OBTAINING HALF-SPACE CSG REPRESENTATION OF SOLID MODELS INTERACTIVELY

**Abstract**

Constructive Solid Geometry (CSG) is one of the techniques in solid modeling. CSG is commonly used in 3D computer graphics, CAD, CAM, CAE, robotics, animation, medical imaging, and reverse engineering of design process. In this study an interactive CSG modeler tool has been developed for interactively creating constructive solid geometry model graphical data. Especially, massively parallel graphics systems called ‘pixel-based’, which generate CSG scenes in real-time, accept scenes in terms of half spaces. Unbounded planes divide space into half-spaces. By combining half-spaces through regularized set operations, new complex solid models are obtained. This work is motivated by the need of obtaining the half-space CSG representation data of a solid model interactively. The tool can export the CSG structure of the model together with its half-space primitives. This data can be used in pixel-based processors, reverse engineering of the design, animation, and robotics because of its hierarchical and constructive structure.

Nilgün ÖZGENÇ
Elektrik-Elekttronik Mühendisliği

Ulus ÇEVİK
Elektrik-Elekttronik Mühendisliği

ÖZET


Anahtar Kelimeler: Katı modelleme, yapışsal katı geometri, yarı-uzay, görüntü uzayı

ABSTRACT

Constructive Solid Geometry (CSG) is one of the techniques in solid modeling. CSG is commonly used in 3D computer graphics, CAD, CAM, CAE, robotics, animation, medical imaging, and reverse engineering of design process. In this study an interactive CSG modeler tool has been developed for interactively creating constructive solid geometry model graphical data. Especially, massively parallel graphics systems called ‘pixel-based’, which generate CSG scenes in real-time, accept scenes in terms of half spaces. Unbounded planes divide space into half-spaces. By combining half-spaces through regularized set operations, new complex solid models are obtained. This work is motivated by the need of obtaining the half-space CSG representation data of a solid model interactively. The tool can export the CSG structure of the model together with its half-space primitives. This data can be used in pixel-based processors, reverse engineering of the design, animation, and robotics because of its hierarchical and constructive structure.

* Yüksek Lisans Tezi-MSc. Thesis*
Key Words: Solid modeling, Constructive Solid Geometry, half-space, image space, object space

Introduction

Constructive Solid Geometry (CSG) represents a powerful solid modeling technique. The idea of CSG is to combine simple 3D shapes with more complex ones with regularized Boolean set operations in 3-dimensional space. Even though CSG is an established technique and it is well understood, it has not become the main method in software systems due to the complex implementations of rendering algorithms to display CSG models. In CSG, the most basic shapes are called primitives. A CSG primitive must be a solid, i.e., given in a way that interior and exterior regions of the primitive are clearly defined. Two primitives or CSG shapes can be combined by one of the following regularized Boolean set operations to define a more complex CSG shape:

- Union: The resultant solid occupies the volume of the both primitives occupy.
- Intersection: The resultant solid occupies the volume of the operand solids that is coincident to all. When the operands do not have a coincident volume, the resultant solid is null.
- Difference: The resultant solid occupies the volume of the operand solids that the other does not occupy.

A CSG solid is represented by a function of the primitives and operations used in its composition. Its structure is a tree where these operations are hierarchized, called the CSG tree. Each operation applied is represented as an internal node (no leaf), and each primitive as a leaf node. The CSG representation is an ordered binary tree with non-terminal nodes operators and terminal nodes representing either primitive or transformation leaves which contain the arguments of rigid motions. The operators can either be rigid motions or Boolean set operations. Because of the mathematical properties of regularized Boolean operations, the resulting CSG shapes are always solid. This is an important advantage over other 3D modeling techniques, which often miss polygons and generate unclosed models. In Figure 1, a CSG tree structure of a model is shown.

CSG modeling and rendering is directly available in several graphics systems for offline rendering, such as POV-Ray or RenderMan. Those graphics systems are used to generate photo-realistic images, and they are not suited for real-time rendering, though. Rendering CSG shapes in real-time using and taking advantage of graphics hardware is difficult, in particular if the CSG shape is modified interactively.

Because of the ease-of-use, CSG lends itself perfectly for use in hybrid representation schemes where it is used as the front-end for a solid modeling engine that processes CSG trees into an output representation (normally b-reps) prior to rendering. Of course, it is also possible to render the completed solid directly from the CSG tree itself without transforming it into an intermediate form.
There are two main approaches of rendering of CSG constructed shapes: image-space approaches and object-space approaches. Object-space CSG rendering algorithms, also known as surface classification-based CSG algorithms (since the surfaces of the primitives contained in the CSG tree are evaluated into an intermediate form prior to rendering) include both boundary evaluation and hybrid algorithms such as CSG/solid-sweep, CSG/general-sweep, and CSG/b-rep (Requicha, 1980). Here, the CSG trees are normally converted to triangles that can then be easily sent to a rendering pipeline such as OpenGL. Of these basic algorithms, the CSG/b-rep hybrid algorithm has been improved to use binary space partitioning (BSP) trees for efficient CSG operations. In object-space approaches, the boundary of the CSG shape can be calculated mathematically and stored in a polygonal model that then is sent to graphics hardware. This is practical for static CSG shapes, but for CSG shapes that are modified interactively, the expensive calculation of the boundary must be repeated for each rendering frame, forbidding animated real-time display. Most of the solid modeling packages use this approach to implement CSG for construct solid models.

