Interactive Graphics for Geometry Generation—A Program with a Contemporary Design

Walter F. LaBozzetta* and Paul E. Cole†
Lockheed-Georgia Company, Marietta, Georgia

The aerodynamic design process is now strongly influenced by the application of advanced computational tools due mainly to recent advances in computer capabilities. Today's aerodynamic analysis codes require large amounts of geometric data to define the surfaces in sufficient detail for proper analysis. A program has been developed to aid the engineer in generating geometric models for computational aerodynamics codes. The program, Interactive Graphics for Geometry Generation (I3G), is not dramatically different on the "outside" from other interactive graphics programs of this type. It is, however, different on the "inside," i.e., in how it performs its functions. I3G is capable of accepting input from a variety of geometry sources and can support the generation of geometric models for many different computational codes. User-friendly operation encourages program use. Graphics devices ranging from sophisticated systems to low-cost display-only terminals can be easily supported by I3G. The program's modular structure allows new modeling capabilities to be readily incorporated. A system for indexing and retrieving existing geometric models is available to the user. In summary, "flexibility" is the key word behind I3G's basic design and structure. This feature allows modifications and enhancements to be readily made as needs change or arise. Succinctly, it is a program designed to last.

Introduction

A RAPID increase in the capabilities of computational aerodynamics began during the 1970s. Continuing advances in both computing hardware and solution algorithms provide the ability to predict transonic and viscous flows over increasingly complex configurations. This growing capability has improved aerospace vehicle design, but not without cost. Generating the geometric input data, i.e., geometric models, for the computational codes can be a time-consuming operation, especially when models must be generated for several codes during the design exercise. The more complex the configuration, the more difficult it is to generate an acceptable model (one that will give credible results when analyzed). Some types of analysis codes, such as the panel methods, require a particular expertise—many times in an iterative process—to produce a proper model. With the advances in hardware and software continuing in the 1980s, this geometry modeling problem has become more acute.

The problem is compounded by the variety of computers and terminals used within an engineering organization. Aerodynamic design systems have felt the impact of this diversity of computer hardware. The various geometry data manipulating and handling programs developed are generally dependent on the host computer or a particular type of terminal, which makes the task of organizing these support programs into an efficient system far more difficult. Another deficiency of these systems is that often no attempt is made to implement a scheme to index existing geometric models to improve model accessibility by users of the various analysis codes. This lack of an indexing system leads to costly duplication of effort.

In addition to the problem of gnerating geometric models for numerous computational codes is the concern that he designer may develop these models from surface information received from any one of a number of sources. These sources can vary from use of a digitizing tablet combined with engineering drawings to surface definitions from a computer-aided design (CAD) system. Therefore, along with the capability to generate geometric models for the various analysis codes, an effective computational design system must be capable of accommodating multiple input geometry sources.

Early attempts to address the problem resulted in analysis systems centered on a particular analysis code, creating a "code-dependent" system.[1] This type of system has several disadvantages. First, it is sensitive to changes in the analysis code. A change to the input portion of the code could require changes to the input support routine. Second, new analysis codes are not readily supported. A new "system" has to be developed for each new code. Third, users must learn how to use a large number of analysis code support routines. Finally, analysis code idiosyncrasies tend to migrate to the code's support routines.

Approach

The problem of easily generating geometric models for multiple aerodynamic codes, while also supporting several geometry data sources, has been addressed by implementing a different approach to eliminate the disadvantages previously encountered. The new approach emphasizes code-independent data formats and common data operations. Figure 1 illustrates the old and new approaches to interfacing multiple geometry data sources to multiple computational codes. The top of the figure shows the traditional approach requiring numerous unique interfaces, while the bottom shows the code-independent format approach. Use of a common intermediate format significantly reduces the number of interfaces required. This reduces the coding, checkout, and most important, the system maintenance.

