To appear in the ACM Symposium on Solid Modeling and Applications, Atlanta, May 14-16, 1997.

Structured Topological Complexes: A Feature-based API for Non-Manifold Topologies

Jarek Rossignac

GVU / College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
www.cc.gatech.edu/gvu/people/jarek.rossignac
jarek@cc.gatech.edu

Abstract

Much of recent research in representation schemes for solid modeling was focused on the extension of boundary representations to support non-manifold topologies. We introduce here a very simple, yet general formalism, which subsumes and simplifies many previous attempts at defining high-level operators for creating and interrogating such models. Our Structured Topological Complex (STC) extends our previous work on Selective Geometric Complexes (SGC) and on Constructive Non-Regularized Geometry (CNRG) and provides the foundations for a new generation of representations–both constructive and evaluated–and of APIs that are independent of the particular geometric domain and even of the particular approximation scheme for geometric primitives.

1. INTRODUCTION

In spite of recent progress in non-manifold topological modeling technologies (see the author's survey [Rossignac94] for a detailed review of the numerous non-manifold modeling schemes) most CAD systems and modeling libraries support a restricted topological domain [Requicha&Rossignac92] and offer APIs biased to specific applications. We propose in this paper a general shape specification scheme for creating general non-manifold structures simultaneously with their partition into components or features that are important for a particular application domain. Because specialists of different application areas may need to collaborate on the same model and may need to manipulate different combinations of the model's components, our scheme supports the simultaneous creation and interrogation of multiple views of the same design. Each view provides a decomposition of the entire model into mutually disjoint features. Because the partition of space into features for one view is compatible with the partition for another view, features from two different views may be composed in various ways to define new features for a third view.

1.1 Contributions

Our Structured Topological Complexes (STC) support Boolean and topological selections of topologically identifiable entities in a space subdivision that is induced by a set of primitives. The primitives may be parameterized and composed of several non-manifold components (such as composite structures or finite element meshes of shapes with internal cracks).

This paper reports several original contributions:

- Designing and maintaining simultaneous and compatible views which partition space into mutually disjoint features provides a unifying and clarifying framework for dealing systematically with collections of possibly overlapping features and with non-manifold sets.
- The support of parameterized primitives which may have a non-uniform structure (i.e. may be decomposed into several non-manifold subsets, or primitive features) permits to specify the selectable entities in a primitive set independently of the actual representation or approximation of their geometry. For example, a cylinder primitive may expose its 3 faces and 2 edges, even though the cylindrical face and the edges may be modeled with several parametric entities in one CAD system and may be approximated with a large number of triangular facets in another CAD system.
- Our simple, yet powerful, scheme for selecting and combining the features from the various views into new features extends the concepts of Constructive Solid Geometry (CSG) in several ways:
    - It combines Boolean operators with topological operators and with adjacency and ordering relations;
    - It supports a CSG formulation of the decomposition of the entire space into features, rather than a formulation of a single pointset;
    - It permits to simultaneously construct and maintain several such parameterized CSG models of the space decomposition that are derived from the same structured primitives.
- Our constructive design process ensures that our extended CSG formulation of two different features of a single view are always disjoint.
- Because our definitions of primitives, features, views, and selection filters are completely independent of the geo-

metric domain supported by a particular geometric core subsystem, the proposed scheme provides the theoretical basis for a universal API for solid and non-manifold modeling. Such an API will isolate the applications from a modeler's dependency on a particular geometric domain and will be well suited for portable applications.

1.2 Prior art

The topological domain of the individual features that may be defined with our Structured Topological Complex (STC) is broadly equivalent to the domain of pointsets with non-manifold internal structures that may be defined using the Constructive Non-Regularized Geometry (CNRG) design scheme [Rossignac&Requicha91]. As such, it significantly expands the topological domain of popular modeling kernels. STC extends the CNRG topological selection operators and the topological filters proposed by Weiler and McLachlan [Weiler&McLachlan91] with adjacency and ordering filters, which are important for differentiating between the connected components of a space subdivision, and thus for capturing persistent references to geometrically selected entities in parameterized constructive models.

The idea of multiple concurrent views upon which STCs are based combines the traditional notion of layers (such as the electric, mechanical, or hydraulic component of a CAD model) found in many CAD systems with the need for supporting sets composed of possibly overlapping non-manifold features [Rossignac90].

Implementations of STC's may be based on a variety of non-manifold representation schemes [Bardis&Patrikalakis91], including the Selective Geometric Complexes (SGC) models [Rossignac&O'Connor89], which support decompositions of space into connected open cells (such as vertices, edges, faces, and regions. The GNOMES system [Sriram&...95] developed at MIT illustrates an object-oriented geometric modeling core based on SGCs and suitable for supporting an STC interface. STC's favor the decomposition of the entire space [Cavalcanti&...] (i.e., the pointset of the model and its complement).

