Non-evaluated manipulation of complex CSG solids

Rafael J. Segura, Francisco R. Feito, Juan Ruiz de Miras
Universidad de Jaén
Escuela Politécnica Superior
Avda. Madrid, 35
Spain (E) 23071, Jaén
rsegura@ujaen.es, ffeito@ujaen.es, demiras@ujaen.es

ABSTRACT

One of the most important problems to solve in Solid Modeling is computing boolean operations for solids (union, intersection and difference). In this paper we present a method to obtain the boolean operators based on covering the solids by simplices without evaluating the boundary. The representation of the obtained solid does not correspond with the minimal boundary of the solid, but using the appropriate algorithms it is possible to calculate some properties of the final solid, such as point-in-polyhedron test, visualization, volume or octree generation. The proposed method is also suitable for complex solids bounded by triangular meshes or CSG with polyhedral primitives.

Keywords

Geometric Modelling, Solid Modelling, boolean operations.

1. INTRODUCTION

Computing boolean operations between solids is a well known formal problem, specially for solids represented using a B-rep scheme. For other schemes of representation, the problem is in practice finally reduced to the B-rep problem (boundary evaluation). However, due to the complexity of computing, most approaches tend to do some simplification; some solutions are only theoretical, but some authors consider that the aim of Computational Geometry must be the finding of useful algorithms that can be implemented in practice[Chaz,Lee96]; other solutions reduce the problem to 2D [Gardan96]. Finally, other authors propose algorithms that work only with convex faces [Sugi94,Prep88,Chaz92]. But usually most of the solids used in practice, specially mechanical pieces, have not simple faces, but complex ones, with holes or other kind of faces.

An alternative to boundary evaluation is to compute boolean operations without evaluating the final solid. This method is very useful in CSG representation [Bron90], because in this scheme of representation the evaluation of the solid boundary is not needed. So, it is specially used to display the solid obtained by making boolean operations.

Using the initial idea proposed by Torres and Feito[Tor93,Fei97a], we have developed a system for Solid Modelling, valid for any type of solid with planar faces (concave or convex, with or without holes, manifold or non-manifold). We have developed in a satisfactory way robust and efficient algorithms to solve the inclusion of points in a solid[Fei97b], and to study the intersection of a segment (or ray) and a polygon[Seg98].

As a result of this research, a method to calculate boolean operations for Solid Modelling is presented. The proposed method does not evaluate the boundary of the solid, and it uses the formal definitions of these operations to work. The boundary of the obtained solid is not minimal, i.e., it does not correspond with the real boundary of the solid. Instead of it, an intermediate representation is proposed, but the main advantages of this representation are speed, validity and robustness of the method.

In the first section the theoretical foundations of Solid Modelling by Simplicial Coverings is presented; these definitions allow us to obtain a theorem to represent solids. Later, we will propose a data structure to represent solids in 3D. After that, algorithms to compute boolean operations between general polyhedra with planar faces will be presented. Finally, the application of this method to complex solids bounded by triangular meshes is shown..

Journal of WSCG, Vol.12, No.1-3, ISSN 1213-6972
WSCG'2004, February 2-6, 2003, Plzen, Czech Republic.

2. THEORETICAL FOUNDATIONS

Definition 1. Let $x \in \Re$. The function sign(x) is defined as

$$sign(x) = \begin{cases} 1 & if \quad x > 0 \\ 0 & if \quad x = 0 \\ -1 & if \quad x < 0 \end{cases}$$

Definition 2. Let T=(A,B,C) be a triangle; the signed area of T (denoted by [T]) is defined as

$$[T] = sign\left(\frac{1}{2} \cdot \begin{vmatrix} x_a & y_a & 1 \\ x_b & y_b & 1 \\ x_c & y_c & 1 \end{vmatrix} \right)$$

Let four points be A,B,C,D $\in$ R^3. The signed volume of the tetrahedron of vertices D, A, B, and C, denoted by [DABC], is defined as

$$[DABC] = sign\left(\frac{1}{6} \cdot \begin{vmatrix} x_a & y_a & z_a & 1 \\ x_b & y_b & z_b & 1 \\ x_c & y_c & z_c & 1 \\ x_d & y_d & z_d & 1 \end{vmatrix} \right)$$

It is said that a triangle/tetrahedron is a positive triangle/tetrahedron if its signed area/volume is positive It can be easily proved that a tetrahedron has positive orientation (that is, the remaining vertices are ordered counterclockwise with respect to one vertex) if the signed volume is positive.

