In Helsinki, a small but general set of manipulative operations for boundary models of solid objects has been used to construct a comprehensive solid modeling system.

GWB: A Solid Modeler with Euler Operators

Martti Mantyla and Reijo Sulonen
Helsinki University of Technology

Any software system making use of three-dimensional computer graphics must include some means for representing three-dimensional objects. Conventional generative graphics uses rather straightforward wireframe models consisting of lines and points. While these models allow easy construction of line figures, they are not adequate for more general operations. For instance, a wireframe model is not generally sufficient for calculating the volume of an object or for eliminating hidden lines.

These problems are emphasized in computer-aided design systems. A CAD system for mechanical engineering, for instance, should include a variety of advanced operations, such as engineering analysis, drafting, and computer-aided manufacturing. All these operations should be based on a single representation or on multiple but consistent representations created by the user of the system.

Object representations in CAD must be capable of generating all data needed during the design cycle of an object. Since the geometric properties of the object form the basis for engineering analysis and CAM operations, it is natural to separate nongeometric data from data dealing with the geometric shape of the object—the geometric model.

To ensure that the information stored in a geometric model is potentially sufficient for calculating any geometric properties, the model should be an unambiguous representation of the object. This means that each model designates a unique physical solid. The geometric models created by a solid modeler satisfy this requirement.

There are two main approaches to solid modeling: constructive solid geometry and boundary representation. In a CSG model, solids are described by Boolean combinations of basic solids; in a BR model, solids are described by a collection of faces, which in turn are represented by their bounding edges and vertices.

The choice between these alternatives depends on the type of operations the solid modeler is expected to support. Boundary modelers tend to support stepwise construction of the models more easily than CSG modelers but require greater storage.

The Geometric Workbench, an experimental solid modeling system we have developed at the Helsinki University of Technology, uses a variation of the BR approach. Our building blocks in GWB are a set of "atomic" functions called the Euler operators. These functions allow the incremental manipulation of BR models, while restoring the "well-formedness" of the underlying data structures. We describe several algorithms based on the use of Euler operators, with special emphasis on an inversion algorithm that parses a given well-formed model into a sequence of Euler operators.

Boundary representations of solids

Mathematically speaking, boundary models describe an object by its topological boundary. The boundary is represented as a collection of faces, which in turn are represented by their bounding edges and vertices. We use the convention that faces need not be simply-connected but may include cavities; the components of the boundary of a face are referred to as loops.

Usually faces are chosen so that the geometric form of each face can be represented as a single (piecewise) parametric function. The modeling space of a BR model is determined by the face geometries allowed. The modeling space considered in this article is the set of basic solids, which may be defined as three-dimensional rectilinear polyhedra whose boundaries are connected manifolds. In this case each face is a planar polygon whose shape is determined by the coordinates of the vertices of its loops.
One of the loops represents the "outer" boundary of the face, while the others describe its cavities. We shall use the term ring to denote an interior loop; for instance, a face with one cavity is said to have one ring.

A boundary model might be represented simply as a set of multiconnected polygons. Each loop of each polygon would then be represented as a list of vertices. To speed up various operations on the model, boundary modelers frequently store other explicit data on the connections between faces, loops, edges, and vertices of the solid. These data are commonly referred to as the topology of the model, while data such as face equations or vertex coordinates are referred to as the geometry of the model.

With regard to explicit topological information, a large number of variants of boundary representations exist. Baer, Eastman, and Henrion enumerate nine distinct topological relations which may be included in a particular representation. The choice among alternatives depends on the mixture of operations desired for the model.

Not all collections of faces define a valid physical solid. The topological integrity of a BR model imposes restrictions on face collections to ensure the validity of the model. For instance, each edge must belong to exactly two (not necessarily distinct) loops, and the ordering of edges in loops must be consistent throughout the model. To satisfy the latter condition, outer loops can be specified to be oriented clockwise (as viewed from outside the solid), and rings can be oriented counterclockwise. Geometric integrity is satisfied when the shape assigned to faces is consistent with the topological information; for instance, faces may intersect only at common edges or vertices, if at all.

Atomic solid-modification operations

As indicated above, any boundary representation is a data structure consisting of faces, loops, edges, and vertices. Since these data structures are often very complex, creating them is laborious and error-prone. For instance, the object depicted later in Figure 9 includes 79 faces, 231 edges, and 154 vertices.

Fortunately, the modifications of these models can be divided into simple atomic operations. Let us consider the cube represented by six faces, twelve edges, and eight vertices as shown in Figure 1(a). Consider for a moment the effect of removing the edge marked by an arrow. In the resulting object, Figure 1(b), the two faces which meet at the edge are united and the remaining collection of five faces, eleven edges, and eight vertices is generated—a simpler model. Of course, the geometry of the face created by the uniting operation is no longer planar.