Investigation of the overall geometric model generation problem showed that, while the geometry data for each code appeared very different in detail, there was, in fact, a high degree of commonality. From the data storage, manipulation, and plotting viewpoint, there was little difference between the wing section definitions used for input to FLO-22 (a finite difference full potential code) and the surface definitions used for input to HESS (a panel/singularity code). In addition, although the geometric model formats for the diverse codes

Presented as Paper 84-2389 at the AIAA/AHS/ASEE Aircraft Design Systems and Operations Meeting, San Diego, CA, Oct. 31-Nov. 2, 1984; received Dec. 4, 1984; revision submitted Sept. 9, 1985. Copyright © American Institute of Aeronautics and Astronautics, Inc., 1985. All rights reserved.

*Specialist Engineer. Member AIAA.

†Specialist Engineer.

are quite different, they easily can be supported by a common geometry manipulation routine if the data storage definition is developed in a code-independent manner.

Realization of these two key points made it clear that the design of the system should be structured as shown in Fig. 2. Here, a code-independent geometry generation core is interfaced to the various data sources and analysis codes by single-function routines. This single system can support any number of analysis codes—a definite advantage over having a separate system for each code, as was the case in the past. The resulting system is easily maintained and modified for new requirements, i.e., new data sources, new codes, or new geometry generation capabilities.

Discussion

The I3G (Interactive Graphics for Geometry Generation) program utilizes modern interactive computer graphics and database management principles to allow rapid manipulation and reformatting of surface definition data stored in a code-independent format. Since the user views the data graphically during model construction, fewer errors are made, and they are easier to detect.

I3G has six major functional areas:

1) Surface Manipulation
2) Surface Grid-Point Generation
3) Surface Generation
4) Display Functions
5) Database Operations
6) Output Functions

In each of these areas, capability was designed to surpass that afforded by existing manual and computerized methods, with specific emphasis on surface point generation, surface manipulation, and graphic display.

It is appropriate at this time to comment about I3G's surface generation capability. I3G does not try to compete with CAD systems in the area of surface generation. Only a limited capability is provided to create new surface descriptions. Instead, a simple-to-use interface is provided to access data from a CAD system, such as CATIA [an interactive three-dimensional (3-D) preliminary design system] or CADAM (an interactive 2.5-D, i.e., 2-D with an auxiliary view capability, design, and drafting system) data. I3G was developed to provide simple and powerful surface grid-point generation and output data formatting. It is an extension of the more powerful CAD systems, not a replacement.

The program capabilities reflected in I3G's six major functional areas are not significantly different from the capabilities exhibited in other programs performing the same function. However, I3G's approach to providing these capabilities, specifically its overall program philosophy and certain internal design features, is noteworthy. I3G's overall design includes the following main features: 1) a code-independent formulation, 2) a user-friendly program operation, and 3) a method of program documentation that allows an up-to-date version that is easy to maintain. I3G's internal design features that are different from other similar programs are: 1) implementation of internal functions, 2) technique for handling internal data, 3) interactively driven low-level functions, and 4) file and database structures. These design features are the major thrust of this paper and are described in detail in the following sections.

Code-Independent Philosophy

The I3G program performs all internal operations on code-independent formatted data. This affords the advantage of only a single geometry program that easily supports multiple-input data sources and multiple aerodynamic codes. Development of a simple interface is all that is required to support either another source or another analysis code.

As shown in Fig. 2, I3G can readily support input from a variety of geometry data sources such as CATIA, CADAM, or manual (for example, from a graphics tablet). The interface between each of these sources and I3G is provided using the Initial Graphics Exchange Specification (IGES) format.[2] The IGES data description is an emerging industry standard for exchange of graphical data. Adoption of this system significantly increases the probability of being compatible with many possible future data sources.

I3G can support the generation of geometric models for various aerodynamic analysis codes, as shown in Fig. 2. Models for codes as diverse as HESS or QUADPAN (panel codes) and FLO-22 (a finite difference code) can be supported by the I3G system. Future codes would be supported by developing an interface routine to reformat from the I3G format to the particular code format. Once this is done, all of the geometry generation capabilities of I3G would be available to generate models for the new code.

Interaction of Program and User

The operation of I3G is intimately tied to its use of menu selecting and prompting to allow both the experienced and uninitiated user to input information easily. During the recent development of several large interactive programs, it was learned that the proper use of menus and prompts (with defaults shown) is preferable to using command language input. The uninitiated user is led through the operation of the program, while the experienced user can rapidly step through the menus without any appreciable loss in speed. Automatic transferring from one menu to any higher level menu (stepping back through every menu is not necessary) also assists in the running of the program.