Definition 3. The signed volume of a pyramid *P* with vertex *V* and base $F(V_1V_2 \ldots V_n)$, is denoted by [P] and is computed as

$$[P] = \sum_{i=1}^{n} [VQV_iV_{i\oplus 1}]$$

being *Q* an arbitrary point laying on plane defined by F. If the vertex of the pyramid coincides with the origin of co-ordinates it is said to be an *original pyramid*.

Definition 4. The sign of a face *F* of a general polyhedra, denoted by *[F]*, is the sign of the pyramid obtained by joining the face with the origin of co-ordinates..

Theorem 1.[Fei97a]} *Generator System*. Let *S* be a solid with faces $F_1F_2...F_m$, given in consistent orientation (the normal vector goes outside the solid). Then

$$S = \sum_{i=1}^{m} [P_i] \cdot P_i$$

where P_i represents the original pyramid obtained by joining the face F_i with the origin of co-ordinates.

Proof. See [Fei97a]

Instead of using pyramids, we can use tetrahedra; this will allow us a simplification in the computations [Fei97a]. As it can be seen, the pyramids do not have to be disjoint. This will allow us to work with coverings of the solids, instead of disjoint partitions of them. The main advantage of this approach is that the covering can be obtained in a very simple way with an linear algorithm, keeping the initial representation of the solid (a vertex-edge-face graph). Another advantage is that it is not necessary to store the triangulation of the solid; it is only needed to know the edges of the solid and an arbitrary point, and therefore, there is no additional information to store.

Definition 5. Let *P* be a polygon, the *covering of the P,* denoted *as* C_p, is the set of triangles obtained by joining an arbitrary point of the plane of P with every edge of the polygon.

Analogy, let a solid be S, *the covering of S*, denoted as C_s, is the set of tetrahedral obtained by joining every triangle of the covering of every face of S with an arbitrary point.

Theorem 2. [Fei97b]. Let Q be a point, and S be a solid *(*a polygon in 2D*)*. Then *Q* is inside *S* if

$$\sum_i sign(Q, T_i) \cdot [T_i] = 1$$

where $T_i \in C_s$, $[T_i]$ is the signed volume (or signed area in 2D) of the simplex, and the function $sign(Q,T_i)\}$returns the signed volume (or area in 2D) of the simplex formed by point *Q* and simplex T_i (an edge in 2D or a face in 3D).

Corollary 1. Let Q be a point inside solid *S*. Then $\exists\ T_i \in C_s$, $[T_i]>0$, with *Q* included in T_i.

Proof. Trivially, it can be seen that, when the inclusion of a point in a solid is computed, we only use algebraic adding operations. So, at any moment it must be true that the sign of T_i is positive to obtain a positive result. Also, it is trivial to prove that the points of the solid included in negative T_i are also included in, at least, two positive T_j, because the result must be positive.

Lemma 2. [Seg01] Let *T*=(*ABC*) be a triangle, and *S*=(QQ') be a segment, with *Q* and Q' placed at opposite sides of plane Π defined by *T*, and ordered in such a way that the [QABC]>0. Then segment *S* cuts triangle *T* if

$\text{sign}(Q'AQB) \geq 0 \wedge \text{sign}(QCBQ) \geq 0 \wedge \text{sign}(Q'ACQ) \geq 0$

Proof. See [Seg01].