An application that exercises a geometric modeling scheme should be independent of the particular implementation of the geometric modeling core. Data encapsulation, offered by object-oriented programming interfaces, insulates the application from the internal data-structures and access methods and may easily offer a domain independent high-level API for solid modeling. Unfortunately, it does not provide by itself, a universal panacea for more detailed non-manifold modeling, as discussed in a parallel effort [Bowyer&...95]. By focusing on topologically identifiable entities, the STC technology introduced here dissociates the API from the geometric domain. For example, an application which manipulates two cylindrical surface primitives and interrogates the topology of their intersection may be consistently executed on different modeling cores that may support implicit surfaces, parametric patches, or faceted approximations (although of course, the topology of the intersection may be affected by the geometry approximation scheme in use). The paper introduces a constructive model directly accessible by the application, and an evaluated boundary representation model, typically hidden from the application. Although a particular geometric modeling core may choose to use alternate schemes for storing the Boundary representation (voxels, octtrees) or choose not to store the Boundary representation at all, we discuss here the relation between the constructive scheme and the properties of a non-manifold Boundary representation.

1.3 Outline of the paper

Section 2 explains the details of constructing and managing multiple views. Section 3 provides the definition of the topological entities and the details of the selection operators. Section 4 discusses representation issues.

2. VIEWS

2.1 Views and features

An STC model typically offers several concurrent views. Each view is a decomposition of the entire space, E3, into features. Each feature is a possibly non-manifold pointset. Two features of the same view are mutually disjoint. The union of the features of a given view covers the entire space.

A view is identified by its view-number or possibly by a view-name, such as "pump 33". Similarly, a feature may be identified by a view (number or name) and a feature identifier (number or name). For example, View1[feature3], refers to Feature3 in View1. In this paper we will often use colors to identify the features in a view.

2.2 Primitive view and primitive features

An application domain may have its own set of primitive views. Mechanical design may have simple primitive solid shapes, such as cylinders, blocks, and spheres. AEC systems may offer other primitive views, such as pipes.

Furthermore, each application domain may decompose its primitive views into different sets of primitive features.

For example, a solid modeling primitive could yield a black and white view with two features: the solid and its complement. Some applications may prefer three-colored primitives, where the interior, the boundary, and the complement of the solid are distinguished. This may be useful for identifying contact regions between primitives.

Further decomposition of a primitive view may identify different subsets of the boundary of the primitive, such as its bounding faces, edges, and vertices, regardless of any particular representation or geometric approximation used for the faces and edges. For example, a cylinder may be composed of three faces and two edges, each distinguished by a different color, although the cylindrical face may in fact be modeled as four parametric patches or 24 triangular facets.

Primitive views are not restricted to match the natural elements that compose the boundary and the interior of manifold or regularized sets. Each feature may be a mixed-dimensional, non-manifold set by itself. For example, a single feature may be the union of a cylindrical face with a disk cap and their common circular edge. A second feature of the same view may group the interior with the other cap and remaining edge.

The representation of primitive views is domain dependent and its implementation is typically associated with the geometric core, capable of displaying and intersecting the features it supports. An application may choose to ignore the topology of a feature (for example, not differentiating between the dimensionality of a region of contact between two solids) and to interrogate it as a whole during analysis.

In summary, the primitive views and their decomposition into primitive features are defined for a given application domain. They would be specified in the API for that domain and would be supported by all geometric modeling core sub-systems serving the domain. A primitive view is supported by a modeling system if the view may be instantiated by invoking it through a parameterized expression in the API and if its features may be selected, combined, and interrogated as defined in this paper. Note that it is not relevant how a particular modeling system represents the individual features of a view.

2.3 Multiple views

The features of one decomposition may partly overlap with the features of another decomposition. Thus, a point may be red in one view (i.e. belong to a red feature in that view) and green in another. The features of a single view correspond to the notion of layers supported by some popular CAD systems (an object belongs to a single layer), while the views extend the notion of sets, hence supporting overlapping features. For example, one view may distinguish the structural elements, pipes, wires, and body of an airplane. Another view may split the plane into section organized front to back. A third may provide the leaves of a design hierarchy. A fourth may present a view of the bill-of material reflecting the domains of a global purchasing strategy.

Even more important is the fact that STC features offer a powerful design abstraction by providing a mechanism through which design intent, constraints, functions, relationships and annotations may be associated simultaneously with subsets of several primitives. For example, in an architectural model, design may be performed in terms of spaces (rooms, corridors) while analysis and norm conformity tests are conducted in terms of physical components (walls, support structures, pipes, wires). A room feature in a space decomposition view may combine the interior faces of four walls. A wall feature in a material view may combine the opposite sides of a wall with its interior.

New views may be created by combining primitive views as explained below.

2.4 Regions and composite views

A juxtaposition of several primitive views induces a decomposition of space into regions.

Each region is the intersection of a number of features, one in each view. A region can therefore be identified by the sorted list of names (or colors) of these features. For instance, if we have 6 views, a feature will be identified by a vector of six colors (one per view).

Note that in any given instantiation of a primitive views (i.e., the choice of their position and dimension parameters) most color-vectors correspond to empty features. In fact, the number of regions defined by simple primitive views is proportional to the cube of the number of views. (By simple view, we mean that the view has a finite number of primitive features.)

Composite views are constructed by the application from its domain dependent primitive views. The features of a new composite view are typically defined as combinations of primitive features or of features in previously defined views. The features of a composite view may for example be set theoretic Boolean combination of primitive features. Such a scenario would provide a simple scheme for simultaneously representing multiple CSG models. More elaborate feature expressions, involving topological selections, will be introduced in the following sections.