Since large interactive programs and an on-line help option go hand-in-hand, I3G is so equipped. At the user's discretion, on-line help information is available for viewing at any point during program execution. Once the help option is selected, an explanation of the various options available to the user at the present location in the program is displayed. Incorporation of the help function directly into the program has the inherent

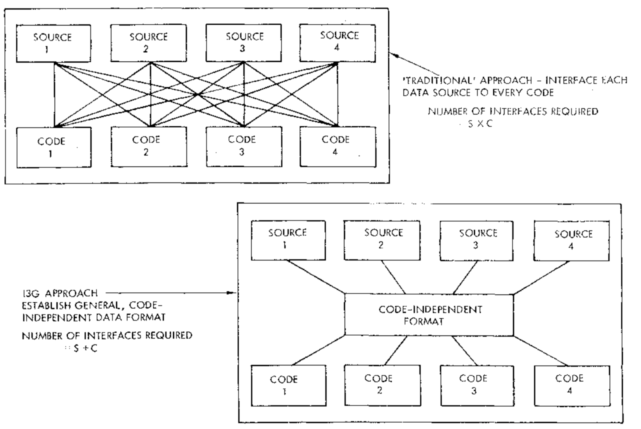

SOURCE 1
SOURCE 2
SOURCE 3
SOURCE 4
CODE 1
CODE 2
CODE 3
CODE 4
'TRADITIONAL' APPROACH – INTERFACE EACH DATA SOURCE TO EVERY CODE
NUMBER OF INTERFACES REQUIRED = S × C
I3G APPROACH
ESTABLISH GENERAL, CODE-INDEPENDENT DATA FORMAT
NUMBER OF INTERFACES REQUIRED = S + C
CODE-INDEPENDENT FORMAT

Fig. 1 Traditional vs code-independent interface concepts.

