_Panel (vmCollection);
F76Fuel_Initialize_Panel (vmCollection);
LackData_Initialize_Panel (vmCollection);
NewBG_Initialize_Panel (vmCollection);
OrdData_Initialize_Panel (vmCollection);
OrdLoad_Initialize_Panel (vmCollection);
OrdSel_Initialize_Panel (vmCollection);
PrintJob_Initialize_Panel (vmCollection);
SaveNewB_Initialize_Panel (vmCollection);
SelBG_Initialize_Panel (vmCollection);
SetUpBGs_Initialize_Panel (vmCollection);
Ship_Initialize_Panel (vmCollection);
}
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:     Imakefile *** */
/* *** Generated:  Nov 27 11:24:06 1992 *** */
/* *********************************************************************
 * PURPOSE:
 * This is the Imakefile of a C application generated by the TAE Plus
 * Code Generator.
 *
 * REGENERATED:
 * This file is generated only once.
 *
 * NOTES:
 * 1. To build your application, type "make". The Makefile generated
 * by the TAE code generator invokes imake using this Imakefile to
 * generate an application specific Makefile.
 *
 * 2. If you change the name of your resource file or application, you
 * will need to either edit this file, or just delete it and regenerate
 * the code.
 *
 * 3. Edit this file to include your application specific source files.
 * *********************************************************************
 */

#define GeneratedApplication

/* PROGRAMMER NOTE:
 * Add a line '#include "Imake.RESFILENAME"' for each resource file in
 * your application.
 */
#include "Imake.BGSetup"


/* PROGRAMMER NOTE:
 * Insert application specific build parameters. These override
 * definitions in the configuration files in $TAE/config.
 */
    CDEBUGFLAGS =
   LDDEBUGFLAGS =
     APP_CFLAGS =
  APP_LOAD_FLAGS =
   APP_LINKLIBS = -L/h/Nauticus/libs -lVids
    APP_DEPLIBS = $(DEPLIBS)
  APP_CINCLUDES = -I$(TAEINC) \
                  -I/h/Nauticus/include/vids/Vids.h\
                  -I/h/bglcss/scripts/gui/setup/bg.h
        PROGRAM = BGSetup

/* PROGRAMMER NOTE:
 * Add $(SRCS_RESFILENAME) and $(OBJS_RESFILENAME) for each resource file
 * in your application.
 */
        GENSRCS = $(PROGRAM).c $(SRCS_BGSetup)
        GENOBJS = $(PROGRAM).o $(OBJS_BGSetup)

/* PROGRAMMER NOTE:
 * Add your application specific srcs and object files (that are not
 * generated by the code generator) here.
 */
        APPSRCS = bg.c
        APPOBJS = bg.o
```

```
/* Macro (defined in TAEmake.tmpl) to generate Makefile targets.
 */
CApplication($(PROGRAM))
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_BGData.c *** */
/* *** Generated:  Jan 19 11:14:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * This file encapsulates the TAE Plus panel:  BGData
 * These routines enable panel initialization, creation, and destruction.
 * Access to these routines from other files is enabled by inserting
 * '#include "pan_BGData.h"'.  For more advanced manipulation of the panel
 * using the TAE routines, the panel's Id, Target, and View are provided.
 * ************************************************************************
 */

#include        "taeconf.inp"
#include        "wptinc.inp"
#include        "global.h"              /* Application globals */
#include        "pan_BGData.h"

/* One "include" for each connected panel */
#include        "pan_BGShips.h"
#include        "pan_DeleteSh.h"
#include        "pan_Ship.h"

/*BERN*/
#include  "pan_SetUpBGs.h"

/*BERN*/
extern      void CancelBG();
extern      int GetBG();
extern      int GetBGs();
extern      void ShowBG();
extern      void ShowBGShips();

Id BGDataTarget, BGDataView, BGDataId;
/* BGDataDispatch is defined at the end of this file */

/* ************************************************************************
 * Initialize the view and target of this panel.
 */
FUNCTION VOID BGData_Initialize_Panel (vmCollection)
   Id vmCollection;
   {
   Id  Co_Find ();

   BGDataView = Co_Find (vmCollection, "BGData_v");
   BGDataTarget = Co_Find (vmCollection, "BGData_t");
   }

/* ************************************************************************
 * Create the panel object and display it on the screen.
 */
FUNCTION VOID BGData_Create_Panel (relativeWindow, flags)
Window     relativeWindow;
COUNT      flags;
{
   /*BERN*/
   int BGIndex;
   BGInfo BGs[MAXBGS];

   if (BGDataId)
           printf ("Panel (BGData) is already displayed.\n");
```

```
  else
          BGDataId = Wpt_NewPanel (Default_Display, BGDataTarget,
          BGDataView, relativeWindow, BGDataDispatch, flags);

          BGIndex = GetBGs(BGs);

          BGIndex = GetBG(StringParm(SetUpBGsTarget,"BGList"), BGIndex);

          ShowBG(BGs, BGIndex, BGDataId);

          ShowBGShips(BGIndex, BGDataId, "BGShips");
}

/* **********************************************************************
 * Erases a panel from the screen and de-allocate the associated panel
 * object.
 */
FUNCTION VOID BGData_Destroy_Panel ()
{
  CancelBG(BGDataId);
  Wpt_PanelErase(BGDataId);
  BGDataId=0;
}

/***********************************************************************
 * Connect to this panel.  Create it or change it's state.
 */
FUNCTION VOID BGData_Connect_Panel (relativeWindow, flags)
  Window     relativeWindow;
  COUNT      flags;
  {
  if (BGDataId)
    Wpt_SetPanelState (BGDataId, flags);
  else
    BGData_Create_Panel (relativeWindow, flags);
  }


/* **********************************************************************
 * Handle event from parameter:  AddShip
 */
EVENT_HANDLER AddShip_Event (value, count)
  TEXT       *value[];              /* string pointers */
  FUNINT     count;                 /* num of values   */
  {
  /* Begin generated code for Connection */
  BGShips_Connect_Panel (NULL, WPT_PREFERRED);
  /* End generated code for Connection */
  }


/* **********************************************************************
 * Handle event from parameter:  Close
 */
EVENT_HANDLER Close_Event (value, count)
  TEXT       *value[];              /* string pointers */
  FUNINT     count;                 /* num of values   */
  {
  /* Begin generated code for Connection */
  BGData_Destroy_Panel ();
  /* End generated code for Connection */
```

```
  }


/* ************************************************************************
 * Handle event from parameter:  Delete
 */
EVENT_HANDLER Delete_Event (value, count)
  TEXT      *value[];              /* string pointers */
  FUNINT     count;                /* num of values   */
  {
  /* Begin generated code for Connection */
  DeleteSh_Connect_Panel (NULL, WPT_PREFERRED);
  /* End generated code for Connection */
  }


/* ************************************************************************
 * Handle event from parameter:  Edit
 */
EVENT_HANDLER Edit_Event (value, count)
  TEXT      *value[];              /* string pointers */
  FUNINT     count;                /* num of values   */
  {
  /* Begin generated code for Connection */
  Ship_Connect_Panel (NULL, WPT_PREFERRED);
  /* End generated code for Connection */
  }


/* ************************************************************************
 * Handle event from parameter:  Help
 */
EVENT_HANDLER Help_Event (value, count)
  TEXT      *value[];              /* string pointers */
  FUNINT     count;                /* num of values   */
  {
  }


/* ************************************************************************
 * Handle event from parameter:  Save
 */
EVENT_HANDLER Save_Event (value, count)
  TEXT      *value[];              /* string pointers */
  FUNINT     count;                /* num of values   */
  {
  }


struct DISPATCH BGDataDispatch[] = {
  {"AddShip", AddShip_Event},
  {"Close", Close_Event},
  {"Delete", Delete_Event},
  {"Edit", Edit_Event},
  {"Help", Help_Event},
  {"Save", Save_Event},
  {NULL, NULL}                     /* terminator entry */
};
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:       pan_BGData.h *** */
/* *** Generated:  Jan 19 13:12:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Header file for panel:  BGData
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    The panel's name is changed (not title)
 * For panel:
 *   BGData
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_PAN_BGData                    /* prevent double include */
#define I_PAN_BGData     0

/* Vm objects and panel Id.  */
extern Id BGDataTarget, BGDataView, BGDataId;

/* Dispatch table (global for calls to Wpt_NewPanel) */
extern struct DISPATCH BGDataDispatch[];

/* Initialize BGDataTarget and BGDataView */
extern VOID BGData_Initialize_Panel ();

/* Create this panel and display it on the screen */
extern VOID BGData_Create_Panel ();

/* Destroy this panel and erase it from the screen */
extern VOID BGData_Destroy_Panel ();

/* Connect to this panel.  Create it or change it's state */
extern VOID BGData_Connect_Panel ();
#endif
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_BGShips.c *** */
/* *** Generated:  Jan 19 11:14:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * This file encapsulates the TAE Plus panel:  BGShips
 * These routines enable panel initialization, creation, and destruction.
 * Access to these routines from other files is enabled by inserting
 * '#include "pan_BGShips.h"'.  For more advanced manipulation of the panel
 * using the TAE routines, the panel's Id, Target, and View are provided.
 * **************************************************************************
 */

#include     "taeconf.inp"
#include     "wptinc.inp"
#include     "global.h"          /* Application globals */
#include     "pan_BGShips.h"

/* One "include" for each connected panel */
#include     "pan_Ship.h"

/*BERN*/
#include     "pan_NewBG.h"
#include     "pan_SetUpBGs.h"
#include     "pan_BGData.h"

extern     void ShowNavyShips();
extern     int GetBGs();
extern     int GetBGShips();
extern     int GetBG();
extern     void ShowBGShips();
extern     char* AddShip();
extern     void SaveBGShips();

Id BGShipsTarget, BGShipsView, BGShipsId;
/* BGShipsDispatch is defined at the end of this file */

/* ************************************************************************
 * Initialize the view and target of this panel.
 */
FUNCTION VOID BGShips_Initialize_Panel (vmCollection)
  Id vmCollection;
  {
  Id Co_Find ();

  BGShipsView = Co_Find (vmCollection, "BGShips_v");
  BGShipsTarget = Co_Find (vmCollection, "BGShips_t");
  }

/* ************************************************************************
 * Create the panel object and display it on the screen.
 */
FUNCTION VOID BGShips_Create_Panel (relativeWindow, flags)
Window     relativeWindow;
COUNT      flags;
{
  /*BERN*/
  TEXT*          bgname[1];
  int            BGIndex = 0;

  if (BGShipsId){
```

```
                printf ("Panel (BGShips) is already displayed.\n");
        }else{
                BGShipsId = Wpt_NewPanel (Default_Display, BGShipsTarget,
                BGShipsView, relativeWindow, BGShipsDispatch, flags);

                /*pass new BG Name from appropriate panel to BG Ships panel*/
                if (NewBGId){
                        strcpy(bgname[0], StringParm(NewBGTarget,"Name"));
                } else {
                        bgname[0] = StringParm(BGDataTarget,"Name");
                }

                Vm_SetString(BGShipsView,"Name.textstrs",1,bgname,P_UPDATE);

                Wpt_ViewUpdate(BGShipsId,"Name", BGShipsView,"Name");

                ShowNavyShips(BGShipsId,"NavyShips");

                if (BGDataId){
                  BGIndex = GetBG(bgname[0],BGIndex);
                  ShowBGShips(BGIndex, BGShipsId, "BGShips");
                }
        }
}

/* ***********************************************************************
 * Erases a panel from the screen and de-allocate the associated panel
 * object.
 */
FUNCTION VOID BGShips_Destroy_Panel ()
  {
  Wpt_PanelErase(BGShipsId);
  BGShipsId=0;
  }

/* ***********************************************************************
 * Connect to this panel. Create it or change it's state.
 */
FUNCTION VOID BGShips_Connect_Panel (relativeWindow, flags)
  Window      relativeWindow;
  COUNT       flags;
  {
  if (BGShipsId)
    Wpt_SetPanelState (BGShipsId, flags);
  else
    BGShips_Create_Panel (relativeWindow, flags);
  }

/* ***********************************************************************
 * Handle event from parameter:  AddShipToBG
 */
EVENT_HANDLER AddShipToBG_Event (value, count)
TEXT      *value[];             /* string pointers */
FUNINT    count;                /* num of values   */
{
        /*BERN*/
        BGInfo                  BGs[MAXBGS];
        F76ShipTypeInfo         F76Table[MAXSHIPTYPES];
        int                     NewBG;
        int                     NewShip;
```

```
        NewBG = GetBGs(BGs);

        NewBG = GetBG( StringParm(NewBGTarget,"Name"), NewBG);

        NewShip = GetBGShips(BGs, NewBG, F76Table);

        AddShip(BGs,NewBG,NewShip,
        StringParm(BGShipsTarget,"NavyShips"));

        SaveBGShips(BGs, NewBG);

        ShowBGShips(NewBG,BGShipsId, "BGShips");
}

/* ***********************************************************************
 * Handle event from parameter:  Close
 */
EVENT_HANDLER Close_Event (value, count)
  TEXT      *value[];            /* string pointers */
  FUNINT     count;               /* num of values  */
  {
  /* Begin generated code for Connection */
  BGShips_Destroy_Panel ();
  /* End generated code for Connection */
  }

/* ***********************************************************************
 * Handle event from parameter:  EditShip
 */
EVENT_HANDLER EditShip_Event (value, count)
  TEXT      *value[];            /* string pointers */
  FUNINT     count;               /* num of values  */
  {
  /* Begin generated code for Connection */
  BGShips_Destroy_Panel ();
  Ship_Connect_Panel (NULL, WPT_PREFERRED);
  /* End generated code for Connection */
  }

/* ***********************************************************************
 * Handle event from parameter:  Help
 */
EVENT_HANDLER Help_Event (value, count)
  TEXT      *value[];            /* string pointers */
  FUNINT     count;               /* num of values  */
  {
  }

/* ***********************************************************************
 * Handle event from parameter:  RemoveFromBG
 */
EVENT_HANDLER RemoveFromBG_Event (value, count)
  TEXT      *value[];            /* string pointers */
  FUNINT     count;               /* num of values  */
  {
  }

/* ***********************************************************************
 * Handle event from parameter:  SaveBGShips
 */
EVENT_HANDLER SaveBGShips_Event (value, count)
```

```
TEXT      *value[];            /* string pointers */
FUNINT     count;              /* num of values  */
{
}


struct DISPATCH BGShipsDispatch[] = {
  {"AddShipToBG", AddShipToBG_Event},
  {"Close", Close_Event},
  {"EditShip", EditShip_Event},
  {"Help", Help_Event},
  {"RemoveFromBG", RemoveFromBG_Event},
  {"SaveBGShips", SaveBGShips_Event},
  {NULL, NULL}                    /* terminator entry */
};
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_BGShips.h *** */
/* *** Generated:  Jan 19 13:12:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Header file for panel:  BGShips
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    The panel's name is changed (not title)
 * For panel:
 *   BGShips
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_PAN_BGShips                  /* prevent double include */
#define I_PAN_BGShips     0

/* Vm objects and panel Id.  */
extern Id BGShipsTarget, BGShipsView, BGShipsId;

/* Dispatch table (global for calls to Wpt_NewPanel) */
extern struct DISPATCH BGShipsDispatch[];

/* Initialize BGShipsTarget and BGShipsView */
extern VOID BGShips_Initialize_Panel ();

/* Create this panel and display it on the screen */
extern VOID BGShips_Create_Panel ();

/* Destroy this panel and erase it from the screen */
extern VOID BGShips_Destroy_Panel ();

/* Connect to this panel.  Create it or change it's state */
extern VOID BGShips_Connect_Panel ();
#endif
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_CloseAll.c *** */
/* *** Generated:  Jan 19 11:14:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * This file encapsulates the TAE Plus panel:  CloseAll
 * These routines enable panel initialization, creation, and destruction.
 * Access to these routines from other files is enabled by inserting
 * '#include "pan_CloseAll.h"'.  For more advanced manipulation of the panel
 * using the TAE routines, the panel's Id, Target, and View are provided.
 * ************************************************************************
 */

#include      "taeconf.inp"
#include      "wptinc.inp"
#include      "global.h"          /* Application globals */
#include      "pan_CloseAll.h"

Id CloseAllTarget, CloseAllView, CloseAllId;
/* CloseAllDispatch is defined at the end of this file */

/* ************************************************************************
 * Initialize the view and target of this panel.
 */
FUNCTION VOID CloseAll_Initialize_Panel (vmCollection)
  Id vmCollection;
  {
  Id  Co_Find ();

  CloseAllView = Co_Find (vmCollection, "CloseAll_v");
  CloseAllTarget = Co_Find (vmCollection, "CloseAll_t");
  }