2.5 Incremental feature definition

It would be difficult in general to ensure that the definitions of the various features in a composite view are compatible (i.e., that the features are disjoint and that their union covers E3). Therefore, features of composite STC views are constructed via incremental color assignments, following a printing metaphor: A selection filter (mask) exposes a pointset to which a color is applied for a given view. The pointset is the union of regions defined by the primitive views.

For example, the red feature of View1 may be constructed in three steps by painting in red over a set A, then painting in red again over a set B possibly overlapping with A, and finally by painting in green over a set C, which may contain points previously marked in red. A possible syntax for an assignment may be: *"view3.Paint(set,color);"* where "set" represents an expression that identifies a pointset. An example of such an expression would be *"Intersect(view2[red],view3[green])"*, which returns the identification of the geometry that corresponds to the intersection of the red feature of view2 with the green feature of view3.

Figure 1 represents, in 2D, a primitive view: View1, which has 3 features: an open dashed rectangular region (the interior of the rectangle) colored in green, its boundary (thick line edges and large disk vertices) colored in red, and its complement colored in white. (Note that we are not showing the colors in the figure, but are using color names to refer to the features.)

Figure 2 represents another instance of this primitive, where the rectangle has a different position and dimension.

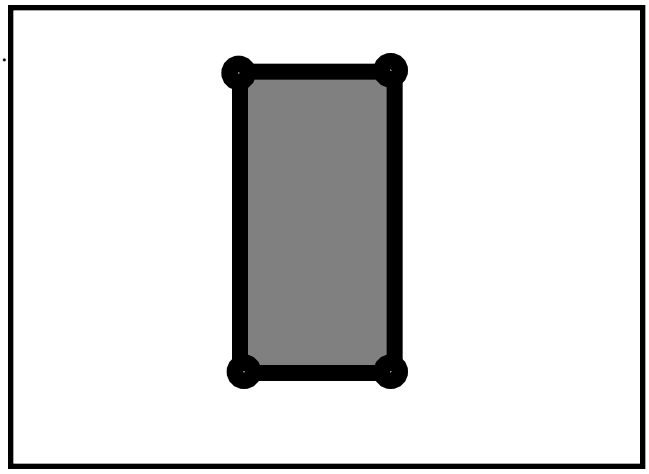

FIG 1: Primitive view 1 with 3 features

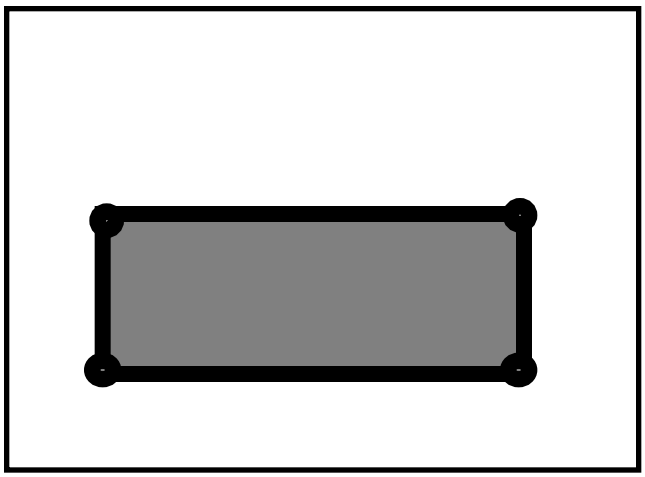

FIG 2: Primitive view 2 with 3 features

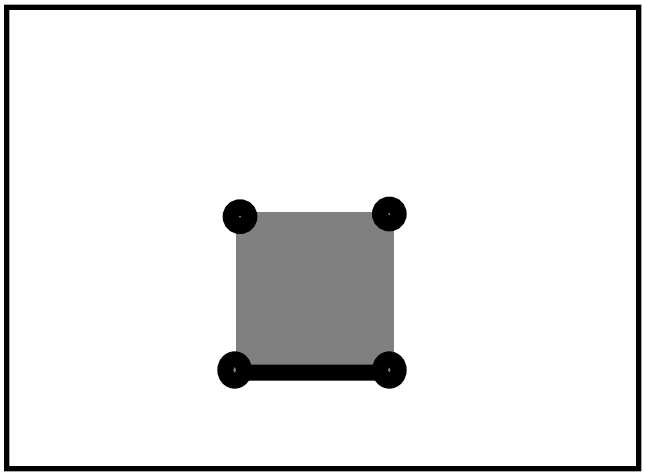

FIG 3: Composite view 3 with 3 features

Figure 3 shows a composite view with three features: the green open set that is the intersection of the intreriors of both triangles (here dashed), the red set composed of a closed edge and two isolated points which is the intersection of the boundaries of both rectangles, (thick lines and points) and a white feature. Note that features are not dimensionally homogeneous

2.6 Constructive model

The views (with their colored features) represent a decomposition of space useful for a particular application. The recipe (sequence of primitive and composite view creation and of color assignment operations) may be stored as a parameterized constructive model containing two entities: a geometry specification graph and a feature definition sequence. It is valuable to distinguish the specification of the geometry (i.e. the choice of the primitive views and of their parameters) from the definitions of the features in the composite views (color assignment).