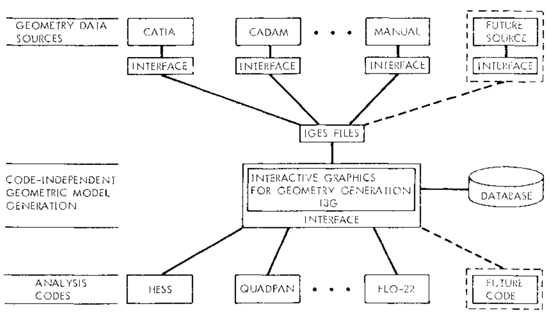
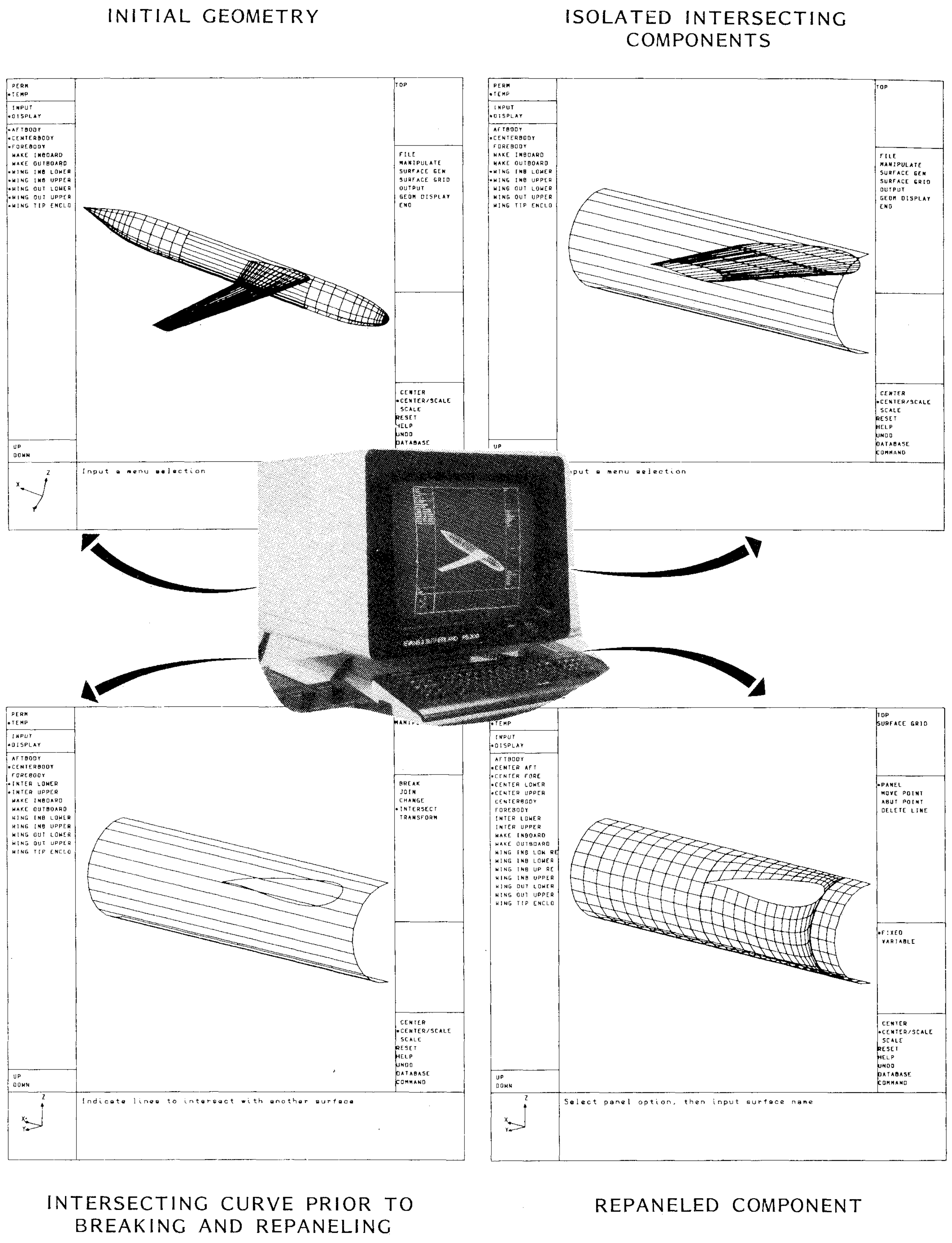


Fig. 2 I3G system overview.



Input a menu selection

Input a menu selection

Indicate lines to intersect with another surface

Select panel option, then input surface name

Fig. 3 Interactively controlled modeling process.

advantage of being "dynamic," in that it will keep abreast with changing program capabilities.

Combining I3G's features of menu selecting, prompting, and the on-line help function creates a user-friendly environment for operation. Engineers who have never used I3G require only a minimal amount of instruction (about 30 min, mainly regarding data storage) before they are comfortable with the program operation, and shortly thereafter they are able to generate geometric models.

Program Documentation

Since I3G is flexible enough to respond to an environment of constant change, the documentation of such a program also had to be dynamic. The most effective method to accomplish this goal was implementation of a procedure to extensively document each module internally (this is consistent with the emphasis of modern programming toward easily discernible coding). The documentation procedure included the insertion of a header block of comments that contained a statement of the module's purpose, a list and definition of passed (both to and from the module) variables, and a description of the theory involved. In addition, the body of each module is well commented to increase the readability of the code. The complete formal program documentation is generated by automatically extracting this information (through the use of a utility program) from the modules themselves, converting it into an easy-to-use format, and adding the following program information: a list of all modules that call a particular module, a list of all common blocks, and a description of the program's internal files. Since this method of documentation relies heavily on obtaining information directly from the program source code (which will be changing as the need arises), it is easy to maintain an up-to-date version of the program documentation.

Novel Program Design

Modern design techniques, i.e., structured analysis and structured design, were applied to ensure a logical system design with simple data interfaces (both internal and external). The resulting code is modular and extensions are easy to incorporate.

