Complementary Integration of PHIGS and a Geometric Modeling Utility

Paul A. Elletson and Mukul Saxena
GE - Corporate Research and Development
Solid Mechanics Laboratory
Schenectady, NY 12301

Abstract

Graphical display and interaction with geometry constitute the base functionality of the user interface for many CAD/CAM/CAE applications. This paper presents a representation which provides this functionality by integrating a topological model described within a geometric modeling utility with PHIGS. This representation promotes easy and efficient implementation of graphical display and interaction with geometry by leveraging the underlying functionality of the geometric modeling utility. An implementation of this representation is described in the context of a non-manifold topology based geometric modeling utility.

Keywords: PHIGS, Geometry Modeling, Non-Manifold Topology

1. Introduction

Most commercial CAD systems are closed architecture systems which only provide the user with high level operators and primitives for creating and manipulating a geometric model. However, because many CAD/CAM/CAE applications require the ability to define and manipulate low-level geometric and topological entities, an alternative to a closed architecture system is warranted. One can visualize a geometric modeling environment with an open architecture that provides the low-level functionality required by these geometry-based application programs. Such an environment consists of (a) data structures that serve as the repository for the model geometry and (b) operators to manipulate them. An application programmer, operating with such a utility, must now interact with low-level geometric (e.g., points, curves, and surfaces) and topological (e.g., vertices, edges, and faces) entities, which serve as the basic building blocks of the system.

Development of a graphical system in support of such a modeling utility presents some interesting challenges as it requires a tight integration of the geometric model with the graphical structures of the system to ensure a consistency between the topological model and its graphical representation. This paper focuses on the implementation issues of such a topology-based graphical interface and describes how the implicit hierarchy of the topological model can be exploited in efficiently developing the base functionality of the interface.

The paper is organized as follows. Section 2 presents an overview of the enabling technologies, namely the Topology And Geometry Utility System (TAGUS) that provides the modeling environment for the application programs, and the Programmer's Hierarchical Interactive Graphics System (PHIGS). Section 3 describes the association of model geometry with the graphical structures represented in PHIGS. Some implementation details are presented in Section 4 followed by specific examples of complementary interaction between TAGUS entities and PHIGS structures in Section 5. Finally, the conclusions are summarized in Section 6.

2. Enabling Technologies

In order to explain the integration of the graphical structures and the topological entities defined in the geometry modeling utility, TAGUS, it is essential to understand the notions of geometry and topology and the technical aspects of the two software systems used to implement the concepts described in this paper. The first system, TAGUS (Topology And Geometric Utility System), is a non-manifold topology based modeling utility which serves as a bridge between the closed geometric modeling systems and geometry-dependent application programs that require access to, and manipulation of, the model description. The second system if ANSI standard PHIGS (Programmer's Hierarchical Interactive Graphics System). Template Graphics Software's FIGARO+ 2.01 provided the PHIGS functionality to implement the concepts presented in this paper.

2.1 Geometry and Topology Data

A "geometric" model is defined through the mathematical definition of geometric entities such as points, curves, and surfaces and the relationships between these entities. The framework used to represent this relational information is referred to as topology. Figure 1 shows the hierarchical nature of topology and the relationships between the various topological and geometric entities. A set of working definitions for the various topological entities is given below. The definitions which follow are for general non-manifold objects and are fully described by Weiler [1]. The term *non-manifold* is used to represent topological situations which are not constrained to be two-manifolds. Since all the commercial CAD systems are based on two-manifold topology, they cannot be used to represent non-manifold situations such as domains where more than two faces meet at an edge. Additionally, a non-manifold topology based environment allows unambiguous representation of wire-frames, surface-models, and volume-models simultaneously.