Furthermore, the parameter expressions and transformations that scale and position the primitives may be stored in an independent primitive instantiation graph, which is well suited for constraint-based design [Rossignac86] and is also independent of feature definitions. Executing the instantiation graph will yield the desired shape for the primitives. The execution of the sequence of color assignments will yield the subdivision for the desired views into features. This decomposition is particularly important for supporting parametric models, as in CSG-based systems.

3. TOPOLOGICAL DECOMPOSITION

3.1 Limitations of view-based selections

The STC scheme, as described so far is sufficient for working with a large variety of non-manifold models. Consider for example two primitive solids, A and B, each represented by a view that distinguishes their interior (green), boundary (red), and complement (white). The interference of these two solids may be expressed as the intersection of both green features. The contact is the intersection of the two red features. Typically, the geometric core provides simple information about the connected components of the interference or contact regions: number of components, enclosing box, or geometric approximation used for rendering. An application may need to know the topology of the region of contact, which may involve subsets of various dimensions. The set theoretic selection filters (selecting sets by providing a color vector, one color per view) may return possibly non-manifold pointsets of mixed dimensionality. Many applications may require a finer decomposition of such pointsets, and thus more powerful set selection operators.

For example, the nature of a contact region (intersection of boundary features) differs depending on its adjacency to regions of the interference. Appropriate extensions will be discussed later in the paper, but first, we must define the building blocks (atomic elements) of such refinements.

3.2 Decomposing arbitrary pointsets

Consider an arbitrary unknown pointset S. S may correspond to a feature or a selection set. Assume here that the pointsets of interest are well behaved (have nowhere dense boundaries and their intersections have a finite number of connected components). For example, the popular semi-algebraic sets exhibit

such properties. We would like to establish a decomposition of such a pointset into its parts, the natural topological constituents, so that these constituents may be easily and unambiguously identified through a topological API, independently of the particular shape of a parameterized object. This independence is key to guaranty model validity, which for example is the principal advantage of CSG over boundary-based design.

The first step in this definition isolates the interior (three-dimensional open region), iS, from the boundary, bS. Note that, although iS may have several connected possibly unbounded components, it is always well defined. The boundary bS may be further decomposed into its singular set (denoted sS) and its face set (denoted fS) which represents the union of its faces

The singular set, sS, is the set of points of bS whose neighborhood with respect to bS (i.e. the intersection with bS of an infinitely small open ball around the point) is not homeomorphic to a disk. Note that this definition is based on purely topological considerations and not on surface smoothness. Therefore cusps in a surface that bounds S will not be included in the singular set, unless they are isolated as specific features in a primitive view.

The face-set of S is the difference between the boundary of S and its singularities (i.e: fS=bS-sS). Note that fS may contain several maximally connected components called the faces of S. Note that fS may contain non-smooth edges that join the subsets of two different surfaces in some geometric realization of the primitive views. The singular set, sS, may be further decomposed into the edge-set. eS, of S and the vertex-set, vS, of S. The edge-set is the set of points of sS whose neighborhood with respect to sS is homeomorphic to an open line segment. The vertex-set, the complement of eS in sS, is the set of non-manifold points of sS.

Hence, each set S may be decomposed into four parts: iS, fS, eS, and vS, each part being possibly empty or disconnected.

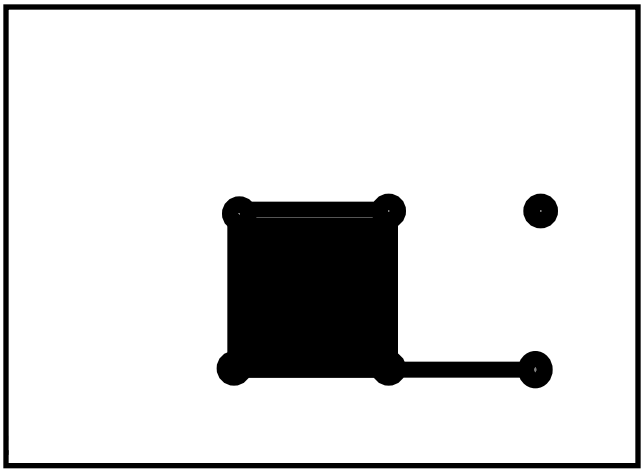

FIG 4: A mixed-dimensional set S

Consider the set S depicted in figure 4. It has three parts: its interior (Fig 5), its edge-set (Fig 6) and its vertex-set (Fig 7).