/* ************************************************************************
 * Create the panel object and display it on the screen.
 */
FUNCTION VOID CloseAll_Create_Panel (relativeWindow, flags)
  Window     relativeWindow;
  COUNT      flags;
  {
  if (CloseAllId)
          printf ("Panel (CloseAll) is already displayed.\n");
  else
          CloseAllId = Wpt_NewPanel(Default_Display, CloseAllTarget,
          CloseAllView, relativeWindow, CloseAllDispatch, flags);
  }

/* ************************************************************************
 * Erases a panel from the screen and de-allocate the associated panel
 * object.
 */
FUNCTION VOID CloseAll_Destroy_Panel ()
  {
  Wpt_PanelErase(CloseAllId);
  CloseAllId=0;
  }

/* ************************************************************************
 * Connect to this panel.  Create it or change it's state.
 */
FUNCTION VOID CloseAll_Connect_Panel (relativeWindow, flags)
```

```
  Window    relativeWindow;
  COUNT     flags;
  {
  if (CloseAllId)
    Wpt_SetPanelState (CloseAllId, flags);
  else
    CloseAll_Create_Panel (relativeWindow, flags);
  }

/* ************************************************************************
 * Handle event from parameter:  message
 */
EVENT_HANDLER message_Event (value, count)
  TEXT      *value[];              /* string pointers */
  FUNINT    count;                 /* num of values   */
  {
  /* Begin generated code for Connection */
  if (count <= 0)
    ;                              /* null value or no value */
  else if (s_equal (value[0], "OK"))
    {
    CloseAll_Destroy_Panel ();
    SET_APPLICATION_DONE;
    }
  else if (s_equal (value[0], "Cancel"))
    {
    CloseAll_Destroy_Panel ();
    }

  /* End generated code for Connection */
  }

struct DISPATCH CloseAllDispatch[] = {
  {"message", message_Event},
  {NULL, NULL}                     /* terminator entry */
};
```

```c
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_CloseAll.h *** */
/* *** Generated:  Jan 19 13:12:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Header file for panel:  CloseAll
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    The panel's name is changed (not title)
 * For panel:
 *   CloseAll
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_PAN_CloseAll                    /* prevent double include */
#define I_PAN_CloseAll     0

/* Vm objects and panel Id.   */
extern Id CloseAllTarget, CloseAllView, CloseAllId;

/* Dispatch table (global for calls to Wpt_NewPanel) */
extern struct DISPATCH CloseAllDispatch[];

/* Initialize CloseAllTarget and CloseAllView */
extern VOID CloseAll_Initialize_Panel ();

/* Create this panel and display it on the screen */
extern VOID CloseAll_Create_Panel ();

/* Destroy this panel and erase it from the screen */
extern VOID CloseAll_Destroy_Panel ();

/* Connect to this panel.  Create it or change it's state */
extern VOID CloseAll_Connect_Panel ();
#endif
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_DelBG.c *** */
/* *** Generated:  Feb 8 10:13:45 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * This file encapsulates the TAE Plus panel:  DelBG
 * These routines enable panel initialization, creation, and destruction.
 * Access to these routines from other files is enabled by inserting
 * '#include "pan_DelBG.h"'. For more advanced manipulation of the panel
 * using the TAE routines, the panel's Id, Target, and View are provided.
 *
 * For the panel items:
 *   Message
 *
 * CHANGE LOG:
 *  8-Feb-93   Initially generated...TAE
 * ************************************************************************
 */

#include     "taeconf.inp"
#include     "wptinc.inp"
#include     "global.h"          /* Application globals */
#include     "pan_DelBG.h"

/* One "include" for each connected panel */

/*BERN*/
#include     "pan_SetUpBGs.h"

extern       int GetBGs();
extern       int GetBG();
extern       void DeleteBG();
extern       void SaveBGs();
extern       void ShowBGs();


Id DelBGTarget, DelBGView, DelBGId;
/* DelBGDispatch is defined at the end of this file */

/* ************************************************************************
 * Initialize the view and target of this panel.
 */
FUNCTION VOID DelBG_Initialize_Panel (vmCollection)
   Id vmCollection;
   {
   Id  Co_Find ();

   DelBGView = Co_Find (vmCollection, "DelBG_v");
   DelBGTarget = Co_Find (vmCollection, "DelBG_t");
   }

/* ************************************************************************
 * Create the panel object and display it on the screen.
 */
FUNCTION VOID DelBG_Create_Panel (relativeWindow, flags)
   Window     relativeWindow;
   COUNT      flags;
   {
   if (DelBGId)
      printf ("Panel (DelBG) is already displayed.\n");
   else
```

```
    DelBGId = Wpt_NewPanel (Default_Display, DelBGTarget, DelBGView,
    relativeWindow, DelBGDispatch, flags);
  }

/* ***********************************************************************
 * Erases a panel from the screen and de-allocate the associated panel
 * object.
 */
FUNCTION VOID DelBG_Destroy_Panel ()
  {
  Wpt_PanelErase(DelBGId);
  DelBGId=0;
  }

/* ***********************************************************************
 * Connect to this panel.  Create it or change it's state.
 */
FUNCTION VOID DelBG_Connect_Panel (relativeWindow, flags)
  Window     relativeWindow;
  COUNT      flags;
  {
  if (DelBGId)
    Wpt_SetPanelState (DelBGId, flags);
  else
    DelBG_Create_Panel (relativeWindow, flags);
  }

/* ***********************************************************************
 * Handle event from parameter:  Message
 */
EVENT_HANDLER Message_Event (value, count)
  TEXT      *value[];            /* string pointers */
  FUNINT    count;               /* num of values   */
  {

  /*BERN*/
  BGInfo                BGs[MAXBGS];
  int                   BGToDelete;
  int                   BGIndex;

  /* Begin generated code for Connection */

  if (count <= 0)
    ;                           /* null value or no value */
  else if (s_equal (value[0], "OK"))
    {
    BGIndex = GetBGs(BGs);

    BGToDelete=GetBG(StringParm(SetUpBGsTarget,"BGList"), BGIndex);

    DeleteBG(BGs,BGToDelete);

    SaveBGs(BGs);

    ShowBGs(SetUpBGsId,"BGList");

    DelBG_Destroy_Panel ();

    }
  else if (s_equal (value[0], "Cancel"))
    {
```

```
        DelBG_Destroy_Panel ();
        }

    /* End generated code for Connection */
    }


struct DISPATCH DelBGDispatch[] = {
    {"Message", Message_Event},
    {NULL, NULL}                    /* terminator entry */
};
```

```c
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_DelBG.h *** */
/* *** Generated:  Jan 19 13:12:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Header file for panel:  DelBG
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    The panel's name is changed (not title)
 * For panel:
 *   DelBG
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_PAN_DelBG                    /* prevent double include */
#define I_PAN_DelBG     0

/* Vm objects and panel Id.  */
extern Id DelBGTarget, DelBGView, DelBGId;

/* Dispatch table (global for calls to Wpt_NewPanel) */
extern struct DISPATCH DelBGDispatch[];

/* Initialize DelBGTarget and DelBGView */
extern VOID DelBG_Initialize_Panel ();

/* Create this panel and display it on the screen */
extern VOID DelBG_Create_Panel ();

/* Destroy this panel and erase it from the screen */
extern VOID DelBG_Destroy_Panel ();

/* Connect to this panel.  Create it or change it's state */
extern VOID DelBG_Connect_Panel ();
#endif
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_Dtg.c *** */
/* *** Generated:  Feb  8 10:13:45 1993 *** */
/* **********************************************************************
 * PURPOSE:
 * This file encapsulates the TAE Plus panel:  Dtg
 * These routines enable panel initialization, creation, and destruction.
 * Access to these routines from other files is enabled by inserting
 * '#include "pan_Dtg.h"'.  For more advanced manipulation of the panel
 * using the TAE routines, the panel's Id, Target, and View are provided.
 *
 * For the panel items:
 *   Message
 *
 * CHANGE LOG:
 *  8-Feb-93   Initially generated...TAE
 * **********************************************************************
 */

#include      "taeconf.inp"
#include      "wptinc.inp"
#include      "global.h"          /* Application globals */
#include      "pan_Dtg.h"

/* One "include" for each connected panel */

Id DtgTarget, DtgView, DtgId;
/* DtgDispatch is defined at the end of this file */

/* **********************************************************************
 * Initialize the view and target of this panel.
 */
FUNCTION VOID Dtg_Initialize_Panel (vmCollection)
   Id vmCollection;
   {
   Id  Co_Find ();

   DtgView = Co_Find (vmCollection, "Dtg_v");
   DtgTarget = Co_Find (vmCollection, "Dtg_t");
   }

/* **********************************************************************
 * Create the panel object and display it on the screen.
 */
FUNCTION VOID Dtg_Create_Panel (relativeWindow, flags)
   Window     relativeWindow;
   COUNT      flags;
   {
   if (DtgId)
      printf ("Panel (Dtg) is already displayed.\n");
   else
      DtgId = Wpt_NewPanel (Default_Display, DtgTarget, DtgView,
                  relativeWindow, DtgDispatch, flags);
   }

/* **********************************************************************
 * Erases a panel from the screen and de-allocate the associated panel
 * object.
 */
FUNCTION VOID Dtg_Destroy_Panel ()
```

```
  {
  Wpt_PanelErase(DtgId);
  DtgId=0;
  }

/* ***************************************************************************
 * Connect to this panel. Create it or change it's state.
 */
FUNCTION VOID Dtg_Connect_Panel (relativeWindow, flags)
  Window      relativeWindow;
  COUNT       flags;
  {
  if (DtgId)
    Wpt_SetPanelState (DtgId, flags);
  else
    Dtg_Create_Panel (relativeWindow, flags);
  }

/* ***************************************************************************
 * Handle event from parameter: Message
 */
EVENT_HANDLER Message_Event (value, count)
  TEXT      *value[];          /* string pointers */
  FUNINT    count;             /* num of values   */
  {
  /* Begin generated code for Connection */

  if (count <= 0)
    ;                          /* null value or no value */
  else if (s_equal (value[0], "OK"))
    {
    Dtg_Destroy_Panel ();
    }
  else if (s_equal (value[0], "Cancel"))
    {
    Dtg_Destroy_Panel ();
    }

  /* End generated code for Connection */
  }


struct DISPATCH DtgDispatch[] = {
  {"Message", Message_Event},
  {NULL, NULL}                 /* terminator entry */
};
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:     pan_Dtg.h *** */
/* *** Generated:  Jan 19 13:12:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Header file for panel:  Dtg
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    The panel's name is changed (not title)
 * For panel:
 *   Dtg
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_PAN_Dtg                    /* prevent double include */
#define I_PAN_Dtg     0

/* Vm objects and panel Id.  */
extern Id DtgTarget, DtgView, DtgId;

/* Dispatch table (global for calls to Wpt_NewPanel) */
extern struct DISPATCH DtgDispatch[];

/* Initialize DtgTarget and DtgView */
extern VOID Dtg_Initialize_Panel ();

/* Create this panel and display it on the screen */
extern VOID Dtg_Create_Panel ();

/* Destroy this panel and erase it from the screen */
extern VOID Dtg_Destroy_Panel ();

/* Connect to this panel.  Create it or change it's state */
extern VOID Dtg_Connect_Panel ();
#endif
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_LackData.c *** */
/* *** Generated:  Feb  8 10:13:45 1993 *** */
/* **********************************************************************
 * PURPOSE:
 * This file encapsulates the TAE Plus panel:  LackData
 * These routines enable panel initialization, creation, and destruction.
 * Access to these routines from other files is enabled by inserting
 * '#include "pan_LackData.h"'.  For more advanced manipulation of the panel
 * using the TAE routines, the panel's Id, Target, and View are provided.
 *
 * For the panel items:
 *   Message
 *
 * CHANGE LOG:
 *  8-Feb-93   Initially generated...TAE
 * **********************************************************************
 */

#include     "taeconf.inp"
#include     "wptinc.inp"
#include     "global.h"          /* Application globals */
#include     "pan_LackData.h"

/* One "include" for each connected panel */

Id LackDataTarget, LackDataView, LackDataId;
/* LackDataDispatch is defined at the end of this file */

/* **********************************************************************
 * Initialize the view and target of this panel.
 */
FUNCTION VOID LackData_Initialize_Panel (vmCollection)
   Id vmCollection;
   {
   Id  Co_Find ();

   LackDataView = Co_Find (vmCollection, "LackData_v");
   LackDataTarget = Co_Find (vmCollection, "LackData_t");
   }

/* **********************************************************************
 * Create the panel object and display it on the screen.
 */
FUNCTION VOID LackData_Create_Panel (relativeWindow, flags)
   Window     relativeWindow;
   COUNT      flags;
   {
   if (LackDataId)
      printf ("Panel (LackData) is already displayed.\n");
   else
         LackDataId =Wpt_NewPanel (Default_Display, LackDataTarget,
          LackDataView,  relativeWindow, LackDataDispatch, flags);
   }

/* **********************************************************************
 * Erases a panel from the screen and de-allocate the associated panel
 * object.
 */
FUNCTION VOID LackData_Destroy_Panel ()
   {
```

```
   Wpt_PanelErase(LackDataId);
   LackDataId=0;
   }


/* *************************************************************************
 * Connect to this panel.  Create it or change it's state.
 */
FUNCTION VOID LackData_Connect_Panel (relativeWindow, flags)
   Window     relativeWindow;
   COUNT      flags;
   {
   if (LackDataId)
      Wpt_SetPanelState (LackDataId, flags);
   else
      LackData_Create_Panel (relativeWindow, flags);
   }


/* *************************************************************************
 * Handle event from parameter:  Message
 */
EVENT_HANDLER Message_Event (value, count)
   TEXT       *value[];              /* string pointers */
   FUNINT     count;                 /* num of values   */
   {
   /* Begin generated code for Connection */

   if (count <= 0)
      ;                            /* null value or no value */
   else if (s_equal (value[0], "OK"))
      {
      LackData_Destroy_Panel ();
      }
   else if (s_equal (value[0], "Cancel"))
      {
      LackData_Destroy_Panel ();
      }

   /* End generated code for Connection */
   }


struct DISPATCH LackDataDispatch[] = {
   {"Message", Message_Event},
   {NULL, NULL}                      /* terminator entry */
};
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:     pan_LackData.h *** */
/* *** Generated:  Jan 19 13:12:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Header file for panel:  LackData
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    The panel's name is changed (not title)
 * For panel:
 *   LackData
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_PAN_LackData              /* prevent double include */
#define I_PAN_LackData     0

/* Vm objects and panel Id.  */
extern Id LackDataTarget, LackDataView, LackDataId;

/* Dispatch table (global for calls to Wpt_NewPanel) */
extern struct DISPATCH LackDataDispatch[];

/* Initialize LackDataTarget and LackDataView */
extern VOID LackData_Initialize_Panel ();

/* Create this panel and display it on the screen */
extern VOID LackData_Create_Panel ();

/* Destroy this panel and erase it from the screen */
extern VOID LackData_Destroy_Panel ();

/* Connect to this panel.  Create it or change it's state */
extern VOID LackData_Connect_Panel ();
#endif
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_NewBG.c *** */
/* *** Generated:  Jan 19 11:14:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * This file encapsulates the TAE Plus panel:  NewBG
 * These routines enable panel initialization, creation, and destruction.
 * Access to these routines from other files is enabled by inserting
 * '#include "pan_NewBG.h"'. For more advanced manipulation of the panel
 * using the TAE routines, the panel's Id, Target, and View are provided.
 * ************************************************************************
 */

#include      "taeconf.inp"
#include      "wptinc.inp"
#include      "global.h"           /* Application globals */
#include      "pan_NewBG.h"

/* One "include" for each connected panel */
#include      "pan_BGShips.h"

/*BERN*/
#include            "pan_LackData.h"
#include            "pan_SetUpBGs.h"
#include            "pan_SaveNewB.h"

/*BERN*/
extern              int GetBGs();
extern              int MakeBG();
extern              void SaveBGs();

int                 SaveFlag = 0;

Id NewBGTarget, NewBGView, NewBGId;
/* NewBGDispatch is defined at the end of this file */

/* ************************************************************************
 * Initialize the view and target of this panel.
 */
FUNCTION VOID NewBG_Initialize_Panel (vmCollection)
   Id vmCollection;
   {
   Id  Co_Find ();

   NewBGView = Co_Find (vmCollection, "NewBG_v");
   NewBGTarget = Co_Find (vmCollection, "NewBG_t");
   }