Internal Functions

Possibly the most important aspects of the internal design of I3G are the features which maintain input, output, and surface geometry access functions at a "logical" (without regard to detailed implementation for specific hardware devices, such as graphics terminals) level in the program. Program functions are performed through a hierarchical sequence of events, proceeding from the highest level modules to the lowest. This approach allows the majority of code to be written independent of terminal type or the details of the various types of surface definitions. Code maintenance is simplified, and additional capabilities may be added more easily.

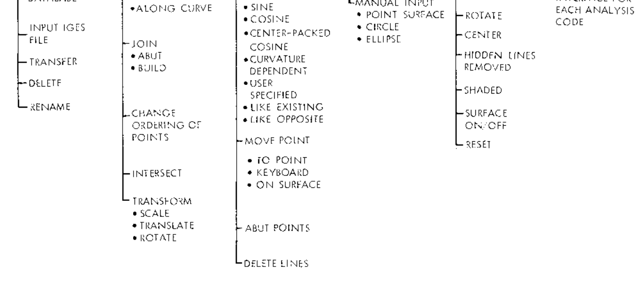


Fig. 4 Options available to the user within I3G.

Within I3G, all data input is requested through a single high-level input module, rather than at various points throughout the program. The module requesting data passes a keyword defining the type of data requested, and a user prompt if desired, to the input module. The input module then manages the request, ultimately calling a low-level terminal-dependent module. The required input data is then returned to the requesting module.

Two significant advantages are derived from this approach to program input. As previously mentioned, terminal independence throughout most of the program is accomplished. The program would only require a few (approximately ten) short modules to be added to support a new terminal type. I3G was designed to be run from terminals as diverse as the Tektronix 4014 and the Evans & Sutherland PS 330 with very little difference either internally or in the user interface. A program designed in this manner allows new advanced terminals to be supported without major program modifications.

Perhaps a more significant advantage is the ability to handle the diversity of requests a user may make at any time. In any truly interactive graphics program, the user should be able to request more than one function at any given time. For example, the user may want to change the display's view or scale, request on-line help, or return to a higher level menu. Attempting to handle this diversity of requests at every input point in the program would be too cumbersome. The single logical input module circumvents this problem and easily services all of the functions provided within I3G.

Interactive output from I3G is handled in a similar manner. Two modules are needed for output—one to output text and one to plot geometry. The calls to these modules contain only "what to do" information, leaving the "how to do" details to the low-level modules.

In a similar manner, when a module needs surface geometry data, it makes a call to a general module that returns this information for any of the various surface descriptions (point definition, polynomial, etc.). The calling module makes a logical request for the data, and then a small number of lower level modules determine the type of surface definition, and, therefore, the detailed operations to be performed. Other than the display of point definition surfaces, essentially all access to surfaces is reduced to determining one surface coordinate in terms of its parametric (u,v) location. Thus, the logical module to return x,y,z (given a surface name and u,v) is used extensively in I3G. High-level modules never need to determine actual surface type, and new surface entities may be added with only a small amount of new code and minimal impact on existing modules.

Internal Data Handling

Information hiding is another technique of structured programming used to simplify the internal data handling within I3G. This program, like any large program, has a number of internal data stores created by some of the modules, modified by others, and used by still others. An example of one such data store is the list of current surfaces and their display status. In most programs these data are contained in a common storage area accessed by many modules. Past history has shown that when many modules access the same common data area, an error generated in one area of the program can propagate to other areas, making error detection and correction difficult. The problem is compounded when several programmers are working on the same program and sharing common data.

The information-hiding technique involves storing and handling a set of data within a limited number of modules (typically about seven). Any high-level module that needs to use or modify that set of data does so through an information-hiding module. The details of the data storage are "hidden" from the requesting module, and the chance of introducing an error into the data is reduced significantly. This technique is provided within I3G by using a module with multiple entry points. Each entry point has its own return, maintaining the programming clarity of a "single-entry, single-exit" module. One entry is provided for each function needed on that data set. All operations on the data are clearly visible within the one module, and it is straightforward to range check and error check all accesses to the data.