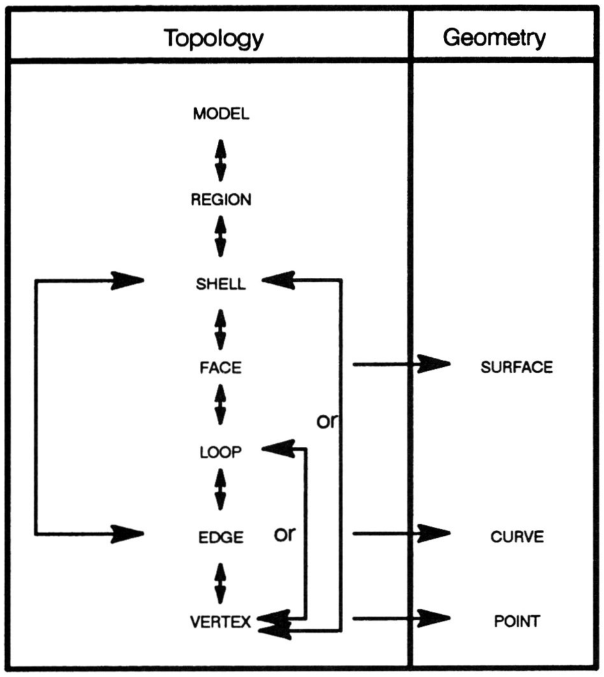

Figure 1. Non-manifold topology representation

1. A *vertex* is a topological representation of a point in three-dimensional space. A geometric point is associated with a vertex.
2. An *edge* is the topological equivalent of a curve and consists of a starting and ending vertex, which defines the direction of edge traversal.
3. A *loop* is defined by an ordered, closed, connected, non-self-intersecting collection of edges. Loops are used to define the boundaries of a face.
4. A *face* is defined by one or more loops, one of which defines the outer boundary and additional loops to define the interior boundaries (e.g., holes). A face has an associated geometric surface.
5. A *shell* is a set of faces which is an oriented boundary for a region.
6. A *region* consists of one or more shells and represents a volume of space.
7. A *model* is a collection of regions.

The topological entities are hierarchical in nature, with higher order entities being comprised of lower order entities.

2.2 Topology And Geometry Utility System

TAGUS is a combination of data structures and operators designed to meet the requirements of geometry-based applications by providing the capability to directly access and manipulate geometry, regardless of the source of model geometry [2]. Since most geometric modeling systems have closed architectures, TAGUS provides a logical bridge between such modeling environments and applications which rely on those systems for their geometry definition. Figure 2 shows a conceptual view of the TAGUS system, along with the way in which it can be utilized in a CAE environment. There are three basic components to the system: (1) geometry source, (2) TAGUS, and (3) the target applications which are built with the TAGUS operators.

TAGUS supports three data structure types: the non-manifold topology (NMT) data, the geometric entity data, and the attribute data. The topology and geometry data were described earlier in this section. The NMT model provides a framework for describing the geometry data, whereas the attribute data provides for additional application-specific information. For example, in a graphical rendering system, color might be an attribute of a model edge. Attributes may be assigned to any of the available topological entities.

2.3 Programmer's Hierarchical Interactive Graphics System (PHIGS)

PHIGS, an ANSI standard, is a set of operators which provides the user with the ability to closely control the creation, modification, archiving, deleting, and viewing of graphical objects. Furthermore, PHIGS provides the user with the ability to directly interact with these graphical objects. For example, PHIGS can return the identifier of a picked graphical object. Brown, et. al. [3] delivers an overview of the functionality provided by PHIGS.

PHIGS' principal data structure is known as a *structure*. Henceforth, the term *structure* will always refer to a PHIGS structure. Each structure is identified by a unique integer value. Every structure consists of a sequential display list of structure elements. There are four types of structure elements in PHIGS

- Graphical Primitives
- Graphical Attributes
- Structure Invocation Elements
- Application Data