Corollary 2. [Seg98]} Let $P=P_1P_2...P_n$ be a polygon with covering $Cp=\{T_1T_2....T_n\}$, and S=(QQ') be a segment with Q and Q' placed at opposite sides of plane Π defined by P, in such a way that *P* is ordered counterclockwise with regard to Q. Then S intersects *P* if

$$\sum_i \text{inter}(S, T_{i)}) [T_i] = 1$$

$$\text{with inter}(S, T) = \begin{cases} 1 & \text{if S cuts inside T} \\ \frac{1}{2} & \text{if S cuts on an edge of T} \\ 0 & \text{in other case} \end{cases}$$

Proof. Let *M* be the intersection point between the ray and the plane Π. Then $M \in P$ if *M* belongs to *n* positive triangles and *n-1* negative ones (see corollary 1). So, $1 \cdot n + (-1) \cdot (n-1) = n - n + 1 = 1 > 0$

In the case of P is ordered clockwise with respect to *Q*, then the corollary can be expressed in an analogous way, but changing the sense of the comparison. The case of intersections with vertices or edges of the polygon can be detected by studying the intersection with the triangles of the covering.

3. REPRESENTING SOLIDS BY SIMPLICIAL COVERINGS.

Once we have presented the theoretical basis of the formal method of simplicial coverings, we are going to propose a simple data structure to represent solids in 3D. The data structure is based on another existing data structure for boundary representation, but adapting it to represent non-manifold solids. The idea is to store two arrays: an array of *n* 3D points representing the vertex of the solid; and a list of faces, where each one is represented as an array of integer values (the position of every vertex of the face in the list of vertex of the solid), representing the topology of the faces of the solid.. In the structure of the face are also stored the normal vector and the sign of the face (see def. 4).

In order to represent different loops of a face a false edge is introduced in the list of indices representing a face. In figure 1 this fact is shown. Polygon is composed by two loop (the first one delimited by vertices 0,1,2,3,4,5 and 6, and th second one delimited by vertices 7,8,9,10,11,12 and 13). The correct definition of the face will be 0,1,2,3,4,5,6,7,8,9,10,11,12,13,7,6,-1. As it can be seen, a false edge delimited by vertices 6 and 7 has been introduced twice, once in each sense. So, when the covering of the polygon is constructed, two triangles are obtained, each one with different sign.

Another solution to store different loops is to consider them as different faces, but orientating their normal vectors properly, so the external loops will be oriented in a way that their normal vectors point outside the solid, and the internal loops will be oriented so that their normal vectors point inside the solid (the sign of the internal loop is negative respect to the external one).

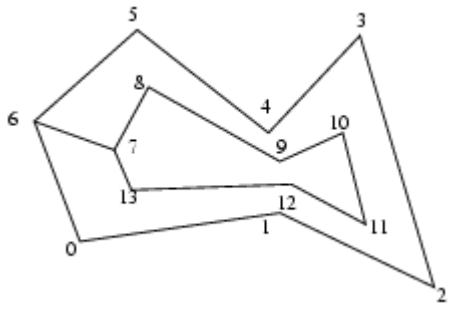

Figure 1. Introducing false edges on the representation of a face

4. BOOLEAN OPERATIONS.

By using the previous data structure, we can use exactly the definition of the Boolean operations to implement them as follows:

Complementary.

In order to implement the unary complementary operation is enough to invert the list of indices of the solid. So, the normal vector of each face changes its orientation (but not its direction).

Union.

The union operation can be implemented by joining the lists of vertices from both solids, and doing the same with the lists of indices from them. The resultant list of indices must be arranged in a way that the indices of the vertices corresponding to the second solid must be increased in *n*, being *n* the number of vertices of the first solid.

Difference.

The difference operation is defined as the set of points that belong to *A* and do not belong to *B*. So, in order to implement this operation it is enough to join the lists of vertices from both solids. The list of indices of the resultant solids is built by joining the lists of faces from the solids, but inverting the sign of the faces of the second solid.

Intersection.

The intersection is defined as the set of points belonging to *A*, that also belong to *B*. So, the implementation of this operation is similar to the union operation, but it will be necessary to take into account the condition $\wedge$ when the inclusion of points

in $A \cup B$ is studied. In order to create the data structure asociated to $A \cap B$, we use the De Morgan's law, where

$$A \cap B = \overline{(\overline{A} \cup \overline{B})}$$

5. PROPERTIES OF SOLIDS.

Area and Volume.