Low-Level Functions

Experience has shown that a high percentage of geometry manipulation requirements can be met with a set of general-purpose "tools" run in an interactive manner. Therefore, the basic philosophy behind I3G operation is to supply low-level geometry manipulating functions that are interactively driven by the user. This approach gives the user a large degree of control over the geometry modeling process. The alternate approach of having the user specify a high-level operation would require a "smart" code, capable of making certain decisions while performing its task. This latter approach forces the user to accept whatever "logic" has been programmed into the code, thus limiting functionality and flexibility. The preferred approach, which is implemented in I3G, is illustrated in Fig. 3, as a user interactively controls the steps to perform the modeling (in this case, paneling) of two intersecting components. The general-purpose functions of determining the intersection curve and then breaking and repaneling the components are performed sequentially to generate a completed panel model.

I3G improves the effective use of the aerodynamic analysis codes because it provides a versatile geometry manipulation and perturbation capability. Since the integrity of theoretical results is a strong function of the quality of the geometric model input to the analysis code, the user's ability to manipulate the geometry in the "exact" manner he wishes is crucial to generating an accurate representation of the geometry for an analysis code. I3G affords this capability to the user by containing a large repertoire of geometry modeling options combined with interactive user control. The current list of these options is shown in Fig. 4, but is not limited to these since I3G's design allows for easy additions.

The use of interactive graphics to control operations on geometric surfaces is the main theme of I3G. A surface may be defined as a point, a three-dimensional curve, or a full three-dimensional surface. All manipulations and perturbations can be performed on any "surface" as it is defined here. Program modules perform "general" operations which avoid duplication of code that would otherwise be needed to perform the same operation on either a point, a curve, or a surface. In addition, this approach allows general operation modules to be used with other modules to perform different program functions.

File and Database Structures

Provision for simultaneous access by any number of users is a mandatory feature, considering the multi-user environment in which I3G operates. This was realized early in its development and is reflected in the internal file structure. Multi-user access is accomplished by having the provision for both permanent and temporary "files" (data stores) available for use at any time during program operation. While all of the data manipulation is peformed within a localized temporary file, the permanent data store is available for access by multiple users simultaneously. Operations on permanent file data are restricted to transfer between the files (permanent to/from temporary) and deletions (with appropriate privilege).

I3G's database, which contains completed geometric models, is structured to afford the user ready access to and retrieval of existing models for use in subsequent aerodynamic analysis exercises. An indexing scheme allows inquiries about elements in the database to be made at any point during program operation. Control of access to the database is accomplished by assigned privileges. The average user, for example, may have only a read privilege within certain areas of the database, while the database manager might have read, write, and delete privileges over the entire database.

I3G's internal files and database structures provide the user with the capability to easily locate and/or retrieve existing models. This allows users of the system to reuse these models, thereby avoiding duplication of effort and improving the efficiency of computational aerodynamic analysis.

Conclusions

An interactive geometry manipulation program, I3G, has been developed that allows aerodynamic computational code models to be produced in a timely fashion. The design and structure of the program conforms to modern standards and allows for easy evolution as future needs and capabilities arise. Although I3G is currently centered around aerodynamic analysis codes, its design is flexible enough to allow it to support other computational codes (i.e., not necessarily aerodynamic in nature) requiring a description of geometry. A program change of this type would require only a minimal amount of program modification and a small number of interfaces. Some of the specific advantages I3G's design affords are:

1) Supports multiple geometry data sources and multiple analysis codes and is easily modified to support new sources and new codes.

2) Easy to use due to menu selecting, prompting, and on-line help operation.

3) Expandable to drive new graphic display devices with minor program modifications.

4) Flexible geometry manipulation capability due to interactive user control of low-level functions.

5) Easily expanded to supply additional capabilities.

6) Easy retrieval of existing geometric models to reduce the geometry generation effort.

Acknowledgment

This work was supported by the Lockheed-Georgia Company Independent Research and Development program.

Reference

[1]Halsey, N. D. and Hess, J. L., "A Geometry Package for Generation of Input Data for a Three-Dimensional Potential-Flow Program," NASA CR-2962, June 1978.

[2]Smith, B. M., Brauner, K. M., Kennicott, P. R., Liewald, M., and Wellington, J., "Initial Graphics Exchange Specification (IGES), Version 2.0," U. S. Department of Commerce, Rept. NBSIR 82-2631 (AF), Feb. 1983.