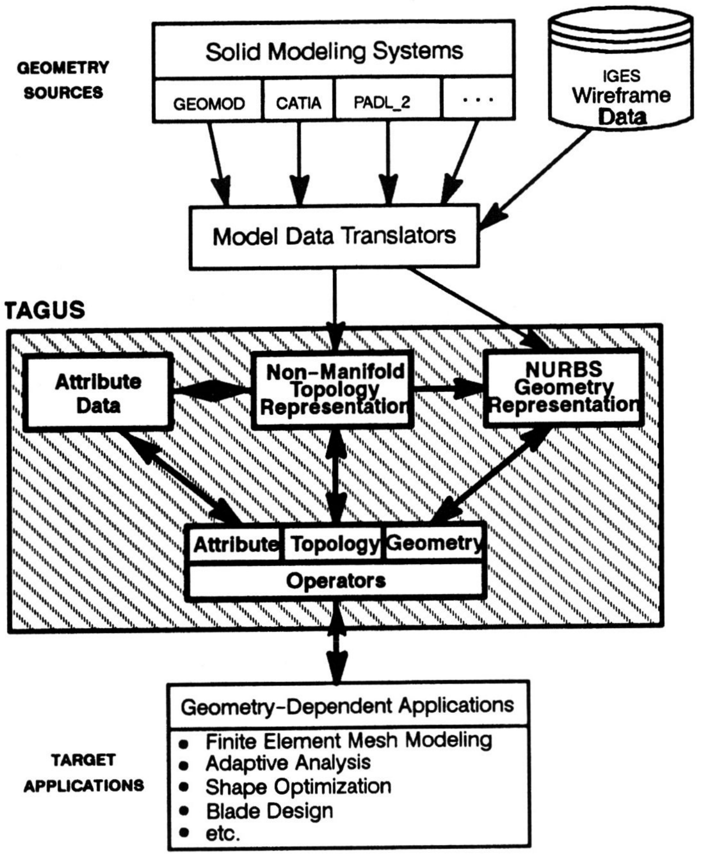

Figure 2. TAGUS system architecture.

Graphical primitives (e.g. polylines and polymarkers) generate graphical output. Graphical attributes (e.g. color and pickability specifications) affect the appearance and behavior of graphical primitives. Application data storage reserves storage in a structure for application-specific information. Structure invocation elements hierarchically associate one structure with another. The latter elements are used to construct hierarchies of structures. Lower-level structures may inherit attributes from higher-level structures.

PHIGS supports two kind of structures: retained structures and non-retained structures. Non-retained structures are not preserved in memory after they are created. Hence, a non-retained structure can generate graphical output only at the time it is created. Retained structures, on the other hand, are retained in memory after they are created. each retained structure may generate graphical output many times. Furthermore, its contents may be selectively changed through the PHIGS' structure editing facility. In the context of this paper, the term *structure* always refers to a retained structure.

3. Association of PHIGS Structures with Topological Entities

In the representation presented by this paper, all graphics-based functionality is provided by PHIGS. Hence, given the model, a corresponding set of PHIGS structures must be created. Important features of this set of structures are the number of structures created, the hierarchical associations among these structures, and the content of each structure. These three features constitute a framework which characterizes the design presented in this paper:

(1) One structure is created for every topological entity. Each structure represents a particular topological entity. Hence, vertex structures represent vertices, edge structures represent edges, and so on. No other structures are created.

(2) The structures are not associated with one another, and, therefore, a structure hierarchy is not maintained.

(3) The content of every structure can be classified into either two or three of the following categories (see Figure 3):

:

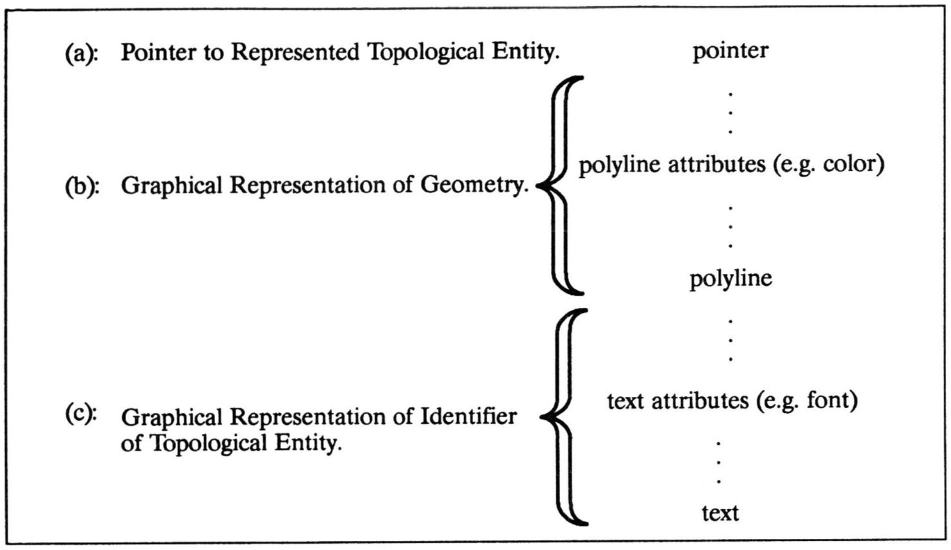
(a): Pointer to Represented Topological Entity.
pointer
(b): Graphical Representation of Geometry.
polyline attributes (e.g. color)
polyline
(c): Graphical Representation of Identifier of Topological Entity.
text attributes (e.g. font)
text