The volume of the solid obtained by applying one of the previous operations is computed by adding the signed volume of all the tetrahedra obtained by covering the solid. To obtain these tetrahedra, an arbitrary point is chosen (for example, the origin of coordinates), and then, for each face of the solid, the face is covered by triangles. Again, it is necessary to choose an arbitrary point to make the covering of this face, but any vertex of it can be chosen.

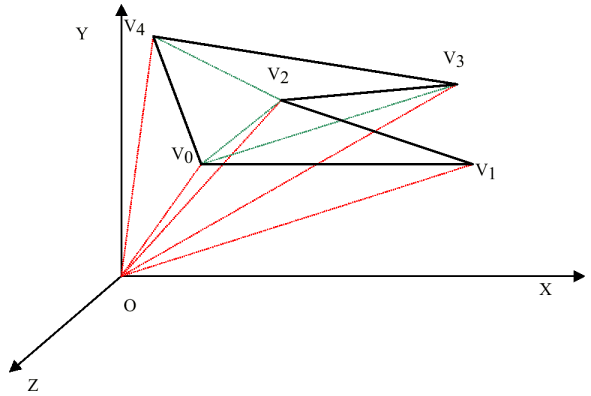

Figure 2. Covering the solid by tetrahedra: the process with a face

With this covering, it is easy to compute the area and the volume of the solid. These properties are directly derived from the formal specification of solids. So, the area can be computed as

$$Area(S) = \sum_{i=1}^{n} \sum_{j=1}^{k} Area(T_i^j)$$

Likewise, the volume of the solid is computed as

$$Vol(S) = \sum_{i=1}^{n} \sum_{j=1}^{k} Vol(OT_i^j)$$

It is important to note that those triangles with negative sign provide the final result with a negative area (or volume). So, The final result is valid. Equally, in the case of false edges, as these edges are included twice (once in each sense), then their areas are made void.

Point-in-polyhedron test.

Theorem 2 gives us a method to determine whether a point is inside a solid (or not). Applying the algorithm derived from this theorem (see [Fei97b]) to the representation explained in previous sections, a method to study the inclusion of points in solids is obtained. We must remember that the algorithm returns the position of the point with regard to the solid (INSIDE, OUTSIDE, FACE, EDGE or VERTEX).

In figure 3, two examples of boolean operations are shown. In case a), $A \cup B$ and $A \cap B$ results are shown. In this case, the interior of $A \cup B$ is defined by the points which verify that the inclusion test for them is greater than 0. So, the inclusion test for the points of the common part of *A* and *B* values at 2, and the remaining ones at 1. In order to study the intersection $A \cap B$ the inclusion test must take a value greater than 1, i.e. it must be valued at 2.

In figure 3.b, *A*-*B* case is shown. In this case, the order of the vertices corresponding to solid *B* has been changed, and therefore, the signs from the triangles of the covering of *B* have also been changed. So, the interior of *A*-*B* is defined by the points which verify that the inclusion test values exactly at 1. It is important to note that the result is not regularized because of the common boundary of *A* and *B*. This situation can be solved by multiplying by 2 the result of the inclusion test regarding loop *A* when it returns *EDGE*. In this case, in the common boundary, the values obtained will be 2 for loop *A*, and -1 for loop *B*, and as a consequence, the final result will be 1.

In figure 4, a study of point inclusion in solids obtained by boolean operations is shown. Two mechanical pieces have been used: in figure 4.a, the union of both pieces is shown; in figure 4.b, the difference between both pieces is shown. In figure 4.c a bombing of the figure obtained by the difference operation is shown. A collection of 20,000 random points has been generated in the bounding box of the resultant solid: points inside the solid appear in green and points outside the solid in red.