/* ************************************************************************
 * Create the panel object and display it on the screen.
 */
FUNCTION VOID NewBG_Create_Panel (relativeWindow, flags)
   Window     relativeWindow;
   COUNT      flags;
   {
   if (NewBGId)
      printf ("Panel (NewBG) is already displayed.\n");
   else
      NewBGId = Wpt_NewPanel (Default_Display, NewBGTarget,NewBGView,
      relativeWindow, NewBGDispatch, flags);
   }
```

```
/* ************************************************************************
 * Erases a panel from the screen and de-allocate the associated panel
 * object.
 */
FUNCTION VOID NewBG_Destroy_Panel ()
  {
  Wpt_PanelErase(NewBGId);
  NewBGId=0;
  }


/* ************************************************************************
 * Connect to this panel. Create it or change it's state.
 */
FUNCTION VOID NewBG_Connect_Panel (relativeWindow, flags)
  Window     relativeWindow;
  COUNT      flags;
  {
  if (NewBGId)
    Wpt_SetPanelState (NewBGId, flags);
  else
    NewBG_Create_Panel (relativeWindow, flags);
  }

/* ************************************************************************
 * Handle event from parameter: Close
 */
EVENT_HANDLER Close_Event (value, count)
TEXT      *value[];             /* string pointers */
FUNINT    count;                /* num of values   */
{
        if (SaveFlag == 1){
          /* Begin default generated code */
                NewBG_Destroy_Panel ();
                /* End generated code for Connection */
        } else {
                SaveNewB_Connect_Panel(NULL, WPT_PREFERRED);
        }
}

/* ************************************************************************
 * Handle event from parameter: Help
 */
EVENT_HANDLER Help_Event (value, count)
  TEXT      *value[];             /* string pointers */
  FUNINT    count;                /* num of values   */
  {
  }

/* ************************************************************************
 * Handle event from parameter: Save
 */
EVENT_HANDLER Save_Event (value, count)
TEXT      *value[];             /* string pointers */
FUNINT    count;                /* num of values   */
{
    BGInfo              BGs[MAXBGS];
    int                 BGIndex;
```

```
        BGIndex                 = GetBGs(BGs);

        if (MakeBG(BGs,BGIndex,StringParm(NewBGTarget,"Name"),
            StringParm(NewBGTarget,"Designation"),
            RealParm  (NewBGTarget,"FuelRes"),
            RealParm  (NewBGTarget,"CLFFuelRes"),
            RealParm  (NewBGTarget,"OrdRes"),
            RealParm  (NewBGTarget,"CLFOrdRes"),
            RealParm  (NewBGTarget,"MaxF76"),
            RealParm  (NewBGTarget,"MaxF44"),
            RealParm  (NewBGTarget,"StationSpeed"),
            RealParm  (NewBGTarget,"UnrepSpeed"),
            RealParm  (NewBGTarget,"AcftShipSpeed"))) {

            SaveBGs(BGs);

            /*refresh the BGList in setupbgs panel*/
            BGIndex = GetBGs(BGs);

            ShowBGs(SetUpBGsId,"BGList");

            SaveFlag = 1;

                /* Begin generated code for Connection */
                BGShips_Connect_Panel (NULL, WPT_PREFERRED);
                /* End generated code for Connection */
    } else {
            LackData_Connect_Panel(NULL,WPT_PREFERRED);
    }

}

struct DISPATCH NewBGDispatch[] = {
    {"Close", Close_Event},
    {"Help", Help_Event},
    {"Save", Save_Event},
    {NULL, NULL}                    /* terminator entry */
};
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_NewBG.h *** */
/* *** Generated:  Jan 19 13:12:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Header file for panel:  NewBG
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    The panel's name is changed (not title)
 * For panel:
 *   NewBG
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_PAN_NewBG                    /* prevent double include */
#define I_PAN_NewBG     0

/* Vm objects and panel Id.  */
extern Id NewBGTarget, NewBGView, NewBGId;

/* Dispatch table (global for calls to Wpt_NewPanel) */
extern struct DISPATCH NewBGDispatch[];

/* Initialize NewBGTarget and NewBGView */
extern VOID NewBG_Initialize_Panel ();

/* Create this panel and display it on the screen */
extern VOID NewBG_Create_Panel ();

/* Destroy this panel and erase it from the screen */
extern VOID NewBG_Destroy_Panel ();

/* Connect to this panel.  Create it or change it's state */
extern VOID NewBG_Connect_Panel ();
#endif
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_SaveNewB.c *** */
/* *** Generated:  Feb 8 10:13:45 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * This file encapsulates the TAE Plus panel:  SaveNewB
 * These routines enable panel initialization, creation, and destruction.
 * Access to these routines from other files is enabled by inserting
 * '#include "pan_SaveNewB.h"'.  For more advanced manipulation of the panel
 * using the TAE routines, the panel's Id, Target, and View are provided.
 *
 * For the panel items:
 *   Message
 *
 * CHANGE LOG:
 *  8-Feb-93   Initially generated...TAE
 * ************************************************************************
 */

#include      "taeconf.inp"
#include      "wptinc.inp"
#include      "global.h"              /* Application globals */
#include      "pan_SaveNewB.h"

/*BERN*/
#include      "pan_SetUpBGs.h"
#include      "pan_BGShips.h"
#include      "pan_NewBG.h"
#include      "pan_LackData.h"

extern        int SaveNewBG();

Id SaveNewBTarget, SaveNewBView, SaveNewBId;
/* SaveNewBDispatch is defined at the end of this file */

/* ************************************************************************
 * Initialize the view and target of this panel.
 */
FUNCTION VOID SaveNewB_Initialize_Panel (vmCollection)
  Id vmCollection;
  {
  Id Co_Find ();

  SaveNewBView = Co_Find (vmCollection, "SaveNewB_v");
  SaveNewBTarget = Co_Find (vmCollection, "SaveNewB_t");
  }

/* ************************************************************************
 * Create the panel object and display it on the screen.
 */
FUNCTION VOID SaveNewB_Create_Panel (relativeWindow, flags)
  Window     relativeWindow;
  COUNT      flags;
  {
  if (SaveNewBId)
     printf ("Panel (SaveNewB) is already displayed.\n");
  else
     SaveNewBId = Wpt_NewPanel (Default_Display,
     SaveNewBTarget,SaveNewBView, relativeWindow,
     SaveNewBDispatch, flags);
  }
```

```
/* ***********************************************************************
 * Erases a panel from the screen and de-allocate the associated panel
 * object.
 */
FUNCTION VOID SaveNewB_Destroy_Panel ()
   {
   Wpt_PanelErase(SaveNewBId);
   SaveNewBId=0;
   }


/* ***********************************************************************
 * Connect to this panel.  Create it or change it's state.
 */
FUNCTION VOID SaveNewB_Connect_Panel (relativeWindow, flags)
   Window      relativeWindow;
   COUNT       flags;
   {
   if (SaveNewBId)
      Wpt_SetPanelState (SaveNewBId, flags);
   else
      SaveNewB_Create_Panel (relativeWindow, flags);
   }

/* ***********************************************************************
 * Handle event from parameter:  Message
 */
EVENT_HANDLER Message_Event (value, count)
   TEXT        *value[];               /* string pointers */
   FUNINT      count;                  /* num of values   */
   {
   /*BERN*/
   int                 BGIndex;
   BGInfo              BGs[MAXBGS];

   /* Begin generated code for Connection */

   if (count <= 0)
      ;                                /* null value or no value */
   else if (s_equal (value[0], "OK"))
      {
      if (SaveNewBG(NewBGTarget) == 1){

            BGIndex = GetBGs(BGs);

            ShowBGs(SetUpBGsId,"BGList");

            BGShips_Connect_Panel (NULL, WPT_PREFERRED);

            SaveNewB_Destroy_Panel ();

      } else {

            SaveNewB_Destroy_Panel ();

            LackData_Connect_Panel(NULL,WPT_PREFERRED);
      }

      }
```

```
else if (s_equal (value[0], "Cancel"))
  {
  NewBG_Destroy_Panel();

  SaveNewB_Destroy_Panel ();
  }

/* End generated code for Connection */
}


struct DISPATCH SaveNewBDispatch[] = {
  {"Message", Message_Event},
  {NULL, NULL}                    /* terminator entry */
};
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_SaveNewB.h *** */
/* *** Generated:  Jan 19 13:12:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Header file for panel:  SaveNewB
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    The panel's name is changed (not title)
 * For panel:
 *   SaveNewB
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_PAN_SaveNewB                /* prevent double include */
#define I_PAN_SaveNewB     0

/* Vm objects and panel Id.  */
extern Id SaveNewBTarget, SaveNewBView, SaveNewBId;

/* Dispatch table (global for calls to Wpt_NewPanel) */
extern struct DISPATCH SaveNewBDispatch[];

/* Initialize SaveNewBTarget and SaveNewBView */
extern VOID SaveNewB_Initialize_Panel ();

/* Create this panel and display it on the screen */
extern VOID SaveNewB_Create_Panel ();

/* Destroy this panel and erase it from the screen */
extern VOID SaveNewB_Destroy_Panel ();

/* Connect to this panel.  Create it or change it's state */
extern VOID SaveNewB_Connect_Panel ();
#endif
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:       pan_SelBG.c *** */
/* *** Generated:  Feb  8 10:13:45 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * This file encapsulates the TAE Plus panel:  SelBG
 * These routines enable panel initialization, creation, and destruction.
 * Access to these routines from other files is enabled by inserting
 * '#include "pan_SelBG.h"'. For more advanced manipulation of the panel
 * using the TAE routines, the panel's Id, Target, and View are provided.
 *
 * For the panel items:
 *   Message
 *
 * CHANGE LOG:
 *  8-Feb-93   Initially generated...TAE
 * ************************************************************************
 */

#include      "taeconf.inp"
#include      "wptinc.inp"
#include      "global.h"            /* Application globals */
#include      "pan_SelBG.h"

/* One "include" for each connected panel */

Id SelBGTarget, SelBGView, SelBGId;
/* SelBGDispatch is defined at the end of this file */

/* ************************************************************************
 * Initialize the view and target of this panel.
 */
FUNCTION VOID SelBG_Initialize_Panel (vmCollection)
   Id vmCollection;
   {
   Id  Co_Find ();

   SelBGView = Co_Find (vmCollection, "SelBG_v");
   SelBGTarget = Co_Find (vmCollection, "SelBG_t");
   }


/* ************************************************************************
 * Create the panel object and display it on the screen.
 */
FUNCTION VOID SelBG_Create_Panel (relativeWindow, flags)
   Window     relativeWindow;
   COUNT      flags;
   {
   if (SelBGId)
      printf ("Panel (SelBG) is already displayed.\n");
   else
      SelBGId = Wpt_NewPanel (Default_Display, SelBGTarget, SelBGView,
                     relativeWindow, SelBGDispatch, flags);
   }


/* ************************************************************************
 * Erases a panel from the screen and de-allocate the associated panel
 * object.
 */
```

```
FUNCTION VOID SelBG_Destroy_Panel ()
   {
   Wpt_PanelErase(SelBGId);
   SelBGId=0;
   }


/* ***********************************************************************
 * Connect to this panel.  Create it or change it's state.
 */
FUNCTION VOID SelBG_Connect_Panel (relativeWindow, flags)
   Window      relativeWindow;
   COUNT       flags;
   {
   if (SelBGId)
      Wpt_SetPanelState (SelBGId, flags);
   else
      SelBG_Create_Panel (relativeWindow, flags);
   }


/* ***********************************************************************
 * Handle event from parameter:  Message
 */
EVENT_HANDLER Message_Event (value, count)
   TEXT      *value[];              /* string pointers */
   FUNINT    count;                 /* num of values   */
   {
   /* Begin generated code for Connection */

   if (count <= 0)
      ;                             /* null value or no value */
   else if (s_equal (value[0], "OK"))
      {
      SelBG_Destroy_Panel ();
      }
   else if (s_equal (value[0], "Cancel"))
      {
      SelBG_Destroy_Panel ();
      }

   /* End generated code for Connection */
   }


struct DISPATCH SelBGDispatch[] = {
   {"Message", Message_Event},
   {NULL, NULL}                     /* terminator entry */
};
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_SelBG.h *** */
/* *** Generated: Jan 19 13:12:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Header file for panel: SelBG
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    The panel's name is changed (not title)
 * For panel:
 *   SelBG
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_PAN_SelBG                  /* prevent double include */
#define I_PAN_SelBG     0

/* Vm objects and panel Id.  */
extern Id SelBGTarget, SelBGView, SelBGId;

/* Dispatch table (global for calls to Wpt_NewPanel) */
extern struct DISPATCH SelBGDispatch[];

/* Initialize SelBGTarget and SelBGView */
extern VOID SelBG_Initialize_Panel ();

/* Create this panel and display it on the screen */
extern VOID SelBG_Create_Panel ();

/* Destroy this panel and erase it from the screen */
extern VOID SelBG_Destroy_Panel ();

/* Connect to this panel.  Create it or change it's state */
extern VOID SelBG_Connect_Panel ();
#endif
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_SetUpBGs.c *** */
/* *** Generated:  Jan 19 11:14:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * This file encapsulates the TAE Plus panel:  SetUpBGs
 * These routines enable panel initialization, creation, and destruction.
 * Access to these routines from other files is enabled by inserting
 * '#include "pan_SetUpBGs.h"'.  For more advanced manipulation of the panel
 * using the TAE routines, the panel's Id, Target, and View are provided.
 * ************************************************************************
 */

#include    "taeconf.inp"
#include    "wptinc.inp"
#include    "global.h"          /* Application globals */
#include    "pan_SetUpBGs.h"

/* One "include" for each connected panel */
#include    "pan_CloseAll.h"
#include    "pan_DelBG.h"
#include    "pan_BGData.h"
#include    "pan_NewBG.h"

/*BERN*/
#include    "pan_SelBG.h"
#include    "pan_CloseAll.h"

extern          int GetBGs();
extern          void ShowBGs();

Id SetUpBGsTarget, SetUpBGsView, SetUpBGsId;
/* SetUpBGsDispatch is defined at the end of this file */

/* ************************************************************************
 * Initialize the view and target of this panel.
 */
FUNCTION VOID SetUpBGs_Initialize_Panel (vmCollection)
  Id vmCollection;
  {
  Id  Co_Find ();

  SetUpBGsView = Co_Find (vmCollection, "SetUpBGs_v");
  SetUpBGsTarget = Co_Find (vmCollection, "SetUpBGs_t");
  }
/* ************************************************************************
 * Create the panel object and display it on the screen.
 */
FUNCTION VOID SetUpBGs_Create_Panel (relativeWindow, flags)
  Window    relativeWindow;
  COUNT     flags;
  {

  /*BERN*/
  BGInfo    BGs[MAXBGS];

  if (SetUpBGsId)
    printf ("Panel (SetUpBGs) is already displayed.\n");
  else
    SetUpBGsId = Wpt_NewPanel(Default_Display,
    SetUpBGsTarget,SetUpBGsView, relativeWindow,
```

```
    SetUpBGsDispatch, flags);

    GetBGs(BGs);

    ShowBGs(SetUpBGsId,"BGList");
  }

/* ***********************************************************************
 * Erases a panel from the screen and de-allocate the associated panel
 * object.
 */
FUNCTION VOID SetUpBGs_Destroy_Panel ()
  {
  Wpt_PanelErase(SetUpBGsId);
  SetUpBGsId=0;
  }


/* ***********************************************************************
 * Connect to this panel.  Create it or change it's state.
 */
FUNCTION VOID SetUpBGs_Connect_Panel (relativeWindow, flags)
  Window     relativeWindow;
  COUNT      flags;
  {
  if (SetUpBGsId)
    Wpt_SetPanelState (SetUpBGsId, flags);
  else
    SetUpBGs_Create_Panel (relativeWindow, flags);
  }
/* ***********************************************************************
 * Handle event from parameter:  Close
 */
EVENT_HANDLER Close_Event (value, count)
  TEXT      *value[];             /* string pointers */
  FUNINT    count;                /* num of values   */
  {
  /* Begin generated code for Connection */
  CloseAll_Connect_Panel (NULL, WPT_PREFERRED);
  /* End generated code for Connection */
  }

/* ***********************************************************************
 * Handle event from parameter:  Delete
 */
EVENT_HANDLER Delete_Event (value, count)
TEXT      *value[];               /* string pointers */
FUNINT    count;                  /* num of values   */
{
  /*BERN*/
  if (StringParm(SetUpBGsTarget,"BGList") != NULL){

          /* Begin generated code for Connection */
          DelBG_Connect_Panel (NULL, WPT_PREFERRED);
          /* End generated code for Connection */

  } else {

        SelBG_Connect_Panel(NULL, WPT_PREFERRED);
  }
}
```

```
/* ***************************************************************************
 * Handle event from parameter:  Edit
 */
EVENT_HANDLER Edit_Event (value, count)
TEXT        *value[];                /* string pointers */
FUNINT      count;                   /* num of values   */
{
   /*BERN*/
   if (StringParm(SetUpBGsTarget,"BGList") != NULL){