Performing the same operation on the edge marked in Figure 1(b) creates the object in Figure 1(c). It has an edge belonging twice to the loop indicated by the dashes. The "upper" vertex of the edge is adjacent to no other edges. Another operation removes such a combination of an edge and a vertex and creates the model in Figure 1(d). These two operations can remove all edges and all vertices except one. A third operation can remove the remaining stripped-down, one-vertex model.

Obviously, each "removal" operation has an inverse "creation" counterpart. While the operations described above can be used to destroy models of solids, their positive operations can be used for the perhaps more interesting purpose of creating models. So the inverse of the third removal operation creates an "initial object." The inverses of the two other operations add new vertices, edges, and faces to create models of any complexity. Together the destructive and the creative operations allow us to perform arbitrary modifications necessary in BR models.

Euler operators

The operations informally described in the previous section are examples of Euler operators. We call the three destructive operators "kef," "kev," and "kvsf," for "kill edge and face," "kill edge and vertex," and "kill vertex, solid, and face." Similarly, their inverses are...
Table 1.
Euler operators.

<table>
<thead>
<tr>
<th>OPERATOR</th>
<th>EXPLANATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>evsf(f,v)</td>
<td>MAKE VERTEX, SOLID, FACE</td>
</tr>
<tr>
<td>kvsf()</td>
<td>KILL VERTEX, SOLID, FACE</td>
</tr>
<tr>
<td>mev(v1,v2,v3)</td>
<td>Make EDGE, VERTEX</td>
</tr>
<tr>
<td>kev(v,v)</td>
<td>KILL EDGE, VERTEX</td>
</tr>
<tr>
<td>mef(v1,v2,v3)</td>
<td>MAKE EDGE, FACE</td>
</tr>
<tr>
<td>kef(v)</td>
<td>KILL EDGE, FACE</td>
</tr>
<tr>
<td>kemr(v)</td>
<td>KILL EDGE, MAKE RING</td>
</tr>
<tr>
<td>mekr(v1,v2,v3)</td>
<td>MAKE EDGE, KILL RING</td>
</tr>
<tr>
<td>kfrh(v1,v2,v3)</td>
<td>KILL FACE, MAKE RING, HOLE</td>
</tr>
<tr>
<td>sfrh(v1,v2,v3)</td>
<td>MAKE FACE, KILL RING, HOLE</td>
</tr>
<tr>
<td>semv(v1,v2)</td>
<td>SPLIT EDGE, MAKE VERTEX</td>
</tr>
<tr>
<td>jekv(v1,v2)</td>
<td>JOIN EDGES, KILL VERTEX</td>
</tr>
</tbody>
</table>

Figure 2. Cube topology: a plane model.

Figure 3. Euler operators: plane models.
called "mef," "mev," and "mvsf," where "m" stands for "make."

Euler operators derive their name from the well-known Euler's Law: in any simple polyhedron, the numbers of faces \((f)\), edges \((e)\) and vertices \((v)\) must satisfy the equation

\[ v - e + f = 2. \]

The formula may be generalized to arbitrary solids by introducing three other parameters, namely (1) the total number of rings (cavities in faces, \(r\)) in the solid, (2) the total number of holes \((h)\) through the solid, and (3) the number of disconnected components \((s)\) in a collection of solids. The general formula is

\[ v - e + f = 2(s - h) + r. \]

As noted by Braid, Hillyard, and Stroud, just five operators (with their inverses) are sufficient for describing all objects satisfying the Euler formula. While these five may be chosen in several ways, considerations of modularity and mutual independence have led to only small variations in the collections appearing in the literature.

We list the operators chosen for this project in Table 1. The collection includes the three operators already discussed and two other operator-inverse pairs related to creation of rings \((\text{kemr, mekr})\) and holes \((\text{kfmrh, mfkrh})\). To aid geometric operations, an auxiliary "split-edge" operator is included \((\text{semv, jekv})\).

Operators creating faces, edges, and vertices assign identifiers for them; in the table these output parameters are in italics. Although not shown in the table, vertex coordinates are stated when a vertex is created. Additional orientation parameters must be given when the parameters listed are not sufficient to specify the desired effect unambiguously.

Euler operators work on the topology of a boundary model—that is, on the relative arrangement of its faces, edges, and vertices. To describe the effects of the operators, we can represent the topologies they create by planar graphs that we call plane models. The nodes and arcs of the graph represent the vertices and edges, and the planar areas bounded by the arcs of the graph represent faces. For instance, the topology of a cube is represented by the graph of Figure 2. Plane models of each of the Euler operators are shown in Figure 3. Note that in Figure 3(f), kemr creates a special case of a loop that has no edges at all. A similar loop is also created by the mvsf operator in Figure 3(a).