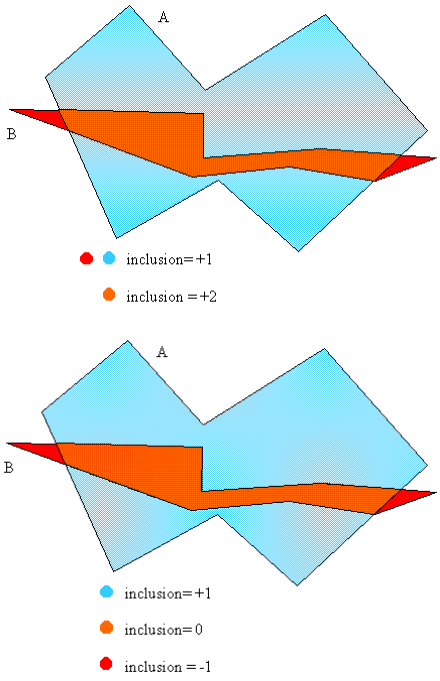

Figure 3. Value of inclusion in Boolean Operations

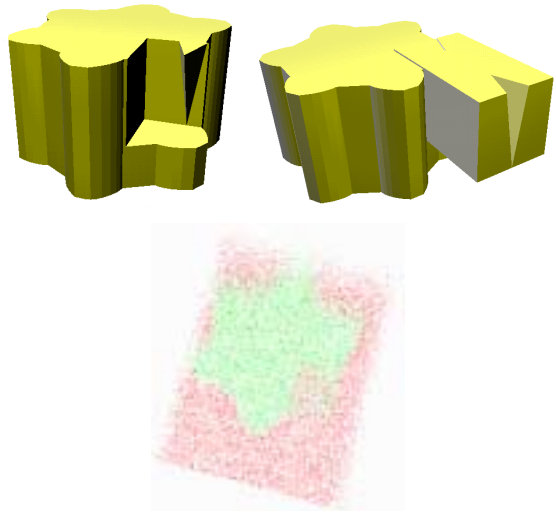

Figure 4. Inclusion in boolean operations

6. VISUALIZATION OF SOLIDS.

In order to obtain a visualization of the solids, several methods can be chosen, depending on the quality of the results. Nowadays, the most used method is using a Z-Buffer to render every polygon of the final result. Some modifications have been made to that method to render a CSG tree using a Z-Buffer, but all of them require that the faces of the used primitives must be convex [Ste00]. To apply this method to the data structure proposed in this paper it is necessary to make some changes to the basic algorithm. So, the negative polygons are not rasterized. There is a problem in this basic solution: if a negative face is overlapped with a positive one, then exists a hole in the solid, and then, the face visible through the pixels of the hole is a back face or the background. To solve this case, the overlapped faces of the solid must be joined with false edge, as it is shown in figure 1.

Ray-Casting.

The ray-casting method is based on following the trajectory of a ray passing through a pixel[Fol94]. For each pixel of the viewing window, a ray starting on the Projection Reference Point (PRP) and passing through the pixel is constructed. Then, for any solid of the scene, the intersection between the ray and the solid is computed; the colour of the pixel is the colour of the solid whose intersection with the ray is the nearest to the PRP. To test the intersection between a ray and arbitrary polygons in 3D, the algorithm proposed in [Seg98] can be used. The main problem of the adaptation of the basic algorithm to the scheme of representation proposed previously is that we can obtain some intersections with the solid in the same point, and maybe, these intersections are not valid because some of them are probably outside the solid. In figure 5 this problem is shown (green polygons are positive faces, and red polygons are negative ones). So, it is necessary to store all the intersections between the ray and the faces of the solid. For each intersection, the intersection point and the sign of the face are stored. Then, the colour of the pixel is the colour of the nearest intersection point where the addition of the signs from faces sharing this intersection point is positive. In this way, holes in faces are solved.

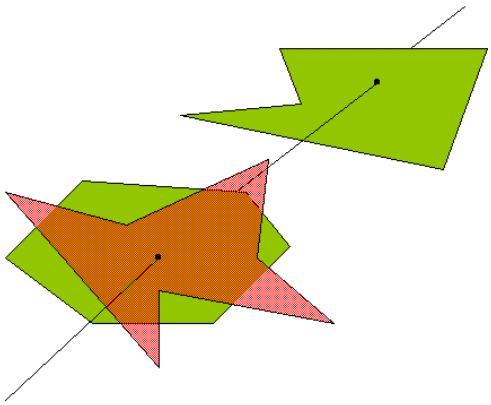

Figure 5. Ray-casting of polygons: sorting the intersections

The additional cost of the proposed modifications is $O(k \cdot log\ k)$ being k the maximum number of intersections found for a concrete ray.