Figure 3. The three components of the structures of this representation.

(a) A pointer to the represented topological entity.

(b) A graphical primitive (e.g. a polyline) which represents the geometry associated with this topological entity and attributes which control the appearance of this graphical primitive.

(c) A graphical primitive (e.g., text) which represents the identifier of this topological entity and attributes which control the appearance of this graphical primitive. This graphical primitive is used to label the represented topological entity.

In this representation, all structures contain (a) and (c). However, only structures which represent classes of topology for which there is associated geometry contain (b). For example, an edge structure will contain all three sections, because every edge is associated with a curve. Conversely, a region structure will contain only (a) and (c), because the geometry of a region is defined by its underlying points, curves, and surfaces which are only associated with vertices, edges, and faces, respectively.

The lack of hierarchical associations among structures greatly eases the effort required to implement this scheme for an environment in which the topological representation may change. Potentially, a representation could have been chosen for which the topological hierarchy was directly mapped to an equivalent hierarchy of PHIGS structures. However, with such a one-to-one mapping, much of the functionality provided by the geometric modeling utility system would have been duplicated.

Hence, if the structures had been associated hierarchically, changes made to the topological representation could require redirection of the pointers (see Figure 3) contained within the structures associated with the affected topology. In the present representation, any change made to the topological representation requires only creation or deletion of associated structures. When new topological entities are created/deleted, counterpart structures are created/deleted. However, when a topological entity is modified, no structures are affected.

Let us consider the following example of *glue–face* functionality supported in TAGUS as one of the core operators for merging topological entities. As the name suggests, *glue–face* allows two topological faces to be merged together. Note that this operator is significantly different than the "boolean merge" functionality [4] available within the commercial modeling systems as explained in Figure 4. The boolean merge operation combines two regions and removes the interface between the two whereas the *glue–face* operator retains the face as an interface between the two regions. The glue-face operator is thus capable of modeling non-manifold situations, like the one illustrated in Figure 4 where all the edges of the interface are non-manifold edges. As a side-effect of this operator, the topological framework defining the model is considerably modified. To list the obvious changes: one of the faces and all its associated low-level topological entities (edges and vertices) are deleted. For such simple deletions it's easy to preserve the consistency between the topological model and the graphical structures. This is accomplished by deleting the associated structures in PHIGS. However, another manifestation of the *glue–face* operator is that the *shells* get modified as well. As explained earlier, a *shell* is simply a collection of faces that define region boundaries. In our specific example when regions are merged their boundaries and thus the associated shells also modified. If such an association between shells and their constituent faces was maintained in the hierarchical representation of PHIGS structures, one must ensure that the corresponding structures are appropriately modified to reflect such changes. However, our integration of the topological entities and the structures in PHIGS circumvents this problem, as all such associations are directly derived from the underlying topological model. Hence, the bookkeeping, and thus the programming effort, required to maintain such a consistency is reduced considerably.