          /* Begin generated code for Connection */
          BGData_Connect_Panel (NULL, WPT_PREFERRED);
          /* End generated code for Connection */

   } else {

       SelBG_Connect_Panel(NULL, WPT_PREFERRED);
   }
}

/* ***************************************************************************
 * Handle event from parameter:  Help
 */
EVENT_HANDLER Help_Event (value, count)
   TEXT        *value[];                /* string pointers */
   FUNINT      count;                   /* num of values   */
   {
   }


/* ***************************************************************************
 * Handle event from parameter:  New
 */
EVENT_HANDLER New_Event (value, count)
   TEXT        *value[];                /* string pointers */
   FUNINT      count;                   /* num of values   */
   {
   /* Begin generated code for Connection */
   NewBG_Connect_Panel (NULL, WPT_PREFERRED);
   /* End generated code for Connection */
   }

struct DISPATCH SetUpBGsDispatch[] = {
   {"Close", Close_Event},
   {"Delete", Delete_Event},
   {"Edit", Edit_Event},
   {"Help", Help_Event},
   {"New", New_Event},
   {NULL, NULL}                    /* terminator entry */
};
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_SetUpBGs.h *** */
/* *** Generated: Jan 19 13:12:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Header file for panel: SetUpBGs
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    The panel's name is changed (not title)
 * For panel:
 *   SetUpBGs
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_PAN_SetUpBGs                /* prevent double include */
#define I_PAN_SetUpBGs     0

/* Vm objects and panel Id.  */
extern Id SetUpBGsTarget, SetUpBGsView, SetUpBGsId;

/* Dispatch table (global for calls to Wpt_NewPanel) */
extern struct DISPATCH SetUpBGsDispatch[];

/* Initialize SetUpBGsTarget and SetUpBGsView */
extern VOID SetUpBGs_Initialize_Panel ();

/* Create this panel and display it on the screen */
extern VOID SetUpBGs_Create_Panel ();

/* Destroy this panel and erase it from the screen */
extern VOID SetUpBGs_Destroy_Panel ();

/* Connect to this panel.  Create it or change it's state */
extern VOID SetUpBGs_Connect_Panel ();
#endif
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:     pan_Ship.c *** */
/* *** Generated:  Feb 8 10:40:35 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * This file encapsulates the TAE Plus panel:  Ship
 * These routines enable panel initialization, creation, and destruction.
 * Access to these routines from other files is enabled by inserting
 * '#include "pan_Ship.h"'. For more advanced manipulation of the panel
 * using the TAE routines, the panel's Id, Target, and View are provided.
 *
 * For the panel items:
 *   Aircraft,    Close,       F44,        F76,
 *   Help,        Ordnance,    Print,      Save,
 *
 * CHANGE LOG:
 * 8-Feb-93   Initially generated...TAE
 * ************************************************************************
 */

#include     "taeconf.inp"
#include     "wptinc.inp"
#include     "global.h"          /* Application globals */
#include     "pan_Ship.h"

/* One "include" for each connected panel */
#include     "pan_AcftLoad.h"
#include     "pan_F44Fuel.h"
#include     "pan_F76Fuel.h"
#include     "pan_OrdSel.h"
#include     "pan_PrintJob.h"

Id ShipTarget, ShipView, ShipId;
/* ShipDispatch is defined at the end of this file */

/* ************************************************************************
 * Initialize the view and target of this panel.
 */
FUNCTION VOID Ship_Initialize_Panel (vmCollection)
  Id vmCollection;
  {
  Id  Co_Find ();

  ShipView = Co_Find (vmCollection, "Ship_v");
  ShipTarget = Co_Find (vmCollection, "Ship_t");
  }

/* ************************************************************************
 * Create the panel object and display it on the screen.
 */
FUNCTION VOID Ship_Create_Panel (relativeWindow, flags)
  Window     relativeWindow;
  COUNT      flags;
  {
  if (ShipId)
     printf ("Panel (Ship) is already displayed.\n");
  else
     ShipId = Wpt_NewPanel (Default_Display, ShipTarget, ShipView,
                  relativeWindow, ShipDispatch, flags);
  }
```

```
/* ***************************************************************************
 * Erases a panel from the screen and de-allocate the associated panel
 * object.
 */
FUNCTION VOID Ship_Destroy_Panel ()
   {
   Wpt_PanelErase(ShipId);
   ShipId=0;
   }


/* ***************************************************************************
 * Connect to this panel.  Create it or change it's state.
 */
FUNCTION VOID Ship_Connect_Panel (relativeWindow, flags)
   Window      relativeWindow;
   COUNT       flags;
   {
   if (ShipId)
      Wpt_SetPanelState (ShipId, flags);
   else
      Ship_Create_Panel (relativeWindow, flags);
   }


/* ***************************************************************************
 * Handle event from parameter:  Aircraft
 */
EVENT_HANDLER Aircraft_Event (value, count)
   TEXT      *value[];              /* string pointers */
   FUNINT     count;                /* num of values   */
   {
   /* Begin generated code for Connection */
   AcftLoad_Connect_Panel (NULL, WPT_PREFERRED);
   /* End generated code for Connection */
   }


/* ***************************************************************************
 * Handle event from parameter:  Close
 */
EVENT_HANDLER Close_Event (value, count)
   TEXT      *value[];              /* string pointers */
   FUNINT     count;                /* num of values   */
   {
   /* Begin generated code for Connection */
   Ship_Destroy_Panel ();
   /* End generated code for Connection */
   }


/* ***************************************************************************
 * Handle event from parameter:  F44
 */
EVENT_HANDLER F44_Event (value, count)
   TEXT      *value[];              /* string pointers */
   FUNINT     count;                /* num of values   */
   {
   /* Begin generated code for Connection */
   F44Fuel_Connect_Panel (NULL, WPT_PREFERRED);
```

```
   /* End generated code for Connection */
   }

/* ************************************************************************
 * Handle event from parameter:  F76
 */
EVENT_HANDLER F76_Event (value, count)
   TEXT       *value[];              /* string pointers */
   FUNINT     count;                  /* num of values   */
   {
   /* Begin generated code for Connection */
   F76Fuel_Connect_Panel (NULL, WPT_PREFERRED);
   /* End generated code for Connection */
   }

/* ************************************************************************
 * Handle event from parameter:  Help
 */
EVENT_HANDLER Help_Event (value, count)
   TEXT       *value[];              /* string pointers */
   FUNINT     count;                  /* num of values   */
   {
   }

/* ************************************************************************
 * Handle event from parameter:  Ordnance
 */
EVENT_HANDLER Ordnance_Event (value, count)
   TEXT       *value[];              /* string pointers */
   FUNINT     count;                  /* num of values   */
   {
   /* Begin generated code for Connection */
   OrdSel_Connect_Panel (NULL, WPT_PREFERRED);
   /* End generated code for Connection */
   }

/* ************************************************************************
 * Handle event from parameter:  Print
 */
EVENT_HANDLER Print_Event (value, count)
   TEXT       *value[];              /* string pointers */
   FUNINT     count;                  /* num of values   */
   {
   /* Begin generated code for Connection */
   PrintJob_Connect_Panel (NULL, WPT_PREFERRED);
   /* End generated code for Connection */
   }

/* ************************************************************************
 * Handle event from parameter:  Save
 */
EVENT_HANDLER Save_Event (value, count)
   TEXT       *value[];              /* string pointers */
   FUNINT     count;                  /* num of values   */
   {
   }
```

```
struct DISPATCH ShipDispatch[] = {
    {"Aircraft", Aircraft_Event},
    {"Close", Close_Event},
    {"F44", F44_Event},
    {"F76", F76_Event},
    {"Help", Help_Event},
    {"Ordnance", Ordnance_Event},
    {"Print", Print_Event},
    {"Save", Save_Event},
    {NULL, NULL}                    /* terminator entry */
};
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_Ship.h *** */
/* *** Generated:  Jan 19 13:12:17 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Header file for panel:  Ship
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    The panel's name is changed (not title)
 * For panel:
 *   Ship
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_PAN_Ship                    /* prevent double include */
#define I_PAN_Ship    0

/* Vm objects and panel Id.  */
extern Id ShipTarget, ShipView, ShipId;

/* Dispatch table (global for calls to Wpt_NewPanel) */
extern struct DISPATCH ShipDispatch[];

/* Initialize ShipTarget and ShipView */
extern VOID Ship_Initialize_Panel ();

/* Create this panel and display it on the screen */
extern VOID Ship_Create_Panel ();

/* Destroy this panel and erase it from the screen */
extern VOID Ship_Destroy_Panel ();

/* Connect to this panel.  Create it or change it's state */
extern VOID Ship_Connect_Panel ();
#endif
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      BGEvents.c *** */
/* *** Generated:  Jan 19 09:08:04 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * This the main program of an application generated by the TAE Plus Code
 * Generator.
 *
 * REGENERATED:
 * This file is generated only once.
 *
 * NOTES:
 * To turn this into a real application, do the following:
 *
 * 1. Each panel that has event generating parameters is encapsulated by
 * a separate file, named by concatenating the string "pan_" with the
 * panel name (followed by a ".c"). Each parameter that you have defined
 * to be "event-generating", has an event handler procedure in the
 * appropriate panel file. Each handler has a name that is a
 * concatentation of the parameter name and the string "_Event". Add
 * application-dependent logic to each event handler. (As generated by
 * the WorkBench, each event handler simply logs the occurrence of the
 * event.)
 *
 * 2. To build the program, type "make". If the symbols TAEINC, ....,
 * are not defined, the TAE shell (source) scripts $TAE/bin/csh/taesetup
 * will define them.
 *
 * ADDITIONAL NOTES:
 * 1. Each event handler has two arguments: (a) the value vector
 * associated with the parameter and (b) the number of components. Note
 * that for scalar values, we pass the value as if it were a vector with
 * count 1.
 *
 * Though it's unlikely that you are interested in the value of a button
 * event parameter, the values are always passed to the event handler for
 * consistency.
 *
 * 2. You gain access to non-event parameters by calling the Vm package
 * using the targetId Vm objects that are created in
 * Initialize_All_Panels. There are macros defined in global.h to assist
 * in accessing values in Vm objects.
 *
 * To access panel Id, target, and view, of other panels, add an
 * "#include" statement for each appropriate panel header file.
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#include    "taeconf.inp"
#include    "wptinc.inp"
#include    "symtab.inc"
#include    "global.h"          /* Application globals */

Display *Default_Display;
BOOL   Application_Done = FALSE;

main (argc, argv)
```

```
FUNINT    argc;
TEXT      *argv[];

{
WptEvent   wptEvent;             /* event data */
CODE       eventType;

COUNT      termLines, termCols;
        CODE       termType,/*BERN*/ret;

/* PROGRAMMER NOTE:
 * add similar extern's for each resource file in this application
 */
extern VOID BGEvents_Initialize_All_Panels ();
extern VOID BGEvents_Create_Initial_Panels ();

struct DISPATCH          *dp;     /* working dispatch pointer */
IMPORT struct VARIABLE    *Vm_Find();
struct VARIABLE          *parmv;  /* pointer to event VARIABLE */

/* initialize terminal without clearing screen */
t_pinit (&termLines, &termCols, &termType);

/* permit upper/lowercase file names */
f_force_lower (FALSE);

Default_Display = Wpt_Init (NULL);

/* initialize resource file */
/* PROGRAMMER NOTE:
 * For each resource file in this application, calls to the appropriate
 * Initialize_All_Panels and Create_Initial_Panels must be added.
 */
BGEvents_Initialize_All_Panels ("/h/bglcss/scripts/gui/events/BGEvents.res");
BGEvents_Create_Initial_Panels ();

        /*BERN*/
        ret = Wpt_SetHelpStyle ("wpthelp.res");
        if (ret != SUCCESS)
          printf("Couldn't set help style\n");

/* main event loop */
/* PROGRAMMER NOTE:
 * use SET_APPLICATION_DONE in "quit" event handler to exit loop.
 * (SET_APPLICATION_DONE is defined in global.h)
 */
while (!Application_Done)
  {
  eventType = Wpt_NextEvent (&wptEvent); /* get next WPT event */

  switch (eventType)
    {
  case WPT_PARM_EVENT:

    /* Event has occurred from a Panel Parm. Lookup the event
     * in the dispatch table and call the associated event
     * handler function.
     */

    dp = (struct DISPATCH *) wptEvent.p_userContext;
    for (; (*dp).parmName != NULL; dp++)
```

```
            if (s_equal ((*dp).parmName, wptEvent.parmName))
              {
              parmv = Vm_Find (wptEvent.p_dataVm, wptEvent.parmName);
              (*(*dp).eventFunction)
                 ((*parmv).v_cvp, (*parmv).v_count);
              break;
              }
         break;

      case WPT_FILE_EVENT:

         /* PROGRAMMER NOTE:
          * Add code here to handle file events.
          * Use Wpt_AddEvent and Wpt_RemoveEvent to register and remove
          * event sources.
          */
         printf ("No EVENT_HANDLER for event from external source.\n");
         break;

      case WPT_WINDOW_EVENT:

         /* PROGRAMMER NOTE:
          * Add code here to handle window events.
          * WPT_WINDOW_EVENT can be caused by windows which you directly
          * create with X (not TAE panels), or by user acknowledgement
          * of a Wpt_PanelMessage (therefore no default print statement
          * is generated here).
          */
         break;

      case WPT_TIMEOUT_EVENT:

         /* PROGRAMMER NOTE:
          * Add code here to handle timeout events.
          * Use Wpt_SetTimeOut to register timeout events.
          */
         printf ("No EVENT_HANDLER for timeout event.\n");
         break;

      default:

         printf("Unknown WPT Event\n");
         break;
         }
      }                    /* end main event loop */

   Wpt_Finish();/* close down all display connections */

   /* PROGRAMMER NOTE:
    * Application has ended normally. Add application specific code to
    * close down your application
    */

   }                      /* end main */
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:     BGEvents_creat_init.c *** */
/* *** Generated:  Jan 19 09:08:04 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Displays all panels in the initial panel set of this resource file
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    A panel is added to the initial panel set
 *    A panel is deleted from the initial panel set
 * For the set of initial panels:
 *   BGEvents
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */
#include      "taeconf.inp"
#include      "wptinc.inp"
#include      "global.h"            /* Application globals */

/* One include for each panel in initial panel set */
#include      "pan_BGEvents.h"


FUNCTION VOID BGEvents_Create_Initial_Panels ()
  {
  /* Show panels */

  BGEvents_Create_Panel (NULL, WPT_PREFERRED);
  }
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      BGEvents_init_pan.c *** */
/* *** Generated:  Feb 5 14:09:32 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Initialize all panels in the resource file.
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    A panel is deleted
 *    A new panel is added
 *    A panel's name is changed (not title)
 * For the panels:
 *   AAWThret, Aircraft, AirData, ASW,     ASWOrd,  ASWThret, BGCrsSpd,
 *   BGEvents, BGShips, BGSSCom, BGSumCom, CloseAll, CommentL, CommList,
 *   Consol,  ConsolDa, ConsolRe, DelAcft, DelComm, DelEvent, DelOrd,
 *   DelShip, Dtg,    EditComm, EventLis, F44Fuel, F76Fuel, FuelTran,
 *   LatLong, OrdData, Ordnance, OrdTrans, OrdTrSel, PrintJob, Raid,
 *   RaidShip, SelBG,   SelOrd, SelShiOr, SelShip, SelSumm, SetStati,
 *   ShCrsSpd, Ship,   Shuttle, StatRes, Strike,  StrikeSh, Unrep,
 *   UnrepDat, UnrepRes, USumComm, USumOrd
 *
 * CHANGE LOG:
 *  5-Feb-93  Initially generated...TAE
 * ************************************************************************
 */

#include     "taeconf.inp"
#include     "wptinc.inp"
#include     "symtab.inc"
#include     "global.h"          /* Application globals */

/* One "include" for each panel in resource file */
#include     "pan_AAWThret.h"
#include     "pan_Aircraft.h"
#include     "pan_AirData.h"
#include     "pan_ASW.h"
#include     "pan_ASWOrd.h"
#include     "pan_ASWThret.h"
#include     "pan_BGCrsSpd.h"
#include     "pan_BGEvents.h"
#include     "pan_BGShips.h"
#include     "pan_BGSSCom.h"
#include     "pan_BGSumCom.h"
#include     "pan_CloseAll.h"
#include     "pan_CommentL.h"
#include     "pan_CommList.h"
#include     "pan_Consol.h"
#include     "pan_ConsolDa.h"
#include     "pan_ConsolRe.h"
#include     "pan_DelAcft.h"
#include     "pan_DelComm.h"
#include     "pan_DelEvent.h"
#include     "pan_DelOrd.h"
#include     "pan_DelShip.h"
#include     "pan_Dtg.h"
#include     "pan_EditComm.h"
#include     "pan_EventLis.h"
#include     "pan_F44Fuel.h"
#include     "pan_F76Fuel.h"
#include     "pan_FuelTran.h"
```