The effects of the \(\text{kmfrh/mfkrh}\) operators (Figures 3(g) and (h)) are not easily represented by plane models, since they affect only the labeling of the areas. Effectively, \(\text{kmfrh}\) transforms the boundary of one face \((f_1)\) into a ring of another \((f_2)\) and ordinarily creates a hole through the object. This is pictured in Figure 4(a). Operators \(\text{kmfrh/mfkrh}\) modify the topological genus of the solid—the number of holes through it. The same operators can also modify the connectivity of the solid—the number of components in it. For instance, the "gluing" of two touching cubes may be accomplished by joining their adjacent faces by the \(\text{kmfrh}\) operator, as depicted in Figure 4(b). Thus \(\text{kmfrh}\) really decreases the value of the expressions \((s - h)\) by unity either by removing a solid or by adding a hole, while \(\text{mfkrh}\) increases it by the reverse operation.

For further clarification, we represent in Figure 5 the creation of a cube with a hole. The "exterior" cube is created with operations (a) through (f). The "bottom" of the hole to be drilled is created with operations (g) through (i), while its side faces are created by operations (j) through (k). The last operation \((\text{kmfrh})\) is also shown in Figure 4(a). For brevity, the parameter lists of the operators have been excluded.

Analysis of the plane models shows that Euler operators guarantee the topological validity of BR models. For instance, each model is consistently oriented. On the other hand, in the algorithms using the operators, geometric validity must be verified, since the intermediate geometries generated can be invalid.

In a formal sense, Euler operators form a sufficient set of solid definition and manipulation operations. Even so, a solid modeler must include more user-oriented ways to create and handle solid models. Euler operators themselves are rather unintuitive and primitive and should not be directly visible to the end user. Nevertheless, it is advantageous to use them as primitives of higher-level operations to ensure that the models created are well formed.

In the following section we describe in detail various operations and tools for solid modeling which can be constructed from the primitive Euler operators. To give some practical flavor, we first discuss the operations needed in a solid modeler based on our experience with the GWB system.
QWB: structure of a solid modeler

The Geometric Workbench is designed to be an open-ended solid modeler that can be tailored for particular applications. Its structure is diagrammed in Figure 6. GWB provides the basic object manipulation operations in the form of Euler operators, and it provides query functions for scanning the underlying data representation. These primitives allow the creation of extendable tools for solid definition and manipulation.

Solid-definition operations create solid models by means of parameterized functions or drafting-like commands. These models can be manipulated and combined by solid-manipulation operations such as the Boolean set operations. Thus definition operations operate on one solid at a time, while manipulation operations operate on collections of solids.

The internal data structures created by these operations are stored in a workspace, from which they can be retrieved and used to create graphic displays or to calculate geometric properties. The objects created by solid-definition operations can be represented in secondary storage simply by the sequence of the definition commands used. When needed, the internal data structure can easily be reconstructed from this external data format.

Unfortunately, the opposite is true for the solid-manipulation operations, which effectively destroy the straightforward correspondence between the external representation of a solid (for instance its Euler-operator sequence) and the internal data structure. It would clearly be beneficial to be able to reconstruct a list of Euler operators that conveys the information of the internal data structure. This brings us to what we call solid inversion, which is covered in the last part of this section.

Figure 5. A sample definition.
Martti J. Mantyla is a research associate with the Laboratory of Information Processing Science at the Helsinki University of Technology, Finland. Since 1980, he has worked in the CAD project headed by Prof. Sulonen (below) and financed by the Finnish Academy. His research interests include geometric modeling, computer-aided design, computer graphics, and database management. He received the MSc in computer science and mathematics in 1979 and the Licentiate of Technology in computer science in 1981, both from the Helsinki University of Technology. He is currently writing his doctoral dissertation on topological manipulations and interrogations in geometric modeling. He is a student member of the IEEE and the ACM.

Reijo Sulonen is a professor of computer science at the Helsinki University of Technology and currently also leads the CAD project of the Finnish Academy. He has received the PhD in computer science from the Helsinki University of Technology in 1975. Besides various activities in Finland, he has worked as a visiting research associate at Brown University in 1972-1973 and as a visiting associate professor at Stanford University in 1977-1978. His current research activities include computer graphics, computer-aided design, and database technology.

Sulonen is a member of the IEEE and an associate member of the ACM. He is also active in the Eurographics Association as a member of its executive committee and an editor for its Computer Graphics Forum.

Superior Interactive Design 3-D Graphics Workstation

If you need high-performance calligraphic design capability with 4K x 4K resolution and true high-speed stroke graphics, your best choice is the new IMI-500 Intelligent Graphics Design Workstation. In fact, the cost-effective and user-friendly IMI-500 is priced about 30% below well-known competitive systems which offer much lower performance.

If the IMI-500 sounds like it could benefit your graphics design projects, we'd welcome the chance to show you why the IMI-500 is perfect for stand-alone or large system applications. Why not call or write for complete details, today!

INTERACTIVE MACHINES INCORPORATED
2500 B Townsgate Road
Westlake Village, CA 91361  (213) 707-1880

Reader Service Number 15