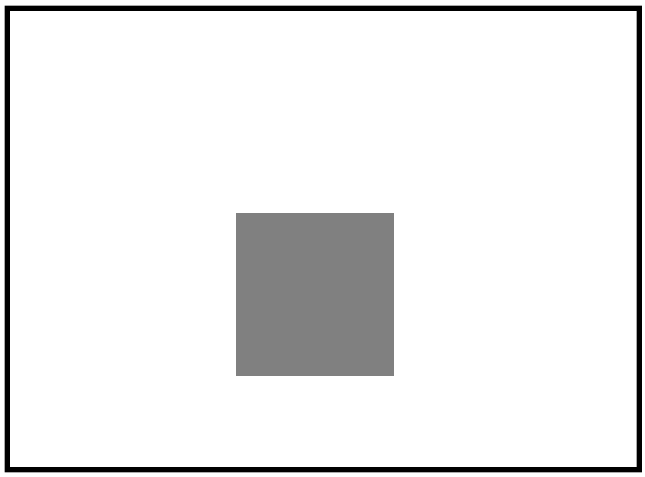

Fig 5: The face-set (i.e. the interior) of S

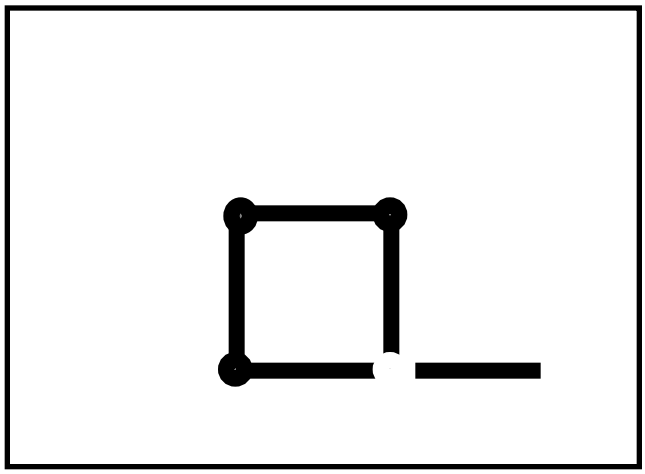

Fig 6: The edge-set of S

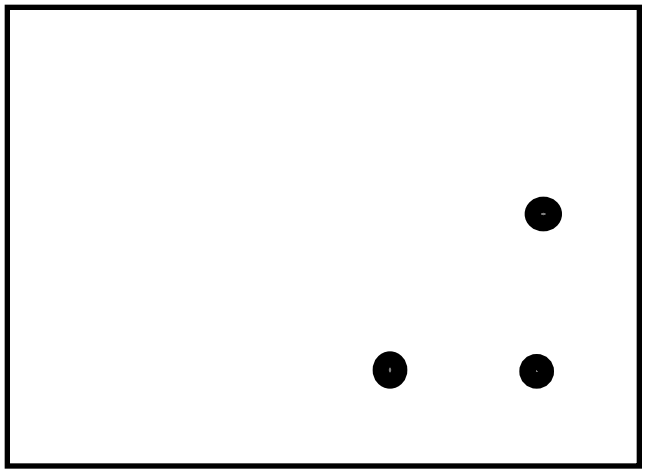

Fig 7: The vetex-set of S

The above decomposition of a set S into its interior, faces-set, edges-set, and vertex-set is unique. When a feature S is represented in a model of the entire STC as a collection of regions, the STC system must be able to correctly identify and process the iS, fS, eS, and vS parts of S, independently of the way these parts are stored in the particular representation scheme.

For example, modelers based on Simplicial Geometric Complexes [Rossignac&O'Connor89 and Sriram&...95] represent cells that are connected and relatively open subsets of smooth sub-manifolds. SGC cells are pairwise disjoint and the boundary of a higher-dimensional cell is the union of lower dimensional cells in the SGC. In particular, each connected component of a part of S (iS, fS, eS, or vS) would be represented in SGC as the union of a number of SGC cells of possibly different dimensions that form a connected dimensionally-homogeneous set.

The simplification process, when applied to a pointset or feature S, identifies the pointsets of all the parts of S. When the underlying geometric core uses cells (such as SGC cells), the simplification process associates the cells with the part of S they belong to. Note that it is not necessary to compute and to represent a simplified versions of S, as long as the exact pointset for each part of S may be identified.

On the other hand, it may be desirable to simplify the entire space decomposition so as to reduce storage, while preserving the compatibility of the regions of the decomposition with each active view. (The application may declare that some views are no-longer active and should not be referenced.)

For example, the boundary of S may contain a subset X of another feature S'. In order to represent correctly kS, S' must be decomposed into at least two regions: X and S'-X. The existence and computability of such a refinement are fundamental to the support of boundary and closure operations in STCs. A solution and proof of existence were discussed as part of the mathematical foundations of the geometric complexes that support SGCs [Rossignac&O'Connor89].

3.4 Other topological operators

So far, we have seen two techniques for decomposing space, given a collection of primitive views: Set theoretic Boolean operators (i.e.: the definition of sets as the intersection of features selected in different views by using color-vectors and their combinations through union, intersection, difference, and complement operators) and topological operators that decompose a pointset into its interior and bounding parts.

Other popular and useful operators include the complement cS, and the closure kS. Regularization, rS, may be obtained as k(iS) or kiS for short. Such operators take a set and return another set, which may be used as argument for another operation or for a color assignment in a view.

3.5 Operators on structures

A structure, A, is a collection of features in a view. It may be important to perform topological operations on the pointset of A, without destroying its internal structure, i.e. its decomposition into features.

We consider, without loss of generality, that a structure is represented by its own view, which differentiates each feature of the structure by a different color and where the complement of the structure's pointset is a single black feature.