```
#include     "pan_LatLong.h"
#include     "pan_OrdData.h"
#include     "pan_Ordnance.h"
#include     "pan_OrdTrans.h"
#include     "pan_OrdTrSel.h"
#include     "pan_PrintJob.h"
#include     "pan_Raid.h"
#include     "pan_RaidShip.h"
#include     "pan_SelBG.h"
#include     "pan_SelOrd.h"
#include     "pan_SelShiOr.h"
#include     "pan_SelShip.h"
#include     "pan_SelSumm.h"
#include     "pan_SetStati.h"
#include     "pan_ShCrsSpd.h"
#include     "pan_Ship.h"
#include     "pan_Shuttle.h"
#include     "pan_StatRes.h"
#include     "pan_Strike.h"
#include     "pan_StrikeSh.h"
#include     "pan_Unrep.h"
#include     "pan_UnrepDat.h"
#include     "pan_UnrepRes.h"
#include     "pan_USumComm.h"
#include     "pan_USumOrd.h"


FUNCTION VOID BGEvents_Initialize_All_Panels (resfileSpec)
  TEXT      *resfileSpec;
  {
  extern Id  Co_Find ();
  extern Id  Co_New ();
  Id vmCollection ;

  /* read resource file */
  vmCollection = Co_New (P_ABORT);
  Co_ReadFile (vmCollection, resfileSpec, P_ABORT);

  /* initialize view and target Vm objects for each panel */
  AAWThret_Initialize_Panel (vmCollection);
  Aircraft_Initialize_Panel (vmCollection);
  AirData_Initialize_Panel (vmCollection);
  ASW_Initialize_Panel (vmCollection);
  ASWOrd_Initialize_Panel (vmCollection);
  ASWThret_Initialize_Panel (vmCollection);
  BGCrsSpd_Initialize_Panel (vmCollection);
  BGEvents_Initialize_Panel (vmCollection);
  BGShips_Initialize_Panel (vmCollection);
  BGSSCom_Initialize_Panel (vmCollection);
  BGSumCom_Initialize_Panel (vmCollection);
  CloseAll_Initialize_Panel (vmCollection);
  CommentL_Initialize_Panel (vmCollection);
  CommList_Initialize_Panel (vmCollection);
  Consol_Initialize_Panel (vmCollection);
  ConsolDa_Initialize_Panel (vmCollection);
  ConsolRe_Initialize_Panel (vmCollection);
  DelAcft_Initialize_Panel (vmCollection);
  DelComm_Initialize_Panel (vmCollection);
  DelEvent_Initialize_Panel (vmCollection);
  DelOrd_Initialize_Panel (vmCollection);
  DelShip_Initialize_Panel (vmCollection);
```

```
Dtg_Initialize_Panel (vmCollection);
EditComm_Initialize_Panel (vmCollection);
EventLis_Initialize_Panel (vmCollection);
F44Fuel_Initialize_Panel (vmCollection);
F76Fuel_Initialize_Panel (vmCollection);
FuelTran_Initialize_Panel (vmCollection);
LatLong_Initialize_Panel (vmCollection);
OrdData_Initialize_Panel (vmCollection);
Ordnance_Initialize_Panel (vmCollection);
OrdTrans_Initialize_Panel (vmCollection);
OrdTrSel_Initialize_Panel (vmCollection);
PrintJob_Initialize_Panel (vmCollection);
Raid_Initialize_Panel (vmCollection);
RaidShip_Initialize_Panel (vmCollection);
SelBG_Initialize_Panel (vmCollection);
SelOrd_Initialize_Panel (vmCollection);
SelShiOr_Initialize_Panel (vmCollection);
SelShip_Initialize_Panel (vmCollection);
SelSumm_Initialize_Panel (vmCollection);
SetStati_Initialize_Panel (vmCollection);
ShCrsSpd_Initialize_Panel (vmCollection);
Ship_Initialize_Panel (vmCollection);
Shuttle_Initialize_Panel (vmCollection);
StatRes_Initialize_Panel (vmCollection);
Strike_Initialize_Panel (vmCollection);
StrikeSh_Initialize_Panel (vmCollection);
Unrep_Initialize_Panel (vmCollection);
UnrepDat_Initialize_Panel (vmCollection);
UnrepRes_Initialize_Panel (vmCollection);
USumComm_Initialize_Panel (vmCollection);
USumOrd_Initialize_Panel (vmCollection);
}
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      Imakefile *** */
/* *** Generated:  Jan 13 11:28:36 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * This is the Imakefile of a C application generated by the TAE Plus
 * Code Generator.
 *
 * REGENERATED:
 * This file is generated only once.
 *
 * NOTES:
 * 1.  To build your application, type "make".  The Makefile generated
 * by the TAE code generator invokes imake using this Imakefile to
 * generate an application specific Makefile.
 *
 * 2.  If you change the name of your resource file or application, you
 * will need to either edit this file, or just delete it and regenerate
 * the code.
 *
 * 3.  Edit this file to include your application specific source files.
 * ************************************************************************
 */

#define GeneratedApplication

/* PROGRAMMER NOTE:
 * Add a line '#include "Imake.RESFILENAME"' for each resource file in
 * your application.
 */
#include "Imake.BGEvents"


/* PROGRAMMER NOTE:
 * Insert application specific build parameters.  These override
 * definitions in the configuration files in $TAE/config.
 */
    CDEBUGFLAGS =
   LDDEBUGFLAGS =
     APP_CFLAGS =
  APP_LOAD_FLAGS =
   APP_LINKLIBS = -L/h/Nauticus/libs -lVids
    APP_DEPLIBS = $(DEPLIBS)
  APP_CINCLUDES = -I$(TAEINC)\
                  -I/h/Nauticus/include/vids/Vids.h\
                  -I/h/bglcss/scripts/gui/events/BGEventsLib.h\
                  -I/h/bglcss/scripts/gui/events/bg.h
       PROGRAM = BGEvents

/* PROGRAMMER NOTE:
 * Add $(SRCS_RESFILENAME) and $(OBJS_RESFILENAME) for each resource file
 * in your application.
 */
       GENSRCS = $(PROGRAM).c $(SRCS_BGEvents)
       GENOBJS = $(PROGRAM).o $(OBJS_BGEvents)

/* PROGRAMMER NOTE:
 * Add your application specific srcs and object files (that are not
 * generated by the code generator) here.
 */
       APPSRCS = /h/bglcss/scripts/gui/events/bg.c\
```

```
                /h/bglcss/scripts/gui/events/BGEventsLib.c
        APPOBJS = /h/bglcss/scripts/gui/events/bg.o\
            /h/bglcss/scripts/gui/events/BGEventsLib.o


/* Macro (defined in TAEmake.tmpl) to generate Makefile targets.
 */
CApplication($(PROGRAM))
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_BGCrsSpd.c *** */
/* *** Generated:  Jan 19 15:25:41 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * This file encapsulates the TAE Plus panel:  BGCrsSpd
 * These routines enable panel initialization, creation, and destruction.
 * Access to these routines from other files is enabled by inserting
 * '#include "pan_BGCrsSpd.h"'. For more advanced manipulation of the panel
 * using the TAE routines, the panel's Id, Target, and View are provided.
 * ************************************************************************
 */

#include      "taeconf.inp"
#include      "wptinc.inp"
#include      "global.h"           /* Application globals */
#include      "pan_BGCrsSpd.h"

/* One "include" for each connected panel */
#include      "pan_Dtg.h"

/*BERN*/
#include  "pan_SelBG.h"
#include  "pan_BGEvents.h"

/*BERN*/
extern          BGEVENT* SaveBGEvents();
extern          BGEVENT* GetBGEvents();
extern          BGEVENT* MakeBGEvent();
extern          BGEVENT* InsertBGEvent();
extern          int GetBG();
extern          int GetBGs();
extern          int dtg();
extern          int validdtg();
extern          BGHEADER* MakeBGHeader();
extern          BGHEADER* GetBGHeaders();
extern          BGHEADER* InsertBGHeader();
extern          BGHEADER* SaveBGHeaders();

Id BGCrsSpdTarget, BGCrsSpdView, BGCrsSpdId;
/* BGCrsSpdDispatch is defined at the end of this file */

/* ************************************************************************
 * Initialize the view and target of this panel.
 */
FUNCTION VOID BGCrsSpd_Initialize_Panel (vmCollection)
  Id vmCollection;
  {
  Id  Co_Find ();

  BGCrsSpdView = Co_Find (vmCollection, "BGCrsSpd_v");
  BGCrsSpdTarget = Co_Find (vmCollection, "BGCrsSpd_t");
  }


/* ************************************************************************
 * Create the panel object and display it on the screen.
 */
FUNCTION VOID BGCrsSpd_Create_Panel (relativeWindow, flags)
  Window    relativeWindow;
  COUNT     flags;
```

```
  {
  if (BGCrsSpdId)
    printf ("Panel (BGCrsSpd) is already displayed.\n");
  else
    BGCrsSpdId = Wpt_NewPanel(Default_Display, BGCrsSpdTarget,
    BGCrsSpdView, relativeWindow, BGCrsSpdDispatch, flags);

  }

/* ***********************************************************************
 * Erases a panel from the screen and de-allocate the associated panel
 * object.
 */
FUNCTION VOID BGCrsSpd_Destroy_Panel ()
  {
  Wpt_PanelErase(BGCrsSpdId);
  BGCrsSpdId=0;
  }

/* ***********************************************************************
 * Connect to this panel. Create it or change it's state.
 */
FUNCTION VOID BGCrsSpd_Connect_Panel (relativeWindow, flags)
  Window     relativeWindow;
  COUNT      flags;
  {
  if (BGCrsSpdId)
    Wpt_SetPanelState (BGCrsSpdId, flags);
  else
    BGCrsSpd_Create_Panel (relativeWindow, flags);
  }

/* ***********************************************************************
 * Handle event from parameter: AddEvent
 */
EVENT_HANDLER AddEvent_Event (value, count)
TEXT      *value[];           /* string pointers */
FUNINT    count;              /* num of values  */
{
  /*BERN*/
  BGInfo              BGs[MAXBGS];
  int                 BGIndex = 0;
  int                 BGHeaderIndex = 0;
  BGHEADER*           NewHeader;
  BGHEADER*           HeadHeader;
  BGEVENT*            NewEvent;
  BGEVENT*            HeadEvent;

  HeadEvent = (BGEVENT* ) malloc(sizeof (struct BGEvent ));

  HeadEvent->DTG = 0;

  HeadHeader = (BGHEADER* ) malloc(sizeof (struct BGHeader ));

  HeadHeader->DTG = 0;

  BGIndex = GetBGs(BGs);
```

```
  BGIndex = GetBG(StringParm(BGEventsTarget,"BGList"), BGIndex);

  HeadEvent = GetBGEvents(BGIndex);

  if (HeadEvent->DTG == 0){
          HeadEvent = (BGEVENT* ) malloc(sizeof (struct BGEvent ));
  }

  NewEvent = MakeBGEvent(0, dtg(StringParm(BGCrsSpdTarget,"Dtg")),
  BGCourseSpeed,Orphan,All,Low,100,
  RealParm(BGCrsSpdTarget,"Course"),
  RealParm(BGCrsSpdTarget, "Speed"));

  NewHeader=MakeBGHeader(BGCourseSpeed,
  StringParm(BGCrsSpdTarget,"Dtg"),
  RealParm(BGCrsSpdTarget,"Course"),
  RealParm(BGCrsSpdTarget, "Speed"));

  HeadHeader = GetBGHeaders(BGIndex);

  HeadHeader = InsertBGHeader(HeadHeader, NewHeader);

  SaveBGHeaders(BGIndex, HeadHeader);

  HeadEvent = InsertBGEvent(HeadEvent, NewEvent);

  SaveBGEvents(BGIndex, HeadEvent);

  free(HeadEvent);

  free(NewEvent);

  /* Begin generated code for Connection */
  BGCrsSpd_Destroy_Panel ();
  /* End generated code for Connection */
}


/* ***************************************************************************
 * Handle event from parameter:  Close
 */
EVENT_HANDLER Close_Event (value, count)
  TEXT      *value[];           /* string pointers */
  FUNINT    count;              /* num of values   */
  {
  /* Begin generated code for Connection */
  BGCrsSpd_Destroy_Panel ();
  /* End generated code for Connection */
  }


/* ***************************************************************************
 * Handle event from parameter:  Dtg
 */
EVENT_HANDLER Dtg_Event (value, count)
TEXT      *value[];           /* string pointers */
FUNINT    count;              /* num of values   */
{
  int DtgInteger;
  int* pointer;
```

```
    (int*) DtgInteger = pointer;

    DtgInteger = dtg(StringParm(BGCrsSpdTarget, "Dtg"));

    if (validdtg((StringParm(BGCrsSpdTarget, "Dtg")), pointer) == 0){
            /* Begin generated code for Connection */
            Dtg_Connect_Panel (NULL, WPT_PREFERRED);
            /* End generated code for Connection */
    }
}

/* ************************************************************************
 * Handle event from parameter:  Help
 */
EVENT_HANDLER Help_Event (value, count)
    TEXT      *value[];              /* string pointers */
    FUNINT    count;                 /* num of values   */
    {
    }


struct DISPATCH BGCrsSpdDispatch[] = {
    {"AddEvent", AddEvent_Event},
    {"Close", Close_Event},
    {"Dtg", Dtg_Event},
    {"Help", Help_Event},
    {NULL, NULL}                     /* terminator entry */
};
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_BGCrsSpd.h *** */
/* *** Generated:  Jan 19 09:08:04 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Header file for panel:  BGCrsSpd
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    The panel's name is changed (not title)
 * For panel:
 *   BGCrsSpd
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_PAN_BGCrsSpd                  /* prevent double include */
#define I_PAN_BGCrsSpd     0

/* Vm objects and panel Id.  */
extern Id BGCrsSpdTarget, BGCrsSpdView, BGCrsSpdId;

/* Dispatch table (global for calls to Wpt_NewPanel) */
extern struct DISPATCH BGCrsSpdDispatch[];

/* Initialize BGCrsSpdTarget and BGCrsSpdView */
extern VOID BGCrsSpd_Initialize_Panel ();

/* Create this panel and display it on the screen */
extern VOID BGCrsSpd_Create_Panel ();

/* Destroy this panel and erase it from the screen */
extern VOID BGCrsSpd_Destroy_Panel ();

/* Connect to this panel.  Create it or change it's state */
extern VOID BGCrsSpd_Connect_Panel ();
#endif
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_BGEvents.c *** */
/* *** Generated:  Feb 8 11:33:14 1993 *** */
/* ****************************************************************************
 * PURPOSE:
 * This file encapsulates the TAE Plus panel:  BGEvents
 * These routines enable panel initialization, creation, and destruction.
 * Access to these routines from other files is enabled by inserting
 * '#include "pan_BGEvents.h"'. For more advanced manipulation of the panel
 * using the TAE routines, the panel's Id, Target, and View are provided.
 *
 * For the panel items:
 *   AAWThreatLevel,  ASWEvent,     ASWThreatLevel,  BGCourseSpeed,
 *   Close,           CommentList,  Consol,          EventList,
 *   FuelTransfer,    Help,         OrdnanceTransfe, Predict,
 *   Raid,            SetStation,   ShipCourseSpeed, ShipList,
 *   Strike,          Unrep
 *
 * CHANGE LOG:
 *  8-Feb-93   Initially generated...TAE
 * ****************************************************************************
 */

#include     "taeconf.inp"
#include     "wptinc.inp"
#include     "global.h"          /* Application globals */
#include     "pan_BGEvents.h"

/* One "include" for each connected panel */
#include     "pan_AAWThret.h"
#include     "pan_ASW.h"
#include     "pan_ASWThret.h"
#include     "pan_BGCrsSpd.h"
#include     "pan_CloseAll.h"
#include     "pan_CommentL.h"
#include     "pan_Consol.h"
#include     "pan_EventLis.h"
#include     "pan_FuelTran.h"
#include     "pan_OrdTrSel.h"
#include     "pan_SelSumm.h"
#include     "pan_Raid.h"
#include     "pan_SetStati.h"
#include     "pan_ShCrsSpd.h"
#include     "pan_BGShips.h"
#include     "pan_Strike.h"
#include     "pan_Unrep.h"