7. CONVERSION TO ENUMERATIVE SCHEMES OF REPRESENTATION.

Voxelization of solids.

Voxelization of solids is a very useful tool to represent volumetric information. The problem consists on dividing the volume into regularly spaced pixels (in 2D) or voxels (in 3D) implemented typically as an array in which it is stored information such as color or opacity of each pixel or voxel occupied by the solid.

The main problem of the voxelization is to obtain the points of the space occupied by the solid. In 2D there are several algorithms to do it, although the most used is the scan-line algorithm [Fol94]. In 3D, there are also several algorithms, but a very interesting approach is the one proposed in [Fan00], which proposes a voxelization algorithm using the 2D hardware to obtain slices of the solid. However this algorithm requires computing the intersection between the solid and each slice, because all the solid is used as input on the algorithm. Our method uses only part of the information of the solid, and therefore, the computation of the intersections is simpler.

Theorem 2 gives us a simple method to voxelize solids. The solids are not voxelized directly on the 3D array, but in a special buffer called *presence buffer*, PB. This buffer stores one bit per pixel indicating the presence of the solid in the volume covered by this voxel. The operation needed for this buffer is only the negation of the value stored previously on every voxel.

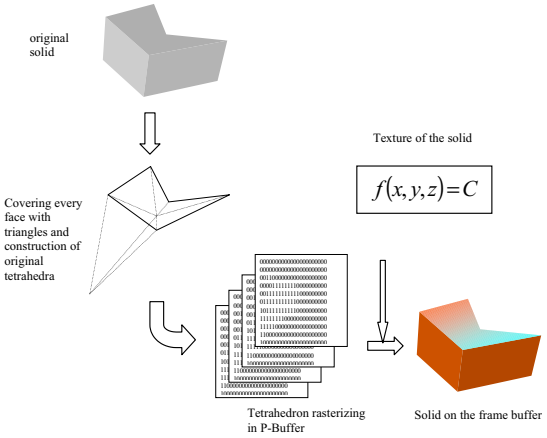

Figure 6. Rasterization of solids

Once the voxelization of the solid has been completed, the information stored on the presence buffer is transferred to the final array, applying its corresponding volumetric properties. The properties of every voxel depends on the properties of the solid. If we consider only homogeneous solids, then we can consider that the properties of every voxel are the same. Only two values are possible in the PB: 0 or 1. Only the voxels with presence value 1 are rendered to the array using the corresponding properties. The voxelization process is as follows (see figure 6):

- Rasterize every tetrahedra of the covering of the solid in the PB, changing the value of the positions covered by the tetrahedron.
- Transfer all positions with presence value being equal than 1 to the frame buffer, by applying a function such as: given a point of the solid, it returns the corresponding colour of the solid in that point. The definition of this function depends on every solid.

A complete description of the algorithm can be found on [Rue02].

Direct octree generation.

Traditional algorithms to obtain octrees require computation of plane-plane intersections. However, our conversion process is based on the point-in-polyhedron test previously detailed.

Basically, the algorithm consists on classifying each octant with regard to the solid, beginning at top level (figure 7.a) and recursively decomposing the universe cube until a fixed level is reached (figure 7.b). Octant classification is carried out by testing whether some points, eight vertices of each octant plus its centroid, are inside the solid or not.