The topological interior, iA, of a structure is a structure that has as features the intersection of the features of A with the interior of the pointset of S (i.e. the pointset of the union of the non-black features of A). Note that iA is not necessarily equivalent to the union of the interior of the features of A. For example, an open ball A composed of two features, the ball and its central point will be preserved under the interior operation, while applying the interior operator to each one of the features would eliminate the central point.

Similarly, intersection and difference operators that combine a structure A with a pointset S will trim the features of A to their intersection with S or to their difference with S.

Union, boundary, and closure operations on a structure A may add a new feature to A (i.e.: may identify a subset of the black feature in the view associated with A and will change the color of that subset to a new color, which must be specified by the application).

A semantics for Boolean operations that preserve the structure of its arguments, A and B, has been proposed as part of the CNRG specification scheme [Rossignac&Requicha91b]. Such operations may easily be supported in an STC environment, since the core modeler that supports STCs must already be able to provide a compatible subdivision of the two views associated with A and B, and since sequences of color assignments may be executed for each combination of colors from both views. However, specifying the resulting colors for all possible combinations of regions of overlap between a feature of A and a feature of B may be tedious. We suggest procedural specifications (macros) for color combination schemes. Alternatively, it may be preferable to simply not compute the result of a Boolean combination of structures, but to extract the specific subsets of interest.

3.6 Syntactic considerations

So far, our discussion of STC operations has been abstract. We do not wish to commit to any particular syntax for the STC API, because such a choices should take under considerations many programming aspects that fall largely outside the scope of this paper. Some of these issues are discussed elsewhere [Bowyer&...97].

Nevertheless, we provide below a naive API to help the reader understand the concepts developed in the paper and to help the designer of an STC system in her considerations of the appropriate syntax. We use an object-oriented notation in a high level pseudo-language that should be self-explanatory. Declarations are omitted.

A:= Cylinder(r1, l1, red, black);
B:= Cylinder(r2, l2, yellow, black);
create two primitive views, A and B, that correspond to cylindrical solid primitives in standard position, although having different dimensions (the first two parameters). Each view has two features, the cylinder and its complement. The colors of these features are defined at creation (the latter two parameters). The complement of both cylinders is a black feature.

B.move(M);
applies transformation M to B. For example, M may be a matrix that combines the effects of scaling, rotation, and translation. Note that in some design environments, it may be desired not to alter B, but to produce a new instance of B at the new location.

C:=View(black);
creates a new view and initializes it to a single black feature.

C.Paint(Intersect(A[red],B[yellow]),green);
identifies a pointset that is the intersection of the red feature in A with the yellow feature in B, creates the corresponding feature in C and paints it green.

3.7 Identifying connected components

The set selection operators discussed so far may fail to distinguish between the connected components of a part and to isolate the subset of a part that may exhibit a particular relationship to other parts. In this section, we introduce additional operators for distinguishing amongst the various subsets of a part. These operators are important for expressing finer set selections during feature definition and interrogation. We will use the term "component" to define such subsets, although a component needs not be connected.

A component of a part may sometimes be identified, independently of the particular shape of the part, by specifying topological relationship to other components of this or of other parts.

For example, the face-set, fS, of a part S may always be decomposed into the set of dangling faces and the set of bounding faces of S. We define these sets as follows.

The bounding face-set of part S is the set of points of fS that is adjacent to iS (i.e.: the set of points P such that any open ball around P intersects iS).

The dangling face-set of part S is the set of points of fS that is not adjacent to iS (i.e.: the difference between fS and its bounding face-set).

Similar definitions may be useful for identifying the sets of dangling and bounding edges and vertices,

Figure 8 and 9 illustrate such sets that decompose the face-set of Figure 6 as a part of set S in Figure 4.

The face-sets (or edge-sets for 2D models) may be further decomposed into components according to the three colors of the features they belong two or bound on each side.

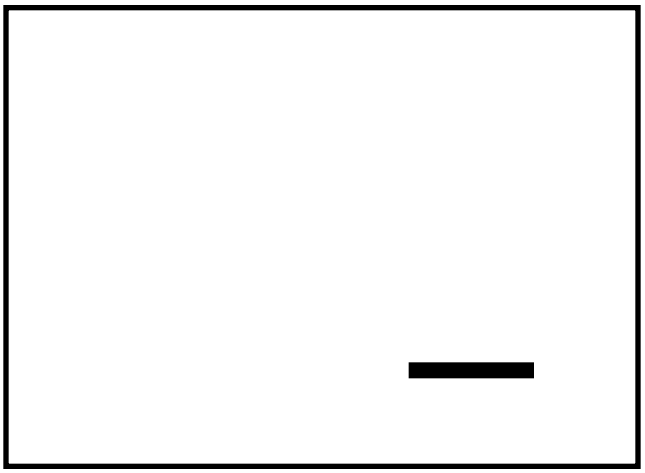

Fig 8: The dangling-edges of the set S in Figure 4.