/*BERN*/
extern       int GetBGs();
extern       void ShowBGs();


Id BGEventsTarget, BGEventsView, BGEventsId;
/* BGEventsDispatch is defined at the end of this file */

/* ****************************************************************************
 * Initialize the view and target of this panel.
 */
FUNCTION VOID BGEvents_Initialize_Panel (vmCollection)
  Id vmCollection;
  {
```

```
  Id Co_Find ();

  BGEventsView = Co_Find (vmCollection, "BGEvents_v");
  BGEventsTarget = Co_Find (vmCollection, "BGEvents_t");
  }

/* **********************************************************************
 * Create the panel object and display it on the screen.
 */
FUNCTION VOID BGEvents_Create_Panel (relativeWindow, flags)
  Window     relativeWindow;
  COUNT      flags;
  {
  /*BERN*/
  BGInfo    BGs[MAXBGS];

  if (BGEventsId)
    printf ("Panel (BGEvents) is already displayed.\n");
  else
    BGEventsId =Wpt_NewPanel (Default_Display, BGEventsTarget, BGEventsView,
                  relativeWindow, BGEventsDispatch, flags);
        GetBGs(BGs);

        ShowBGs(BGEventsId,"BGList");

  }

/* **********************************************************************
 * Erases a panel from the screen and de-allocate the associated panel
 * object.
 */
FUNCTION VOID BGEvents_Destroy_Panel ()
  {
  Wpt_PanelErase(BGEventsId);
  BGEventsId=0;
  }

/* **********************************************************************
 * Connect to this panel. Create it or change it's state.
 */
FUNCTION VOID BGEvents_Connect_Panel (relativeWindow, flags)
  Window     relativeWindow;
  COUNT      flags;
  {
  if (BGEventsId)
    Wpt_SetPanelState (BGEventsId, flags);
  else
    BGEvents_Create_Panel (relativeWindow, flags);
  }

/* **********************************************************************
 * Handle event from parameter: AAWThreatLevel
 */
EVENT_HANDLER AAWThreatLevel_Event (value, count)
  TEXT      *value[];          /* string pointers */
  FUNINT    count;             /* num of values  */
  {
  /* Begin generated code for Connection */
```

```
  AAWThret_Connect_Panel (NULL, WPT_PREFERRED);
  /* End generated code for Connection */
  }

/* ************************************************************************
 * Handle event from parameter: ASWEvent
 */
EVENT_HANDLER ASWEvent_Event (value, count)
  TEXT      *value[];              /* string pointers */
  FUNINT     count;                /* num of values  */
  {
  /* Begin generated code for Connection */
  ASW_Connect_Panel (NULL, WPT_PREFERRED);
  /* End generated code for Connection */
  }

/* ************************************************************************
 * Handle event from parameter: ASWThreatLevel
 */
EVENT_HANDLER ASWThreatLevel_Event (value, count)
  TEXT      *value[];              /* string pointers */
  FUNINT     count;                /* num of values  */
  {
  /* Begin generated code for Connection */
  ASWThret_Connect_Panel (NULL, WPT_PREFERRED);
  /* End generated code for Connection */
  }

/* ************************************************************************
 * Handle event from parameter: BGCourseSpeed
 */
EVENT_HANDLER BGCourseSpeed_Event (value, count)
  TEXT      *value[];              /* string pointers */
  FUNINT     count;                /* num of values  */
  {
  /* Begin generated code for Connection */
  BGCrsSpd_Connect_Panel (NULL, WPT_PREFERRED);
  /* End generated code for Connection */
  }

/* ************************************************************************
 * Handle event from parameter: Close
 */
EVENT_HANDLER Close_Event (value, count)
  TEXT      *value[];              /* string pointers */
  FUNINT     count;                /* num of values  */
  {
  /* Begin generated code for Connection */
  CloseAll_Connect_Panel (NULL, WPT_PREFERRED);
  /* End generated code for Connection */
  }

/* ************************************************************************
 * Handle event from parameter: CommentList
 */
EVENT_HANDLER CommentList_Event (value, count)
```

```
   TEXT      *value[];              /* string pointers */
   FUNINT     count;                /* num of values   */
   {
   /* Begin generated code for Connection */
   CommentL_Connect_Panel (NULL, WPT_PREFERRED);
   /* End generated code for Connection */
   }


/* ************************************************************************
 * Handle event from parameter:  Consol
 */
EVENT_HANDLER Consol_Event (value, count)
   TEXT      *value[];              /* string pointers */
   FUNINT     count;                /* num of values   */
   {
   /* Begin generated code for Connection */
   Consol_Connect_Panel (NULL, WPT_PREFERRED);
   /* End generated code for Connection */
   }


/* ************************************************************************
 * Handle event from parameter:  EventList
 */
EVENT_HANDLER EventList_Event (value, count)
   TEXT      *value[];              /* string pointers */
   FUNINT     count;                /* num of values   */
   {
   /* Begin generated code for Connection */
   EventLis_Connect_Panel (NULL, WPT_PREFERRED);
   /* End generated code for Connection */
   }


/* ************************************************************************
 * Handle event from parameter:  FuelTransfer
 */
EVENT_HANDLER FuelTransfer_Event (value, count)
   TEXT      *value[];              /* string pointers */
   FUNINT     count;                /* num of values   */
   {
   /* Begin generated code for Connection */
   FuelTran_Connect_Panel (NULL, WPT_PREFERRED);
   /* End generated code for Connection */
   }


/* ************************************************************************
 * Handle event from parameter:  Help
 */
EVENT_HANDLER Help_Event (value, count)
   TEXT      *value[];              /* string pointers */
   FUNINT     count;                /* num of values   */
   {
   /* Begin generated code for Connection */
   EventLis_Connect_Panel (NULL, WPT_PREFERRED);
   /* End generated code for Connection */
   }
```

```
/* *************************************************************************
 * Handle event from parameter:  OrdnanceTransfe
 */
EVENT_HANDLER OrdnanceTransfe_Event (value, count)
   TEXT       *value[];              /* string pointers */
   FUNINT     count;                  /* num of values   */
   {
   /* Begin generated code for Connection */
   OrdTrSel_Connect_Panel (NULL, WPT_PREFERRED);
   /* End generated code for Connection */
   }


/* *************************************************************************
 * Handle event from parameter:  Predict
 */
EVENT_HANDLER Predict_Event (value, count)
   TEXT       *value[];              /* string pointers */
   FUNINT     count;                  /* num of values   */
   {
   /* Begin generated code for Connection */
   SelSumm_Connect_Panel (NULL, WPT_PREFERRED);
   /* End generated code for Connection */
   }


/* *************************************************************************
 * Handle event from parameter:  Raid
 */
EVENT_HANDLER Raid_Event (value, count)
   TEXT       *value[];              /* string pointers */
   FUNINT     count;                  /* num of values   */
   {
   /* Begin generated code for Connection */
   Raid_Connect_Panel (NULL, WPT_PREFERRED);
   /* End generated code for Connection */
   }


/* *************************************************************************
 * Handle event from parameter:  SetStation
 */
EVENT_HANDLER SetStation_Event (value, count)
   TEXT       *value[];              /* string pointers */
   FUNINT     count;                  /* num of values   */
   {
   /* Begin generated code for Connection */
   SetStati_Connect_Panel (NULL, WPT_PREFERRED);
   /* End generated code for Connection */
   }


/* *************************************************************************
 * Handle event from parameter:  ShipCourseSpeed
 */
EVENT_HANDLER ShipCourseSpeed_Event (value, count)
   TEXT       *value[];              /* string pointers */
   FUNINT     count;                  /* num of values   */
   {
   /* Begin generated code for Connection */
   ShCrsSpd_Connect_Panel (NULL, WPT_PREFERRED);
```

```
  /* End generated code for Connection */
  }


/* *************************************************************************
 * Handle event from parameter:  ShipList
 */
EVENT_HANDLER ShipList_Event (value, count)
  TEXT      *value[];              /* string pointers */
  FUNINT    count;                 /* num of values   */
  {
  /* Begin generated code for Connection */
  BGShips_Connect_Panel (NULL, WPT_PREFERRED);
  /* End generated code for Connection */
  }


/* *************************************************************************
 * Handle event from parameter:  Strike
 */
EVENT_HANDLER Strike_Event (value, count)
  TEXT      *value[];              /* string pointers */
  FUNINT    count;                 /* num of values   */
  {
  /* Begin generated code for Connection */
  Strike_Connect_Panel (NULL, WPT_PREFERRED);
  /* End generated code for Connection */
  }


/* *************************************************************************
 * Handle event from parameter:  Unrep
 */
EVENT_HANDLER Unrep_Event (value, count)
  TEXT      *value[];              /* string pointers */
  FUNINT    count;                 /* num of values   */
  {
  /* Begin generated code for Connection */
  Unrep_Connect_Panel (NULL, WPT_PREFERRED);
  /* End generated code for Connection */
  }


struct DISPATCH BGEventsDispatch[] = {
  {"AAWThreatLevel", AAWThreatLevel_Event},
  {"ASWEvent", ASWEvent_Event},
  {"ASWThreatLevel", ASWThreatLevel_Event},
  {"BGCourseSpeed", BGCourseSpeed_Event},
  {"Close", Close_Event},
  {"CommentList", CommentList_Event},
  {"Consol", Consol_Event},
  {"EventList", EventList_Event},
  {"FuelTransfer", FuelTransfer_Event},
  {"Help", Help_Event},
  {"OrdnanceTransfe", OrdnanceTransfe_Event},
  {"Predict", Predict_Event},
  {"Raid", Raid_Event},
  {"SetStation", SetStation_Event},
  {"ShipCourseSpeed", ShipCourseSpeed_Event},
  {"ShipList", ShipList_Event},
```

```
    {"Strike", Strike_Event},
    {"Unrep", Unrep_Event},
    {NULL, NULL}                    /* terminator entry */
};
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_BGEvents.h *** */
/* *** Generated:  Jan 19 09:08:04 1993 *** */
/* **********************************************************************
 * PURPOSE:
 * Header file for panel:  BGEvents
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    The panel's name is changed (not title)
 * For panel:
 *   BGEvents
 *
 * CHANGE LOG:
 * 19-Jan-93   Initially generated...TAE
 * **********************************************************************
 */

#ifndef I_PAN_BGEvents                  /* prevent double include */
#define I_PAN_BGEvents     0

/* Vm objects and panel Id.  */
extern Id BGEventsTarget, BGEventsView, BGEventsId;

/* Dispatch table (global for calls to Wpt_NewPanel) */
extern struct DISPATCH BGEventsDispatch[];

/* Initialize BGEventsTarget and BGEventsView */
extern VOID BGEvents_Initialize_Panel ();

/* Create this panel and display it on the screen */
extern VOID BGEvents_Create_Panel ();

/* Destroy this panel and erase it from the screen */
extern VOID BGEvents_Destroy_Panel ();

/* Connect to this panel.  Create it or change it's state */
extern VOID BGEvents_Connect_Panel ();
#endif
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      Overview.c *** */
/* *** Generated:  Jan 13 13:52:27 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * This the main program of an application generated by the TAE Plus Code
 * Generator.
 *
 * REGENERATED:
 * This file is generated only once.
 *
 * NOTES:
 * To turn this into a real application, do the following:
 *
 * 1. Each panel that has event generating parameters is encapsulated by
 * a separate file, named by concatenating the string "pan_" with the
 * panel name (followed by a ".c"). Each parameter that you have defined
 * to be "event-generating", has an event handler procedure in the
 * appropriate panel file. Each handler has a name that is a
 * concatentation of the parameter name and the string "_Event". Add
 * application-dependent logic to each event handler. (As generated by
 * the WorkBench, each event handler simply logs the occurrence of the
 * event.)
 *
 * 2. To build the program, type "make". If the symbols TAEINC, ...,
 * are not defined, the TAE shell (source) scripts $TAE/bin/csh/taesetup
 * will define them.
 *
 * ADDITIONAL NOTES:
 * 1. Each event handler has two arguments: (a) the value vector
 * associated with the parameter and (b) the number of components. Note
 * that for scalar values, we pass the value as if it were a vector with
 * count 1.
 *
 * Though it's unlikely that you are interested in the value of a button
 * event parameter, the values are always passed to the event handler for
 * consistency.
 *
 * 2. You gain access to non-event parameters by calling the Vm package
 * using the targetId Vm objects that are created in
 * Initialize_All_Panels. There are macros defined in global.h to assist
 * in accessing values in Vm objects.
 *
 * To access panel Id, target, and view, of other panels, add an
 * "#include" statement for each appropriate panel header file.
 *
 * CHANGE LOG:
 * 13-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#include      "taeconf.inp"
#include      "wptinc.inp"
#include      "symtab.inc"
#include      "global.h"          /* Application globals */

Display *Default_Display;
BOOL   Application_Done = FALSE;

main (argc, argv)
```

```
FUNINT    argc;
TEXT      *argv[];

{
WptEvent  wptEvent;             /* event data */
CODE      eventType;

COUNT     termLines, termCols;
CODE      termType, ret;/*BERN*/

/* PROGRAMMER NOTE:
 * add similar extern's for each resource file in this application
 */
extern VOID Overview_Initialize_All_Panels ();
extern VOID Overview_Create_Initial_Panels ();

struct DISPATCH          *dp;    /* working dispatch pointer */
IMPORT struct VARIABLE     *Vm_Find();
struct VARIABLE          *parmv;   /* pointer to event VARIABLE */

/* initialize terminal without clearing screen */
t_pinit (&termLines, &termCols, &termType);

/* permit upper/lowercase file names */
f_force_lower (FALSE);

Default_Display = Wpt_Init (NULL);

/* initialize resource file */
/* PROGRAMMER NOTE:
 * For each resource file in this application, calls to the appropriate
 * Initialize_All_Panels and Create_Initial_Panels must be added.
 */
Overview_Initialize_All_Panels ("Overview.res");
Overview_Create_Initial_Panels ();

/* main event loop */
/* PROGRAMMER NOTE:
 * use SET_APPLICATION_DONE in "quit" event handler to exit loop.
 * (SET_APPLICATION_DONE is defined in global.h)
 */

        ret = Wpt_SetHelpStyle("wpthelp.res");
while (!Application_Done)
  {
  eventType = Wpt_NextEvent (&wptEvent); /* get next WPT event */

  switch (eventType)
    {
  case WPT_PARM_EVENT:

    /* Event has occurred from a Panel Parm.  Lookup the event
     * in the dispatch table and call the associated event
     * handler function.
     */

    dp = (struct DISPATCH *) wptEvent.p_userContext;
    for (;  (*dp).parmName != NULL;  dp++)
      if (s_equal ((*dp).parmName, wptEvent.parmName))
        {
        parmv = Vm_Find (wptEvent.p_dataVm, wptEvent.parmName);
```

```
            (*(*dp).eventFunction)
              ((*parmv).v_cvp, (*parmv).v_count);
            break;
            }
        break;

      case WPT_FILE_EVENT:

        /* PROGRAMMER NOTE:
         * Add code here to handle file events.
         * Use Wpt_AddEvent and Wpt_RemoveEvent to register and remove
         * event sources.
         */
        printf ("No EVENT_HANDLER for event from external source.\n");
        break;

      case WPT_WINDOW_EVENT:

        /* PROGRAMMER NOTE:
         * Add code here to handle window events.
         * WPT_WINDOW_EVENT can be caused by windows which you directly
         * create with X (not TAE panels), or by user acknowledgement
         * of a Wpt_PanelMessage (therefore no default print statement
         * is generated here).
         */
        break;

      case WPT_TIMEOUT_EVENT:

        /* PROGRAMMER NOTE:
         * Add code here to handle timeout events.
         * Use Wpt_SetTimeOut to register timeout events.
         */
        printf ("No EVENT_HANDLER for timeout event.\n");
        break;

      default:

        printf("Unknown WPT Event\n");
        break;
        }
      }                         /* end main event loop */

    Wpt_Finish();/* close down all display connections */

    /* PROGRAMMER NOTE:
     * Application has ended normally.  Add application specific code to
     * close down your application
     */

    }                           /* end main */
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:     Overview_creat_init.c *** */
/* *** Generated:  Jan 13 13:52:27 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Displays all panels in the initial panel set of this resource file
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    A panel is added to the initial panel set
 *    A panel is deleted from the initial panel set
 * For the set of initial panels:
 *   Overview
 *
 * CHANGE LOG:
 * 13-Jan-93   Initially generated...TAE
 * ************************************************************************
 */
#include      "taeconf.inp"
#include      "wptinc.inp"
#include      "global.h"          /* Application globals */

/* One include for each panel in initial panel set */
#include      "pan_Overview.h"