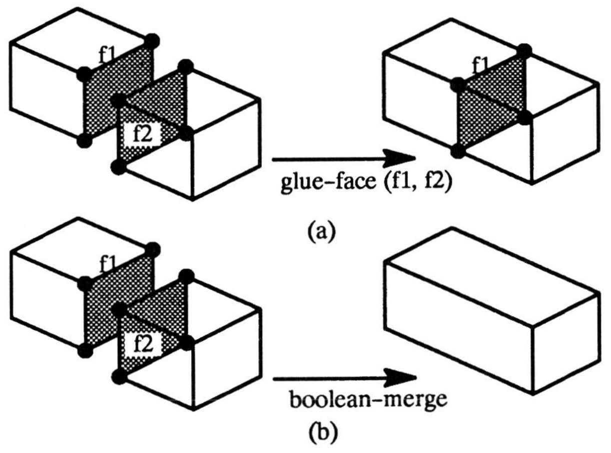

Figure 4. Modeling non-manifolds using glue-face operator in TAGUS (a). Boolean merge operation in a conventional modeling system produces a manifold object (b).

4. Implementation in TAGUS Environment

The authors have implemented the representation described in the previous section for TAGUS models. The resulting implementation, known as the f_toolkit (FIGARO Toolkit), comprises a set of operators written in C which maintain this underlying representation while providing an applications programmer with tools for interactive viewing, constructing, manipulating, and archiving of TAGUS models. Reference [5] provides a detailed description of the f_toolkit.

This implementation will be described in the context of the framework established in Section 3. Description of this implementation under characteristics (1) and (2) requires little additional explanation. In this implementation, every TAGUS topological entity is associated with a PHIGS structure. Every structure is completely independent of every other structure. No other structures exist.

On the other hand, description of this implementation under characteristic (3) merits longer discussion. Each of the three components of characteristic (3) (a, b, and c) will be discussed in turn.

The pointer contained within a structure forms one-half of a bidirectional link between that structure and its corresponding TAGUS entity. This pointer has been implemented by storing in the structure an application data element which contains the type and id of this topological entity. Pointers in the opposite direction are stored within the TAGUS representation using the TAGUS attribute facility. Every TAGUS topological entity points to its corresponding structure by storing the integer identifier of that structure. Figure 5 illustrates this bidirectional association.

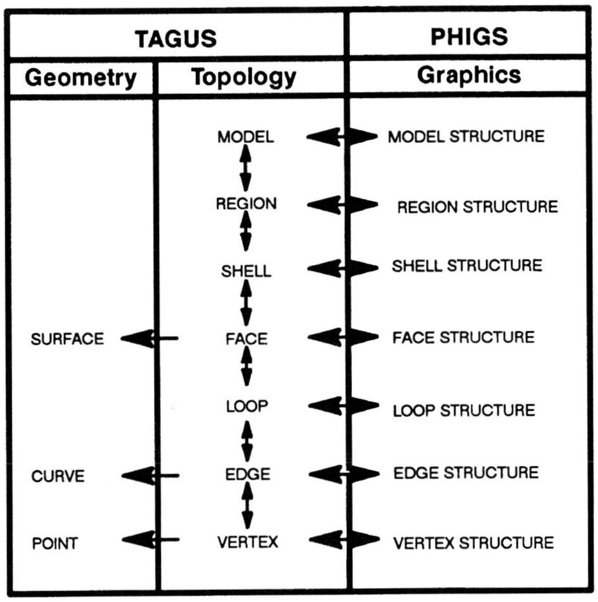

Figure 5. Association between TAGUS Entities and PHIGS Structures

Only vertex and edge structures contain component (b). In TAGUS, vertex, edge, and face entities have counterpart geometry (see Figure 5). However, because surface rendering can be computationally expensive, the face structures of this implementation contain no graphical primitives to represent their counterpart trimmed surfaces. Hence, while vertex and edge structures contain all three characteristics discussed in the previous section, loop, face, shell, region, and model structures contain only components (a) and (c). A polymarker represents the geometry of a vertex structure, while a polyline represents the geometry of an edge structure.

5. Examples

The association between TAGUS and PHIGS enables the functionality provided by both systems to be used in a complementary fashion. This section provides two examples of this by describing the f_toolkit's picking and toggling functionality.