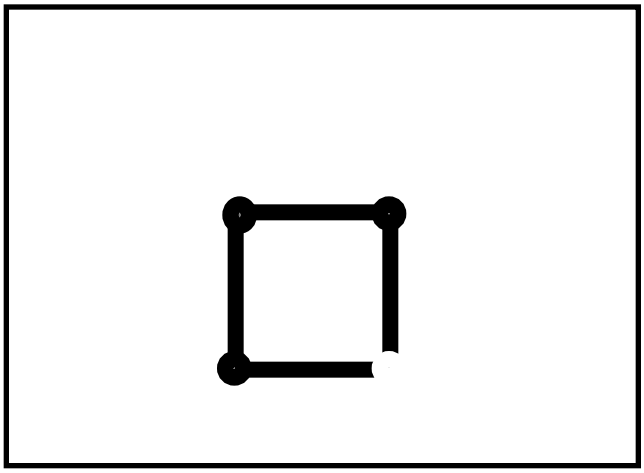

Fig 9: The bounding-edges of set S.

The components of a face parts may be further differentiated by identifying the colors of edge parts and vertex parts that bound them.

Further differentiation may be obtained by considering the different topological relationships between the face component and its bounding elements. For example, a connected subset of a red face in a given view may be distinguished from the rest of the red face by requiring that it be abutting a red edge on one side, a green edge on three sides (the green edge could be a self-intersecting singularity of the surface that supports the red face), and an isolated blue vertex (not connected to any edge that bounds the red face).

Finally a face component may be identified because it is adjacent to (i.e.: it shares a common edge with) another face component of the same view or of a different view.

Similar adjacency-based selection mechanisms may be invoked for identifying the connected components of the interior of a feature or set and the connected components of the edge and vertex parts.

Components may also be differentiated by their adjacency order. Consider an oriented edge E that bounds a red, a green, and a blue face. Several branches of each face may be abutting the edge. Such a pathological case may be simply constructed by attaching several thin red, green, and blue stripes to the edge as feathers are attached to an arrow's tail and then connecting groups of stripes of the same color by thin face ribbons that may involve twists. The ordering of faces around E may be defined by the cyclic sequence of colors corresponding to the colors of the abutting faces as encountered by turning around E in a predefined direction. If each face component was orientable, each color in the orientation sequence may be accompanied by a binary flag indicating whether the orientation of the face is compatible with the counter-clockwise direction around E or not. However, since a face component needs not be orientable (for instance it may e equivalent to a Moebius strip), this additional binary information is not always well defined.

A component of a face part may for example be identified by specifying that it is the component that follows or precedes another face component around E. A similar order may be used for edge components abutting a regular vertex of a surface [Rossignac94]. However, difficulties may arise at singular non-manifold vertices, such as the self-intersection point of a torus whose two radii are equal.

3.8 Boundary representation for STCs

We suggest to use an extension of the SGC structure to represent the subdivisions and selections defined through an STC API.

The geometric complexes introduced to support SGCs combine cells that are bounded, connected, relatively open subsets of smooth sub-manifolds, that are mutually disjoint, and whose boundary is a collection of other cells in the complex. (The term relatively open refers to the embedding of the cell in a supporting manifold of the same dimension as the cell. For example, an edge that does not form a loop is relatively open if it does not contain its end-points, because the term relatively open refers here to a curve that supports the edge. Note however, that the edge is not open relatively to a supporting surface or three-space. Similarly, a face, or 2-cell, is relatively open if it does not contain its bounding edges and vertices. Note that the open face may have missing interior edges or vertices. A sub-manifold is smooth if a normal is uniquely defined at each one of its points. For example, a cone surface minus its apex will form a smooth sub-manifold which contains two maximally connected components.

The adjacency between cells of SGCs is stored as bi-directional links between a cell and each one of its bounding cells. Links between cells of consecutive dimensions were labeled with neighborhood information which specified whether the higher dimensional cell is abutting the lower dimensional cell on the left, on the right, or on both sides. (Left and right were unambiguously defined in terms of the intrinsic orientations of both cells.)

A significant advantage of SGC cells over other schemes for representing cell complexes is the fact that an SGC n-cell need not be homeomorphic to an open n-ball, but can be multiply connected (i.e., have holes of various dimensions).

We have chosen to preserve the constraint that the cells of an STC representation be open and connected. However, they need not be smooth sub-manifolds of some surface or curve. Such a departure from the semi-algebraic domain invalidates the simple three valued neighborhood exploited in SGCs. (As explained above, a face of an STC may not be orientable and may have several branches abutting onto the same edge.)

Consequently, the links to the face cells abutting an STC edge will be organized in a cyclic manner that reflects their order around the edge. Several links to the same face cell may be presented in a cycle, one per abutting branch. When a face cell is orientable, the link to the 3D elements it bounds will contain the SGC neighborhood. Similarly, link to abutting edge cells may be ordered in a cyclic manner around a regular vertex for each side of each shell in the vertex cone. These shells correspond to the boundaries of the connected open regions formed by subtracting the abutting faces from an infinitely small sphere centered at the vertex. Typically, when the vertex is regular, there are only two regions and the two cycles are the inverse of each other.

4. CONCLUSIONS

In this paper we have introduced the following concepts.