FUNCTION VOID Overview_Create_Initial_Panels ()
  {
  /* Show panels */

  Overview_Create_Panel (NULL, WPT_PREFERRED);
  }
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      Overview_init_pan.c *** */
/* *** Generated:  Jan 13 13:52:27 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Initialize all panels in the resource file.
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    A panel is deleted
 *    A new panel is added
 *    A panel's name is changed (not title)
 * For the panels:
 *   Overview
 *
 * CHANGE LOG:
 * 13-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#include      "taeconf.inp"
#include      "wptinc.inp"
#include      "symtab.inc"
#include      "global.h"          /* Application globals */

/* One "include" for each panel in resource file */
#include      "pan_Overview.h"


FUNCTION VOID Overview_Initialize_All_Panels (resfileSpec)
    TEXT       *resfileSpec;
    {
    extern Id  Co_Find ();
    extern Id  Co_New ();
    Id vmCollection ;

    /* read resource file */
    vmCollection = Co_New (P_ABORT);
    Co_ReadFile (vmCollection, resfileSpec, P_ABORT);

    /* initialize view and target Vm objects for each panel */
    Overview_Initialize_Panel (vmCollection);
    }
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:    pan_Overview.c *** */
/* *** Generated:  Jan 14 08:16:26 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * This file encapsulates the TAE Plus panel:  Overview
 * These routines enable panel initialization, creation, and destruction.
 * Access to these routines from other files is enabled by inserting
 * '#include "pan_Overview.h"'. For more advanced manipulation of the panel
 * using the TAE routines, the panel's Id, Target, and View are provided.
 *
 * NOTES:
 * For each parameter that you have defined to be "event-generating" in
 * this panel, there is an event handler procedure below. Each handler
 * has a name that is a concatenation of the parameter name and "_Event".
 * Add application-dependent logic to each event handler. (As generated
 * by the WorkBench, each event handler simply logs the occurrence of the
 * event.)
 *
 * You may want to flag any changes you make to this file so that if you
 * regenerate this file, you can more easily cut and paste your
 * modifications back in. For example:
 *
 *   generated code ...
 *   /* (+) ADDED yourinitials * /
 *   your code
 *   /* (-) ADDED * /
 *   more generated code ...
 *
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *     The panel's name is changed (not title)
 * For panel:
 *   Overview
 *
 * The following WorkBench operations will also cause regeneration:
 *     An item is deleted
 *     A new item is added to this panel
 *     An item's name is changed (not title)
 *     An item's data type is changed
 *     An item's generates events flag is changed
 *     An item's valids changed (if item is type string and connected)
 *     An item's connection information is changed
 * For the panel items:
 *   BackUp,        Close,        Events,        Forward,
 *   Help,          Index,        SetUp
 *
 * CHANGE LOG:
 * 14-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#include    "taeconf.inp"
#include    "wptinc.inp"
#include    "global.h"          /* Application globals */
#include    "pan_Overview.h"

/* One "include" for each connected panel */

Id OverviewTarget, OverviewView, OverviewId;
```

```
/* OverviewDispatch is defined at the end of this file */

/* ************************************************************************
 * Initialize the view and target of this panel.
 */
FUNCTION VOID Overview_Initialize_Panel (vmCollection)
  Id vmCollection;
  {
  Id  Co_Find ();

  OverviewView = Co_Find (vmCollection, "Overview_v");
  OverviewTarget = Co_Find (vmCollection, "Overview_t");
  }


/* ************************************************************************
 * Create the panel object and display it on the screen.
 */
FUNCTION VOID Overview_Create_Panel (relativeWindow, flags)
  Window     relativeWindow;
  COUNT      flags;
  {
  if (OverviewId)
    printf ("Panel (Overview) is already displayed.\n");
  else
    OverviewId = Wpt_NewPanel (Default_Display, OverviewTarget, OverviewView,
                     relativeWindow, OverviewDispatch, flags);
  }


/* ************************************************************************
 * Erases a panel from the screen and de-allocate the associated panel
 * object.
 */
FUNCTION VOID Overview_Destroy_Panel ()
  {
  Wpt_PanelErase(OverviewId);
  OverviewId=0;
  }


/* ************************************************************************
 * Connect to this panel.  Create it or change it's state.
 */
FUNCTION VOID Overview_Connect_Panel (relativeWindow, flags)
  Window     relativeWindow;
  COUNT      flags;
  {
  if (OverviewId)
    Wpt_SetPanelState (OverviewId, flags);
  else
    Overview_Create_Panel (relativeWindow, flags);
  }


/* ************************************************************************
 * Handle event from parameter:  BackUp
 */
EVENT_HANDLER BackUp_Event (value, count)
  TEXT      *value[];              /* string pointers */
  FUNINT    count;                 /* num of values  */
```

```
    {
    /* Begin default generated code */

    printf ("Panel Overview, parm BackUp: value = %s\n",
          count>0 ? value[0] : "none");

    /* End default generated code */
    }


/* ***********************************************************************
 * Handle event from parameter:  Close
 */
EVENT_HANDLER Close_Event (value, count)
    TAEINT     value[];                /* integer vector  */
    FUNINT     count;                  /* num of values   */
    {
    /* Begin default generated code */

    printf ("Panel Overview, parm Close: value = %d\n",
          count>0 ? value[0] : 0);

    /* End default generated code */

    /* Begin generated code for Connection */

    Overview_Destroy_Panel ();
    SET_APPLICATION_DONE;

    /* End generated code for Connection */
    }


/* ***********************************************************************
 * Handle event from parameter:  Events
 */
EVENT_HANDLER Events_Event (value, count)
    TEXT      *value[];                /* string pointers */
    FUNINT     count;                  /* num of values   */
    {
    /* Begin default generated code */

    printf ("Panel Overview, parm Events: value = %s\n",
          count>0 ? value[0] : "none");

    /* End default generated code */
    }


/* ***********************************************************************
 * Handle event from parameter:  Forward
 */
EVENT_HANDLER Forward_Event (value, count)
    TEXT      *value[];                /* string pointers */
    FUNINT     count;                  /* num of values   */
    {
    /* Begin default generated code */

    printf ("Panel Overview, parm Forward: value = %s\n",
          count>0 ? value[0] : "none");
```

```
  /* End default generated code */
  }

/* ************************************************************************
 * Handle event from parameter:  Help
 */
EVENT_HANDLER Help_Event (value, count)
  TEXT       *value[];              /* string pointers */
  FUNINT     count;                 /* num of values   */
  {
  /* Begin default generated code */

  printf ("Panel Overview, parm Help: value = %s\n",
        count>0 ? value[0] : "none");

  /* End default generated code */
  }

/* ************************************************************************
 * Handle event from parameter:  Index
 */
EVENT_HANDLER Index_Event (value, count)
  TEXT       *value[];              /* string pointers */
  FUNINT     count;                 /* num of values   */
  {
  /* Begin default generated code */

  printf ("Panel Overview, parm Index: value = %s\n",
        count>0 ? value[0] : "none");

  /* End default generated code */
  }

/* ************************************************************************
 * Handle event from parameter:  SetUp
 */
EVENT_HANDLER SetUp_Event (value, count)
  TEXT       *value[];              /* string pointers */
  FUNINT     count;                 /* num of values   */
  {
  /* Begin default generated code */

  printf ("Panel Overview, parm SetUp: value = %s\n",
        count>0 ? value[0] : "none");

  /* End default generated code */
  }


struct DISPATCH OverviewDispatch[] = {
  {"BackUp", BackUp_Event},
  {"Close", Close_Event},
  {"Events", Events_Event},
  {"Forward", Forward_Event},
  {"Help", Help_Event},
  {"Index", Index_Event},
  {"SetUp", SetUp_Event},
```

```
    {NULL, NULL}                    /* terminator entry */
};
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:      pan_Overview.h *** */
/* *** Generated:  Jan 13 13:52:27 1993 *** */
/* ************************************************************************
 * PURPOSE:
 * Header file for panel:  Overview
 *
 * REGENERATED:
 * The following WorkBench operations will cause regeneration of this file:
 *    The panel's name is changed (not title)
 * For panel:
 *   Overview
 *
 * CHANGE LOG:
 * 13-Jan-93   Initially generated...TAE
 * ************************************************************************
 */

#ifndef I_PAN_Overview                    /* prevent double include */
#define I_PAN_Overview    0

/* Vm objects and panel Id.  */
extern Id OverviewTarget, OverviewView, OverviewId;

/* Dispatch table (global for calls to Wpt_NewPanel) */
extern struct DISPATCH OverviewDispatch[];

/* Initialize OverviewTarget and OverviewView */
extern VOID Overview_Initialize_Panel ();

/* Create this panel and display it on the screen */
extern VOID Overview_Create_Panel ();

/* Destroy this panel and erase it from the screen */
extern VOID Overview_Destroy_Panel ();

/* Connect to this panel.  Create it or change it's state */
extern VOID Overview_Connect_Panel ();
#endif
```

APPENDIX C. BGLCSS 2.0 C++ PROGRAM LISTING

C++ files generated by TAE Plus contained is this appendix:

BGSetup.cc
BGSetup.h
BGSetup_creat_init.cc
BGSetup_init_pan.cc
Imakefile
item_SetUpBGs.h
pan_SetUpBGs.cc
pan_SetUpBGs.h

```
// *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 ***
// *** File:        BGSetup.cc ***
// *** Generated:   Mar 10 07:51:07 1993 ***
/////////////////////////////////////////////////////////////////////////
// PURPOSE:
// This the main program of an application generated by the TAE Plus
// Code
// Generator.
//
// REGENERATED:
// This file is generated only once.
//
// NOTES:
// To turn this into a real application, do the following:
//
// 1.  Each panel that has event generating parameters has a class
// definition file, named by concatenating the string "pan_" with the
// panel name followed by a ".h".  The methods are in a separate file,
// named by concatenating the string "pan_" with the panel name
// followed by a ".cc".  Each item has a class definition in a file
// named by concatenating the string "item_" with the panel name
// followed by a ".h".  Each parameter that you have defined to be
// "event-generating", has an event handler method in the appropriate
// panel file.  Add application-dependent logic to each event handler.
// (As generated by the WorkBench, each event handler simply logs the
// occurrence of the event.)
//
// 2.  To build the program, type "make".  If the symbols TAEINC, ...,
// are not defined, the TAE shell (source) scripts $TAE/bin/csh///
// taesetup
// will define them.
//
// ADDITIONAL NOTES:
// 1.  Each event handler has one argument: the actual wptevent
//
// 2.  You gain access to non-event parameters by calling the TaeVar
// and TaeVarTable methods using the instances of TaeVar and
// TaeVarTable associated with the panel.
//
// To access other panels, add an "#include" statement for each
// appropriate panel header file.
//
// CHANGE LOG:
// 10-Mar-93    Initially generated...TAE
/////////////////////////////////////////////////////////////////////////

#include <stream.h>
#include <taepanel.h>
#include <taeitem.h>
#include <taevm.h>
//
// PROGRAMMER NOTE:
// For each resource file in this application, add the appropriate
// header file
//
#include "BGSetup.h"

Display * defaultDisplay;
TaeEventHandler *eventHandler;

main()
    {

    COUNT tlines, tcols;
    CODE  ttype;
    f_force_lower (FALSE);                  // permit upper/lowercase file names
    t_pinit (&tlines, &tcols, &ttype);      // initialize terminal pkg

    defaultDisplay = Wpt_CCInit ( NULL );

    //
    // PROGRAMMER NOTE:
    //  For each resource file in this application, add calls to the
    //  appropriate constructors
    //
    BGSetupResource *BGSetupR = new BGSetupResource();
```

```
eventHandler = new TaeEventHandler();

//
// PROGRAMMER NOTE:
//  For each resource file in this application, calls to the
//  appropriate Initialize_All_Panels and Create_Initial_Panels
//  method must be added.
//

//  Initialize all panel instances

BGSetupR->Initialize_All_Panels();

//  Create and display the initial panel set

BGSetupR->Create_Initial_Panels( *eventHandler );

eventHandler->ProcessEvents();      // Start event processing

Wpt_Finish();                       // Close all display connections
}
```

```
// *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 ***
// *** File:        BGSetup.h ***
// *** Generated:   Mar 10 07:51:07 1993 ***
//////////////////////////////////////////////////////////////////////
// PURPOSE:
// This header file encapsulates the TaeResource that corresponds with
// the resource file /h/bglcss/scripts/gui/setup/c++/BGSetup.res.res.
//
// REGENERATED:
// This file is generated only once.
//
// CHANGE LOG:
// 10-Mar-93    Initially generated...TAE
//////////////////////////////////////////////////////////////////////

#ifndef I_SIMPLE                        // prevent double include
#define I_SIMPLE0

#include <taepanel.h>
#include <taeitem.h>
#include <taevm.h>
//
// BGSetupResource contains methods that have implication on the
// resource file BGSetup.res.
//
class BGSetupResource : public TaeResource
    {
public:

    BGSetupResource () : ("/h/bglcss/scripts/gui/setup/c++/BGSetup.res") {};
    ~BGSetupResource () {};

    void Initialize_All_Panels ();
    void Create_Initial_Panels ( const TaeEventHandler& );
    };

#endif
```

```
// *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 ***
// *** File:        BGSetup_creat_init.cc ***
// *** Generated:   Mar 10 07:51:07 1993 ***
////////////////////////////////////////////////////////////////////////
// PURPOSE:
// Displays all panels in the initial panel set of this resource file
//
// REGENERATED:
// The following WorkBench operations will cause regeneration of this
// file:
//     A panel is added to the initial panel set
//     A panel is deleted from the initial panel set
// For the set of initial panels:
//   SetUpBGs
//
// CHANGE LOG:
// 10-Mar-93    Initially generated...TAE
////////////////////////////////////////////////////////////////////////

#include <stream.h>
#include <taepanel.h>
#include <taeitem.h>
#include <taevm.h>
#include "BGSetup.h"

// One "include" for each panel in the initial panel set
#include        "pan_SetUpBGs.h"


void BGSetupResource::Create_Initial_Panels (const TaeEventHandler& eh)
    {
    SetUpBGsP->Show(eh);
    }
```

```
// *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 ***
// *** File:        BGSetup_init_pan.cc ***
// *** Generated:   Mar 10 07:51:07 1993 ***
////////////////////////////////////////////////////////////////////////
// PURPOSE:
// Initialize all panels in the resource file.
//
// REGENERATED:
// The following WorkBench operations will cause regeneration of this
// file:
//     A panel is deleted
//     A new panel is added
//     A panel's name is changed (not title)
// For the panels:
//   AcftLoad, AirData,  BGData,   BGShips,  CloseAll, DelBG,
//   DeleteSh, DelShip,  Dtg,      F44Fuel,  F76Fuel,  LackData,
//   NewBG,    OrdData,
//   OrdLoad,  OrdSel,   PrintJob, SaveNewB, SelBG,    SetUpBGs, Ship,
//
// CHANGE LOG:
// 10-Mar-93    Initially generated...TAE
////////////////////////////////////////////////////////////////////////
#include <stream.h>
#include <taepanel.h>
#include <taeitem.h>
#include <taevm.h>
#include "BGSetup.h"
// One "include" for each panel in resource file
#include        "pan_AcftLoad.h"
#include        "pan_AirData.h"
#include        "pan_BGData.h"
#include        "pan_BGShips.h"
#include        "pan_CloseAll.h"
#include        "pan_DelBG.h"
#include        "pan_DeleteSh.h"
#include        "pan_DelShip.h"
#include        "pan_Dtg.h"
#include        "pan_F44Fuel.h"
#include        "pan_F76Fuel.h"
#include        "pan_LackData.h"
#include        "pan_NewBG.h"
#include        "pan_OrdData.h"
#include        "pan_OrdLoad.h"
#include        "pan_OrdSel.h"
#include        "pan_PrintJob.h"
#include        "pan_SaveNewB.h"
#include        "pan_SelBG.h"
#include        "pan_SetUpBGs.h"
#include        "pan_Ship.h"
void BGSetupResource::Initialize_All_Panels ()
    {
    // Create an instance of all panels
    AcftLoadP = new AcftLoadC (Collection());
    AirDataP = new AirDataC (Collection());
    BGDataP = new BGDataC (Collection());
    BGShipsP = new BGShipsC (Collection());
    CloseAllP = new CloseAllC (Collection());
    DelBGP = new DelBGC (Collection());
    DeleteShP = new DeleteShC (Collection());
    DelShipP = new DelShipC (Collection());
    DtgP = new DtgC (Collection());
    F44FuelP = new F44FuelC (Collection());
    F76FuelP = new F76FuelC (Collection());
    LackDataP = new LackDataC (Collection());
    NewBGP = new NewBGC (Collection());
    OrdDataP = new OrdDataC (Collection());
    OrdLoadP = new OrdLoadC (Collection());
    OrdSelP = new OrdSelC (Collection());
    PrintJobP = new PrintJobC (Collection());
    SaveNewBP = new SaveNewBC (Collection());
    SelBGP = new SelBGC (Collection());
    SetUpBGsP = new SetUpBGsC (Collection());
    ShipP = new ShipC (Collection());
    }
```