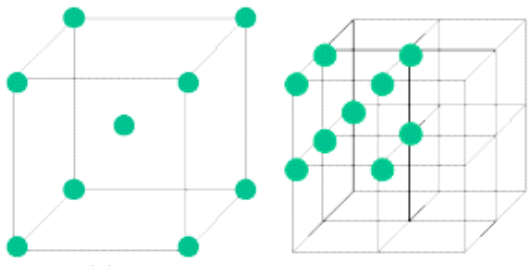

Figure 7. Octant classification

Two problems must be avoided: mistaken results due to earlier classification of octants in models with complex topology, and no necessary subdivision of octants completely inside or outside the solid. To achieve this, the conversion algorithm can stop in an intermediate level if thresholds previously established for the classification of a octant are reached. This

optimization allows us to obtain an agreement between an accurate result and the time consumed for classification of the entire octree. For a detailed explanation of this conversion algorithm see [Rui02].

Next section shows several octrees obtained through this method, and the visual interpretation of thresholds for the octant classification.

8. TRIANGULAR MESHES.

Triangle meshes have become a typical representation for modeling. But using such a representation for other purposes than rendering also requires the implementation of robust and efficient geometric algorithms.

We can easily apply our methods to triangular meshes. The fundamental differences with regard to general polyhedra are: the new algorithms are simpler and more efficient; the application to meshes implies a simplification because the used operations are basic. The methods can be applied to meshes containing the minimum topological information, that is, a labelled list of vertices indicating their coordinates and a list of triangles containing the references to their vertices. Obviously, when more information is available, as triangles sharing edges or vertices, it could be used to accelerate the proposed algorithms. The algorithms are also valid for non-manifold models defined by triangle meshes and for models with holes; moreover, they can be applied to multi-resolution models.

Figures 7.a and 7.b show two solids bounded by triangular meshes (Buddha and chess queen). Figure 7.c shows Buddha ⋃ queen, and figures 8.a, 8.b and 8.c show Buddha ⋂queen, queen - Buddha and Buddha - queen, respectively. Figures 9.a - 9.d show the corresponding octrees for solids in figures 8.a - 8.c. These figures have been generated with ESC-MOD SYSTEM [Rui02], a solid modeling system that can deal with both free-form solids and solids bounded by complex planar triangular meshes. Figure 8.c shows the main problem in our approach: very thin areas of the solid could not be detected due to no calculation of plane-plane intersections. This problem is solved by reducing the threshold that classifies an octant as black, that is, an octant is considered black if only a few number of test points are inside the solid. The resulting octree is shown in figure 9.d.

9. CONCLUSIONS.

We have presented a representation scheme for solids with planar faces (manifold or non-manifold, with or without holes, convex or concave ones). The method is based on covering the solid by simplices, and providing a data structure smaller than other data-structures valid for non-manifold solids.

With this data structure, a method to compute the Boolean operations has been presented. The algorithms are fast and robust, but the solid obtained is not minimal because there is no evaluation of the boundary of the solid.

a) b) c)

Figure 7. Happy Buddha, Chess Queen and Happy Buddha ⋃ Chess Queen.

a) b) c)

Figure 8. Happy Buddha ⋂ Queen, Queen – Happy Buddha, Happy Buddha - Queen.

By using the appropriate algorithms, some typical problems of solid modeling have also been solved: area, volume, inclusion, visualization or octree generation of the solid are obtained in an easy and efficient way.

The proposed model can be useful as an intermediate representation for solids in applications in which the main objective is to know the shape of the solid and not its final boundary. The method proposed is also valid for studying the classification of points in solids. Another field of application of the model could be the conversion of the solids from one scheme of representation to another one. For example, we can use it as an intermediate representation from CSG to B-rep scheme.

10. ACKNOWLEDGEMENTS.

This work has been partially granted by the Ministry of Science and Technology of Spain and the European Union by means of the ERDF funds, under the research project TIC2001-2099-C03-03