- Each view of the model decomposes the entire space into colored features.
- Each feature S is uniquely decomposable into its natural parts (and into parts iS, fS, eS, vS) and further decomposed into components identified through adjacency, order, and topological relationships to other parts.
- The level of refinement of features (which subset of a feature may be identified) is controlled by the definition of the features in the primitive views supported by the particular geometric core best suited for the application at hand. Lack of refinement may be used to hide the nature of a particular geometric approximation scheme for curves and surfaces or to limit which of a geometric primitive are selectable.
- Composite views, which reflect space decompositions that are relevant to specific aspects of an application, are constructed through a series of painting operations that each assign a color to a collection of part components.
- The instantiation and placement of geometric primitives (which defines the geometry for the primitive views, and hence the geometry for the composite views), is achieved by executing a sequence of geometric operations that evaluate parameters and perform operations that modify the dimension, position, and orientation of groups of primitives.

- A construction model generalizes the procedural approach to parametric models [Rossignac&...88] by using two sequences of operations: the geometric transformations that affect the shape of the primitives, and the topological transformations, that define the features in the various views.
- The evaluated boundary model may be constructed to flesh out the relevant part components that support all views declared active. The model is obtained by computing the compatible subdivision induced by the features in the primitive views and by simplifying the representation while guaranteeing that each feature of an active view is represented as the union of the components of the subdivision.
- The proposed boundary model representation extends the SGC model by providing ordering information between adjacent components and by relaxing the smoothness constraint that SGC impose on the extents which support SGC parts.

These concepts lead to a systematic APIs for a new breed of geometric and topological core libraries and systems. The API supports a fully general topology in a manner that is independent of the particular geometric instantiation, and is thus suitable for parametric modeling. Furthermore, the topological entities identifiable by the API are independent of the particular geometric approximation scheme used by a geometric part of the library.

5. BIBLIOGRAPHY

[Bardis&Patrikalakis91], Topological Structures for Generalized Boundary Representations, L Bardis and N. Patrikalakis, Design Laboratory Memorendum 91-18, Department of Ocean Engineering, MIT, 1991.

[Bowyer&...95] Introducing Djinn: A geometric interface for solid modeling, A. Bowyer, S. Cameron, G. Jared, A. Middleditch, M. Sabin, and J .Woodwark, The Geometric Modeling Society, Information Geometers Ltd., Ed. J. Woodrark. 1995.

[Bowyer&...97] Ten Questions that arose in designing the Djinn API for Solid Modeling, A. Bowyer, S. Cameron, G. Jared, R. Martin, A. Middleditch, M. Sabin, J. Woodwark, to appear in the proceedings of the Shape Modeling International'97 conference, University of Aizu, Aizu-Wakamatsu, Japan, 3-6 March 1997, IEEE CS Press.

[Cavalcanti&...] Heterogeneous Object Representation Using Space Decomposition, P. R. Cavalcanti, P. C. Pinto Carvalho, L. F. Martha, IMPA-Instituto de Matematica Pura e Aplicada, Estrada Dona Castorina, 110. 22460-320, Rio de Janeiro, RJ, Brazil. pcezar@visgraf.impa.br.

[Requicha&Rossignac92] Solid Modeling and Beyond, A. Requicha and J. Rossignac. Special issue on CAGD, IEEE Computer Graphics&amp.Applications, pp. 31-44, September 1992.

[Rossignac90] Issues on feature-based editing and interrogation of solid models, J. Rossignac. Computers&Graphics, Vol. 14, No. 2, pp. 149-172, 1990.

[Rossignac94] Through the cracks of the solid modeling milestone, J. Rossignac, in From object modelling to advanced visualization, Eds. S. Coquillart, W. Strasser, P. Stucki, Springer Verlag, pp. 1-75, 1994.

[Rossignac86] Constraints in Constructive Solid Geometry, J. Rossignac, Proc. ACM Workshop on Interactive 3D Graphics, ACM Press, pp. 93-110, Chapel Hill, 1986

[Rossignac&...88] Interactive Design with Sequences of Parameterized Transformations, J. Rossignac, P. Borrel, and L. Nackman, Proc. 2nd Eurographics Workshop on Intelligent CAD Systems: Implementation Issues, April 11-15, Veldhoven, The Netherlands, pp. 95-127, 1988.

[Rossignac&O'Connor89] SGC: A Dimension-independent Model for Pointsets with Internal Structures and Incomplete Boundaries, J. Rossignac and M. O'Connor, in Geometric Modeling for Product Engineering, Eds. M. Wosny, J. Turner, K. Preiss, North-Holland, pp. 145-180, 1989.

[Rossignac&Requicha91] Constructive Non-Regularized Geometry, J. Rossignac, and A. Requicha. Computer-Aided Design, Vol. 23, No. 1, pp. 21-32, Jan./Feb. 1991

[Sriram&...95] GNOMES: An objet-oriented nonmanifold geometric engine, R. Sriram, A. Wong, and L.-X. He, Computer-Aided Design, vol. 27, no. 11, pp. 853-868, November 1995.

[Weiler&McLaclan91] Selection sets and filters in geometric modeling and their application in a non-manifold environment, K. Weiler and D. McLachlan, in Product Modeling for Computer-Aided Design and Manufacture, Ed. J. Turner, J. Pegna, W.J. Wozny, North-Holland, pp. 117-139.