On the other hand, image-based CSG rendering algorithms can determine and store the visible parts of the CSG shape directly in the frame buffer of the graphics hardware. The result of an image-based CSG algorithm is, therefore, just the image of the CSG shape. Based on recent advances in graphics hardware, images of CSG models can be generated instantaneously and, for models of considerable complexity, in real-time.

Image-space CSG rendering approaches draws the CSG trees directly by performing view-dependent surface clipping and visible surface determination on a per-pixel basis. There are a variety of these point classification algorithms, including ray-casting, scan-line methods, and hardware-based z-buffer algorithms. This latter family of z-buffer algorithms has recently become the most promising avenue of research, and several different algorithms have been developed. The
A common feature of all of these algorithms is that they use multiple z-buffers to draw CSG trees in multiple passes and merge the results into a final image. Image-based CSG does not analytically calculate the geometry of 3D objects, but most uses of CSG do not require the explicit 3D geometry and are satisfied with the image of the model. Still, image-based CSG is not commonly used in real-world applications today. Very often, the primitives of CSG shapes are only sketched as wire-frame images, which leads to difficult understanding of the final 3D-shapes' look. Table 1 lists the advantages and disadvantages of each CSG rendering algorithms.

<table>
<thead>
<tr>
<th>Table 1. Advantages and disadvantages of CSG algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Image-space Algorithms</strong></td>
</tr>
<tr>
<td><strong>Advantages</strong></td>
</tr>
<tr>
<td>. Faster evaluation</td>
</tr>
<tr>
<td>. Uses only the CSG representation</td>
</tr>
<tr>
<td><strong>Disadvantages</strong></td>
</tr>
<tr>
<td>. CSG tree must be evaluated render update</td>
</tr>
<tr>
<td>. CSG sub trees cannot be cached</td>
</tr>
</tbody>
</table>

In this work, we developed a solid modeler tool to interactively obtain graphical data of solid models that can be used in image-based rendering of CSG models. In the tool, the CSG model is constructed from the boundary represented primitives (such as, cylinder, box, sphere, and cone). CSG operations are performed in the object space by applying the regularized Boolean set operators on the boundary represented primitives. A model is first constructed on the image space and then is rendered using an algorithm based on triangulated faces, which is the most commonly used in rendering. Since the solid model is constructed on the object space, the characteristics of the CSG model can be changed interactively at any stage of the modeling process using the tool.

The exported format of the CSG model is in half-space format. Especially, massively parallel graphics systems called ‘pixel-based’, which generate CSG scenes in real-time, accept the scenes in terms of half spaces (Çevik, 1996). The half-space format is also useful in animation and reverse engineering of the models. In geometry, a half-space is either of the two parts into which a plane divides the 3D space. More generally, a half-space is either of the two parts into which a hyper plane divides an affine space. One can have open and closed half-spaces. An open half-space is either of the two open sets produced by the.
subtraction of a hyper plane from the affine space. A closed half-space is the union of an open half-space and the hyper plane that defines it. In Figure 2 the plane and two formed half-spaces are shown.

\[
\text{Plane Equation } F(x,y,z) = 0 \\
(a x + b y + c z + d = 0)
\]

For all points in this halfspace
\[F(x_i,y_i,z_i) < 0\]

For all points in this halfspace
\[F(x_i,y_i,z_i) > 0\]

Figure 2. The half-space representation

In half-space representation, the surfaces of the solid are defined by the plane equations that form the half-spaces. The direction of the surface normal determines which half space is selected. If the viewpoint is outside of the half-space defined by the surface plane then this surface is a front (visible) surface. If the viewpoint is inside the half-space defined by the surface plane then this surface is a back (hidden) surface.

The half-space represented solids are rendered in image space. The pixel operations on these solids can be processed in parallel to shorten the rendering time. This representation is very powerful for animation purposes because of its parallelism. Since the Boolean set operations are applied in image space, the half-space representation is also used in rendering algorithms of the pixel-based graphical processor units. Because of their hierarchical and constructive structure, the half-space represented CSG models are useful in reverse engineering of the solid models.

Material and Method

The modeler tool has been developed with Java and Java3D API. Java3D API is an application-programming interface used for writing 3D applications and applets. It is capable of describing very large virtual worlds. It can provide advanced rendering and interaction capabilities, it can also enable collaborative access of 3D information over the network, and has a flexible data input mode.