```
/* *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 *** */
/* *** File:        Imakefile *** */
/* *** Generated:   Mar 10 07:51:07 1993 *** */
/* **********************************************************************
 * PURPOSE:
 * This is the Imakefile of a C++ application generated by the TAE Plus
 * Code Generator.
 *
 * REGENERATED:
 * This file is generated only once.
 *
 * NOTES:
 * 1.  To build your application, type "make".  The Makefile generated
 * by the TAE code generator invokes imake using this Imakefile to
 * generate an application specific Makefile.
 *
 * 2.  If you change the name of your resource file or application, you
 * will need to either edit this file, or just delete it and regenerate
 * the code.
 *
 * 3.  Edit this file to include your application specific source
 * files.
 * **********************************************************************
 */

#define GeneratedApplication

/* PROGRAMMER NOTE:
 * Add a line '#include "Imake.RESFILENAME"' for each resource file in
 * your application.
 */
#include "Imake.BGSetup"


/* PROGRAMMER NOTE:
 * Insert application specific build parameters.  These override
 * definitions in the configuration files in $TAE/config.
 */
    C++DEBUGFLAGS =
   LD++DEBUGFLAGS =
     APP_C++FLAGS =
   APP_LOAD_FLAGS =
     APP_LINKLIBS =-L/h/bglcss/CC2.1/SC1.0/libC.a
      APP_DEPLIBS = $(DEPLIBS)
  APP_C++INCLUDES = -I$(TAEINC)\
                                   -I$(TAEINCXM)\
                                   -I/h/bglcss/CC2.1/SC1.0/include/CC

          PROGRAM = BGSetup

/* PROGRAMMER NOTE:
 * Add $(SRCS_RESFILENAME) and $(OBJS_RESFILENAME) for each resource file
 * in your application.
 */
          GENSRCS = $(PROGRAM).cc $(SRCS_BGSetup)
          GENOBJS = $(PROGRAM).o $(OBJS_BGSetup)

/* PROGRAMMER NOTE:
 * Add your application specific srcs and object files (that are not
 * generated by the code generator) here.
 */
          APPSRCS =
          APPOBJS =

/* Macro (defined in TAEmake.tmpl) to generate Makefile targets.
 */
CPlusPlusApplication($(PROGRAM))
```

```
// *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 ***
// *** File:        item_SetUpBGs.h ***
// *** Generated:   Mar 10 07:51:07 1993 ***
///////////////////////////////////////////////////////////////////////
// PURPOSE:
// This file contains class definitions and instance declarations of
// all items in the TAE Plus panel:
//      SetUpBGs
//
// REGENERATED:
// The following WorkBench operations will cause regeneration of this
// file:
//      The panel's name is changed (not title)
// For panel:
//    SetUpBGs
//
// The following WorkBench operations will also cause regeneration:
//      An item is deleted
//      A new item is added to this panel
//      An item's name is changed (not title)
//      An item's generates events flag is changed
// For the panel items:
//    Close,              Delete,           Edit,             Help,
//    New
//
// CHANGE LOG:
// 10-Mar-93    Initially generated...TAE
///////////////////////////////////////////////////////////////////////

#ifndef I_ITEM_SetUpBGs                        // prevent double include
#define I_ITEM_SetUpBGs       0

#include <taepanel.h>
#include <taeitem.h>
#include <taevm.h>

//
// Class definitions for the items on this panel
//

//*******************************************************************
class SetUpBGs_CloseC : public TaeItem
    {
    void React (WptEvent* event);           // item's event handler
public:
    SetUpBGs_CloseC (TaePanel * a) : (a, "Close") {};
    };

//*******************************************************************
class SetUpBGs_DeleteC : public TaeItem
    {
    void React (WptEvent* event);           // item's event handler
public:
    SetUpBGs_DeleteC (TaePanel * a) : (a, "Delete") {};
    };

//*******************************************************************
class SetUpBGs_EditC : public TaeItem
    {
    void React (WptEvent* event);           // item's event handler
public:
    SetUpBGs_EditC (TaePanel * a) : (a, "Edit") {};
    };

//*******************************************************************
class SetUpBGs_HelpC : public TaeItem
    {
    void React (WptEvent* event);           // item's event handler
public:
    SetUpBGs_HelpC (TaePanel * a) : (a, "Help") {};
    };
//*******************************************************************
class SetUpBGs_NewC : public TaeItem
    {
    void React (WptEvent* event);           // item's event handler
public:
```

```
    SetUpBGs_NewC (TaePanel * a) : (a, "New") {};
    };
//
// Item instances
//
extern SetUpBGs_CloseC *SetUpBGs_CloseI;
extern SetUpBGs_DeleteC *SetUpBGs_DeleteI;
extern SetUpBGs_EditC *SetUpBGs_EditI;
extern SetUpBGs_HelpC *SetUpBGs_HelpI;
extern SetUpBGs_NewC *SetUpBGs_NewI;

#endif
```

```
// *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 ***
// *** File:          pan_SetUpBGs.cc ***
// *** Generated:     Mar 10 07:51:07 1993 ***
//////////////////////////////////////////////////////////////////////
// PURPOSE:
// This file encapsulates the TAE Plus panel:  SetUpBGs
// SetUpBGsP is an instance of the class SetUpBGsC which is a derived
// class of the TaePanel class.  Access to public methods and the
// SetUpBGsP instance from other files is enabled by inserting
// '#include "pan_SetUpBGs.h"'.
//
// NOTES:
// For each parameter that you have defined to be "event-generating"
// in this panel, there is an event handler method defined below.
// Each handler is a method called React in the corresponding item
// class.
// Add application-dependent logic to each event handler.  (As
// generated
// by the WorkBench, each event handler simply logs the occurrence of
// the
// event.)
//
// You may want to flag any changes you make to this file so that if
// you
// regenerate this file, you can more easily cut and paste your
// modifications back in.  For example:
//
//   generated code ...
//   // (+) ADDED yourinitials
//   your code
//   // (-) ADDED
//   more generated code ...
//
//
//
// REGENERATED:
// The following WorkBench operations will cause regeneration of this
// file:
//     The panel's name is changed (not title)
// For panel:
//   SetUpBGs
//
// The following WorkBench operations will also cause regeneration:
//     An item is deleted
//     A new item is added to this panel
//     An item's name is changed (not title)
//     An item's data type is changed
//     An item's generates events flag is changed
//     An item's valids changed (if item is type string and connected)
//     An item's connection information is changed
// For the panel items:
//   Close,          Delete,          Edit,          Help,
//   New
//
// CHANGE LOG:
// 10-Mar-93    Initially generated...TAE
//////////////////////////////////////////////////////////////////////

#include <stream.h>
#include <taepanel.h>
#include <taeitem.h>
#include <taevm.h>
#include "pan_SetUpBGs.h"                       // Panel class declaration
#include "item_SetUpBGs.h"                      // Item class declarations

// One "include" for each connected panel
#include        "pan_CloseAll.h"
#include        "pan_DelBG.h"
#include        "pan_BGData.h"
#include        "pan_NewBG.h"

//
// Panel Instance
//
SetUpBGsC *SetUpBGsP;
```

```
//
// Item Instances
//
SetUpBGs_CloseC *SetUpBGs_CloseI;
SetUpBGs_DeleteC *SetUpBGs_DeleteI;
SetUpBGs_EditC *SetUpBGs_EditI;
SetUpBGs_HelpC *SetUpBGs_HelpI;
SetUpBGs_NewC *SetUpBGs_NewI;


//
// Panel class constructor
//
SetUpBGsC::SetUpBGsC (TaeCollection *collect) : ("SetUpBGs", collect)
    {
    //
    // create an instance of each item in the panel.
    //
    SetUpBGs_CloseI = new SetUpBGs_CloseC (this);
    SetUpBGs_DeleteI = new SetUpBGs_DeleteC (this);
    SetUpBGs_EditI = new SetUpBGs_EditC (this);
    SetUpBGs_HelpI = new SetUpBGs_HelpC (this);
    SetUpBGs_NewI = new SetUpBGs_NewC (this);
    }


//
// Panel class destructor
//
SetUpBGsC::~SetUpBGsC ()
    {
    delete SetUpBGs_CloseI;
    delete SetUpBGs_DeleteI;
    delete SetUpBGs_EditI;
    delete SetUpBGs_HelpI;
    delete SetUpBGs_NewI;
    }


///////////////////////////////////////////////////////////////////////////
// Handle event from parameter:  Close
//
void SetUpBGs_CloseC::React (WptEvent *event)
    {
    // get the target variable from the event
    TaeVar *itemVariable = GetTargetVar( (WptEvent *)event );

    cout << "Panel " << Parent()->Name()
         << ", parm " << itemVariable->Name()
         << ": value = "
         << ( itemVariable->Count()>0 ? itemVariable->String() : "none" )
         << "\n";
    cout.flush();

    // Begin generated code for Connection

    CloseAllP->Show(*Parent()->Handler());

    // End generated code for Connection
    }


///////////////////////////////////////////////////////////////////////////
// Handle event from parameter:  Delete
//
void SetUpBGs_DeleteC::React (WptEvent *event)
    {
    // get the target variable from the event
    TaeVar *itemVariable = GetTargetVar( (WptEvent *)event );

    cout << "Panel " << Parent()->Name()
         << ", parm " << itemVariable->Name()
         << ": value = "
         << ( itemVariable->Count()>0 ? itemVariable->String() : "none" )
         << "\n";
```

```
    cout.flush();

    // Begin generated code for Connection

    DelBGP->Show(*Parent()->Handler());

    // End generated code for Connection
    }

///////////////////////////////////////////////////////////////////////////
// Handle event from parameter:  Edit
//
void SetUpBGs_EditC::React (WptEvent *event)
    {
    // get the target variable from the event
    TaeVar *itemVariable = GetTargetVar( (WptEvent *)event );

    cout << "Panel " << Parent()->Name()
         << ", parm " << itemVariable->Name()
         << ": value = "
         << ( itemVariable->Count()>0 ? itemVariable->String() : "none" )
         << "\n";
    cout.flush();

    // Begin generated code for Connection

    BGDataP->Show(*Parent()->Handler());
    /*  TCL: quit  */

    // End generated code for Connection
    }

///////////////////////////////////////////////////////////////////////////
// Handle event from parameter:  Help
//
void SetUpBGs_HelpC::React (WptEvent *event)
    {
    // get the target variable from the event
    TaeVar *itemVariable = GetTargetVar( (WptEvent *)event );

    cout << "Panel " << Parent()->Name()
         << ", parm " << itemVariable->Name()
         << ": value = "
         << ( itemVariable->Count()>0 ? itemVariable->String() : "none" )
         << "\n";
    cout.flush();
    }


///////////////////////////////////////////////////////////////////////////
// Handle event from parameter:  New
//
void SetUpBGs_NewC::React (WptEvent *event)
    {
    // get the target variable from the event
    TaeVar *itemVariable = GetTargetVar( (WptEvent *)event );

    cout << "Panel " << Parent()->Name()
         << ", parm " << itemVariable->Name()
         << ": value = "
         << ( itemVariable->Count()>0 ? itemVariable->String() : "none" )
         << "\n";
    cout.flush();

    // Begin generated code for Connection

    NewBGP->Show(*Parent()->Handler());
    /*  TCL: quit  */

    // End generated code for Connection
    }
```

```
// *** TAE Plus Code Generator version Tue May 26 14:13:27 EDT 1992 ***
// *** File:        pan_SetUpBGs.h ***
// *** Generated:   Mar 10 07:51:07 1993 ***
///////////////////////////////////////////////////////////////////////////
// PURPOSE:
// Header file for panel:  SetUpBGs
//
//
// For panel:
//   SetUpBGs
//
// CHANGE LOG:
// 10-Mar-93    Initially generated...TAE
///////////////////////////////////////////////////////////////////////////

#ifndef I_PAN_SetUpBGs                      // prevent double include
#define I_PAN_SetUpBGs      0

#include <taepanel.h>
#include <taeitem.h>
#include <taevm.h>

//
// Class definition for the SetUpBGsC class which is a derived class
// of TaePanel class.
//

class SetUpBGsC : public TaePanel
    {
public:
    SetUpBGsC (TaeCollection *collect);
    ~SetUpBGsC ();
    };

//
// The instance of SetUpBGsC class
//
extern SetUpBGsC *SetUpBGsP;

#endif
```

REFERENCES

[CARGILL 92] Cargill, T., "Using Multiple Inheritance in C++", *Supplement to Dr. Dobb's Journal*, pp. 48-51, December 1992.

[COPLIEN 92] Coplien, J. O., *Advanced C++: Programming Styles and Idioms*, Addison-Wesley Publishing Company, 1992.

[FERNANDES 92]Fernandes, K., *User Interface Specifications for Navy Command and Control Systems Version 1.1*, Naval Command, Control, and Ocean Surveillance Center, Research, Development, Test, and Evaluation Division, San Diego, California, June 1992.

[HAMMONDS 91] Hammonds, K., "Software Made Simple: Will Object-Oriented Programming Transform the Computer Industry?", *Business Week*, pp. 92-100, 30 September, 1991.

[HOLUB 92] Holub, A. I., *C+ C++: Programming with Objects in C and C++*, McGraw-Hill, Inc., 1992.

[INRI 91a] Inter-National Research Institute, Government Off-The-Shelf (GOTS)1.1 Style Guide, 1991.

[INRI 91b] Inter-National Research Institute, Government Off-The-Shelf (GOTS)1.1 Software Architecture, 1991.

[INRI 92a] Inter-National Research Institute, TDBM Service, Application Programmer's Interface, Unified Build 2.0, April 1992.

[INRI 92b] Inter-National Research Institute, Unified Build Application/TDA Toolkit: Application Programmer's Interface, Unified Build 2.0, April 1992.

[INRI 92c] Inter-National Research Institute, JOTS II 2.0 User's Guide Draft, July 1992.

[INRI 92d] Inter-National Research Institute, Wizard Toolkit, Application Programmer's Interface, Unified Build 2.0, April 1992.

[LEWIS 92] Lewis, J. A. et al, "On the Relationship Between the Object-Oriented Paradigm and Software Reuse: An Empirical Investigation", *Journal of Object-Oriented Programming*, pp.35-41, July/August 1992.

[MEYER 88] Meyer, B., *Object-Oriented Software Construction*, Prentice Hall, 1988.

[NASA 91a] NASA Goddard Space Flight Center, *TAE Plus Overview V5.1*, April 1991.

[NASA 92a] NASA Goddard Space Flight Center, *Programming Tips and Tricks V5.1*, 1992.

[NASA 92b] NASA Goddard Space Flight Center, *TAE Plus User Interface Developer's Guide V5.2 Beta,* October 1992.

[PERRY 92] Perry, G., *Moving from C to C++*, SAMS Publishing, 1992.

[SCHRADY 90] Schrady, D. A., Wadsworth, D. B., Laverty, R. G., Bednarski, W. S., *Predicting Ship Fuel Consumption*, Technical Report NPSOR-91-03, Naval Postgraduate School, Monterey, California, October 1990.

[SCHRADY 91] Schrady, D. A., Wadsworth, D. B., *User's Guide for the Battle Group Logistics Coordinator Support System (BGLCSS)*, Technical Report NPSOR-91-08, Naval Postgraduate School, Monterey, California, February 1991.

[SETHI 90] Sethi, R., *Programming Languages: Concepts and Constructs*, Addison-Wesley Publishing Company, 1990.

[SHIFFMAN 92]Shiffman, H., "Toward a Less Object-Oriented View of C++", *Supplement to Dr. Dobb's Journal*, pp. 35-38, December 1992.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexanderia, VA 22304-6145	2
2.	Dudley Knox Library Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chairman Computer Science Department Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Dr. C. Thomas Wu Code CS/Wq Associate Professor, Computer Science Department Naval Postgraduate School Monterey, CA 93943-5000	1
5.	Roger Stemp Code CS/Sp Adjunct Instructor, Computer Science Department Naval Postgraduate School Monterey, CA 93943-5000	1
6.	LCDR Donald P. Brutzman Code OR/Br Adjunct Professor, Operations Research Department Naval Postgraduate School Monterey, CA 93943-5000	1
7.	Commander, Space and Navy Warfare Systems Command Don Wayburn, PD-60-L1 Washington, D.C. 20363	1
8.	Bernadette C. Brooks P.O. Box 48 Kensington, MD 20895	6