5.1 Picking Entities

The interactive functionality provided by PHIGS can be used to provide input to TAGUS operators. Picking is one of the forms of interactive input provided by PHIGS. In this implementation, PHIGS has been set up to recognize picks of only the text elements of the structures. The text element of a structure labels its corresponding topological entity.

This label is simply the integer identifier of that entity. If a text element is picked, PHIGS returns to the application the identifier of the associated structure. Given this identifier, PHIGS routines can be used to extract the pointer from the picked structure. As explained earlier, this pointer identifies the TAGUS entity represented by the picked structure. Nothing beyond this identifier is required to perform many useful operations using TAGUS operators. For example, a picked entity might be deleted by using the TAGUS delete operator. This, in turn, requires that the structure corresponding to this entity be deleted in order to preserve the consistency between TAGUS and the graphical representation of the model. Hence, complementing the interactive functionality provided by PHIGS with the operators provided by TAGUS produces a very powerful interactive interface to the TAGUS functionality.

5.2 Selective Display

The display functionality provided by PHIGS can be used in conjunction with the TAGUS topological traversal operators to selectively display any topological entity in the model. That is, entire loops, faces, shells, regions, and the model can be drawn in a distinguishing manner. For example, Figure 6 depicts a cube for which one face has been drawn in bold, distinguishing it from the other faces of the cube. This is accomplished in the following manner: Given the identifier of the face to be marked, TAGUS traversal operators are called to obtain the loop underlying this face, and, in turn, all the edges underlying this loop. Then, the TAGUS attribute operators are called to determine the identifiers of the PHIGS structures corresponding to these edges. Finally, the polyline attributes of these edge structures are modified to draw the edges with a thick width. Hence, complementing the display and structure editing functionality provided by PHIGS with the traversal and attribute operators provided by TAGUS produces a means of graphically manifesting the topological representation of the model.

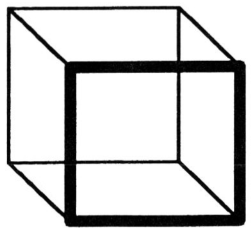

Figure 6. Cube with one marked face.

6. CONCLUSIONS

This paper has presented an approach to integrating graphical structures with a topological representation described in the geometric modeling utility, TAGUS. The distinguishing characteristic of this approach is the reduction of programming effort required to implement the base functionality of the user interface, i.e., graphical display and interaction with topological entities. Thus, a graphical toolkit can be efficiently developed that exploits the functionality available within the modeling utility. In support of this view, the implementation of a graphical system for TAGUS models is described in the paper. Such a system must maintain a graphical representation consistent with the underlying topological model. An alternative scheme of mapping the topological hierarchy to a hierarchy of graphical structures amounts to duplicating much of the information that already exists in the topological framework of the model.

In conclusion, the complementary integration of the graphical structures and the topological entities results in efficient algorithms for implementing the base functionality of the user interface.

Acknowledgements

The authors would like to thank Peter M. Finnigan of the Corporate Research and Development of the General Electric Company for his valuable comments and suggestions during the development and implementation of the concepts described in this paper. The development of the graphics toolkit (*f–toolkit*) described in this paper was supported by GE Aircraft Engines, Evendale.

References

1. Requicha, A.A.G. and Voelcker, H.B., "Boolean operations in solid modeling: Boundary evaluation and merging algorithms", *Proceedings of IEEE*, vol. 3, pp. 30–44, 1985.

2. *F_Toolkit: FIGARO Toolkit: User's Guide & Reference Manual, Version 1.0,* General Electric Corporate Research & Development Mechanical Design Methods Program, October, 1990.

3. Weiler, K.J., *Topological Structures for Geometric Modeling,* PhD Thesis, Rensselaer Polytechnic Institute, August 1986.

4. *TAGUS: Topology and Geometry Utility System: User's Guide & Reference Manual, Version 3.0,* General Electric Corporate Research & Development CAE Automation Project, June, 1990.

5. Brown, Maxine D. and Michael Heck, *Understanding PHIGS: The Hierarchical Computer Graphics Standard*, Template Graphics Software Division of Megatek Corp., San Diego, CA 1985.