Java3D is a scene graph based 3D API for the Java platform from Sun Microsystems. A Java3D scene is defined as a tree-like graph structure, which is
traversed during rendering (Java3D API). The BasicUniverse object is the root of our scene graph and provides the overall framework for scene representation. A Locale object defines a high-resolution position within a BasicUniverse, and serves as a container for the collection of BranchGroup rooted sub graphs (branch graphs) at that position. Objects within a Locale are defined using standard double precision, relative to the origin of the Locale. On the left side of the scene graph is a content branch, which contains the nodes that describe the actual object in the scene and on the right-hand a view branch, which contains nodes that specify viewing related conditions.

The modeler is a hybrid modeler using CSG as a solid model representation and boundary representation as a primitive representation. There are four solid primitives: box, cylinder, cone, and sphere. In Figure 3 the GUI and the primitives are shown. The primitives are represented in polygonal mesh data format. The primitives are defined by setting their related parameters. The user can change the parameters of the primitive to get different solid shapes. The primitives are used to compose new solid models. The regularized set of Boolean operations is used to compose new solids. The composed solid is represented by a CSG tree.

For implementing Boolean operations on b-rep represented solids the Java 3D API which is developed in the study given in (Castanheria, 2003) was used. It is sufficiently efficient and easy to use, being able to be used by any application that makes use of the API Java 3D. It functions in the following way: two solids of the solid type (a specialization of the Shape3D) are passed as parameters for the library. After that, the relative solid to the application in two solids of one of the available Boolean operations is requested by it: union, intersection, and difference. The technique used for performing Boolean operations on b-rep solids is an adaptation of the algorithm given in (Laidlaw, Trumbore, Hughes, 1986). The algorithm operates on two objects at a time. The routines can be called successively using the results of earlier operations to create objects that are more complicated. Each object is represented as a collection of polygons and vertices; each spatially distinct vertex is represented exactly once, and each polygon contains a list of references to vertices. Each polygon also contains a normal that points outwards from the object. Each vertex contains a list of references to other vertices connected to it by an edge.

The algorithm first subdivides all polygons in each of the objects so that no two polygons intersect. Two non-coplanar polygons intersect if a vertex of one lies in the interior of a face or edge of the other, or if an edge of one crosses an edge or face of the other. Polygons that share a vertex or an edge, or that are coplanar, do not intersect.

In the CSG modeler, the half-space data is created as 3D plane equations, and some other data derived from each 3D plane equations. The data is created for each primitive of the CSG solid. In addition, the Boolean expression data that forms the compound solid is created. This data was used as input data in the simulation work of the CSG algorithm described in (Çevik, 1996, and Koç, 1999). The study related with these works was about developing a rendering algorithm for
CSG models, which permits hidden surface removal and object interference testing to be provided as hardware primitives. Also with these algorithms, surface sorting problem, which exits during the hidden surface removal stage, was eliminated for both convex and concave objects.

Figure 3. GUI and primitives of the modeler tool

In the CSG modeler, plane equations are driven from the vertices of triangular meshes. The plane equation is $AX + BY + CZ + D = 0$ where $(A, B, C)$ is outward face normal of the plane and $|D|$ is the perpendicular distance of the plane from the origin. Given the vertices $V1, V2, V3$ in counter-clockwise order the normal vector of the plane is calculated as $N= (V2 – V1) \times (V3 – V1) = (A, B, C)$ where $\times$ denotes cross product. The value of $D$ is simply the dot product of the surface normal with any point in the polygon. The formulation is $-D=N \cdot P$

Results and Discussion

The modeling tool has been developed in Java. The main advantages of Java are its compatibility across different systems/platforms and having the ability to be run remotely through web browsers. Using Java 3D as a graphics engine has also the additional advantage of rapid application development, because Java 3D API incorporates a high-level scene graph model that allows developers to focus
on the objects and the scene composition. Java 3D also promises high performance, because it is capable of taking advantage of the graphics hardware in a system.

In Figure 4, some mechanical parts that are designed with the modeling tool are shown.

The half-space CSG format data exported from the modeling tool was successfully used as input in the simulation study of the image based CSG rendering algorithm of the pixel based system which was developed by (Cevik, 1996, and Koç, 1999).

Figure 4. Some mechanical parts designed with the modeling tool

Conclusions

The CSG modeling tool can export the graphical data of the final model in a new general format. The format includes: The CSG expression that forms the final model and the graphical data of the each primitive in both polygonal and half-space format. In addition, the final model’s graphical data is exported in both half-space and polygonal representation forms. The half-space represented CSG format of the solid models can be used in; reverse engineering, pixel based rendering algorithms, simulation work of pixel based GPU algorithms, animation, and robotics.
Planned future work includes speeding up and easing the modeling process and constructing the CSG tree structures of imported boundary represented models. Modeling process will be speeded up by adding more predefined primitives, adding predefined features (such as holes, fillets, rounds, and adding constraint definitions. To construct the CSG tree structure of a boundary model, firstly the model is imported in the tool in its own internal format and then using the boundary to CSG conversion algorithms, the CSG tree structure of the boundary model will be obtained. Obtaining the structural CSG models of the boundary models are useful in reverse engineering for obtaining the structural design and components of the model.

References