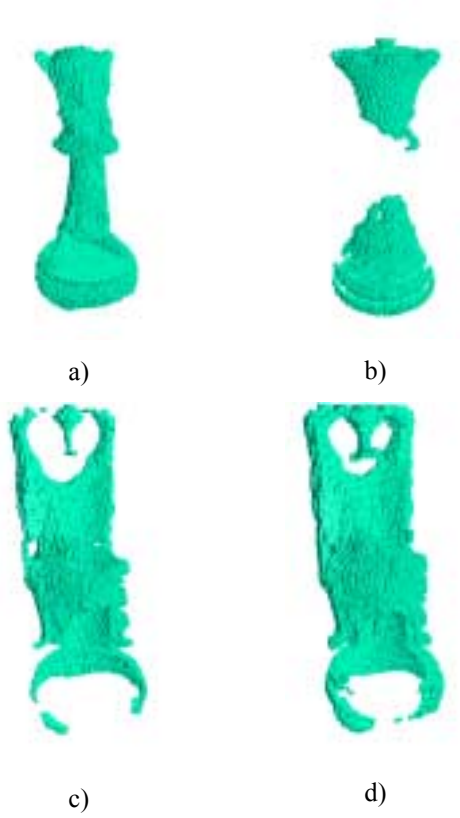

Figure 9. Octree for fig. 8.a, (b) octree for fig. 8.b, and (c,d) octrees for fig. 8.c.

11. REFERENCES

[Bron90] Bronsvoort, W. Direct display algorithms for solid modeling, Delft University Press, 1990.

[Chaz] B. Chazelle e.a., Application Challenges to Computational Geometry, CG Impact Task Force Reports, Technical Report TR-521-96, Princeton University, April 1996.

[Chaz92] Chazelle. B. An Optimal Algorithm for Intersecting Three-Dimensional Convex Polyhedra, SIAM Journal on Computing. Vol. 21(4), 671--696,1992.

[Fan00] Fang, S., Chen, H., Hardware Accelerated voxelization. Computer & Graphics, Vol. 24(3), 433--442, 2000.

[Fei97a] Feito, F., Torres, J.C., Boundary Representation of Polyhedral Heterogeneous in the context of a Graphic Object Algebra. The Visual Computer}, Vol. 13, 64--77, 1997.

[Fei97b] Feito, F., Torres, J.C., Inclusion test in general polyhedra. Computer & Graphics, Vol. 21(1), 23--30, 1997.

[Fol94] Foley , J. e.a. Introduction to Computer Graphics, Addison Wesley, 1994.

[Gardan96] Gardan, Y., Perrin, E., An Algorithm reducing 3D boolean operations to a 2D problem: concepts and results. Computer-Aided Design, Vol 28 (4), 277--287, 1996.

[Lee96] Lee, D.T., Computational Geometry, ACM Computing Surveys}, Vol. 28(1), 27--31, 1996.

[Mant86] Mantyla, M. An Introduction to solid modeling, Computer Science Press, 1986.

[Prep88] Preparata, F.P., Shamos. M.I., Computational Geometry: An Introduction, Springer-Verlag, New York, 1988.

[Rue02] Rueda, A.J., Segura, R.J., Ruiz, J., Feito, F.R., An Unified Approach for 2D and 3D Rasterization. Proc. of 1st Ibero-American Symposium in Computer Graphics, Guimaraes, (Portugal), 2002.

[Rui02] Ruiz de Miras, J., Feito, F.R. Direct and Robust Voxelization and Polygonization of Free-Form CSG Solids". International Symposium on 3D Data Processing, Visualization and Transmission , Padua (Italy). 2002.

[Seg01] Segura, R.J., Feito, F., Algorithms to test ray-triangle intersection. Comparative study, Journal of WSCG, Vol. 9(3), 2001.

[Seg98] Segura, R.J., Feito, F., An Algorithm for Determining Intersection Segment-Polygon in 3D. Computer & Graphics, Vol. 22(5), 1998.

[Ste00] Stewart, N., Leach, G. , John, S., A Z-Buffer CSG rendering algorithm for convex objects, Journal of the WSCG, Vol.8 (1), 2000.

[Sugi94] Sugihara, K.A., A Robust and Consistent Algorithm for Intersecting Convex Polyhedra. Computer Graphics Forum, Vol 13, (3), C.45--C.54, 1994.

[Tor93] Torres, J.C., Clares. B., Graphics Object: A Mathematical Abstract Model for Computer Graphics, Computer Graphics Forum, Vol. 12(5):311-328, 1993.