Lecture 13: Ray tracing and Constructive Solid Geometry

Constructive Solid Geometry (CSG)

So far we have only dealt with simple geometric solid primitives. In contrast to surface models which are represented by triangles, quads or polygons, these solid models can only be rendered using ray tracing. However, in most applications these primitives are not sufficient for more complex shape modeling as required in computer aided design (CAD) or computer aided manufacturing (CAM). Constructive Solid Geometry (CSG) provides a way in which we can combine the basic geometric primitives to form more complex shapes.

Constructive solid models can consists of primitive shapes such as spheres, cylinders, cones, pyramids, cubes or blocks. Constructive solid models cannot consist of half-spaces such as points, lines or planes. CSG combines solid objects by using three (four) different boolean operations which include the intersection (∩), the union (+), the minus (–) and the complement operation. In theory, the minus operation can be replaced by a complement and intersection operation, however in practice the minus operation is often more intuitive as it corresponds to removing a solid volume. An example of the CSG operations is shown in Figure 1.

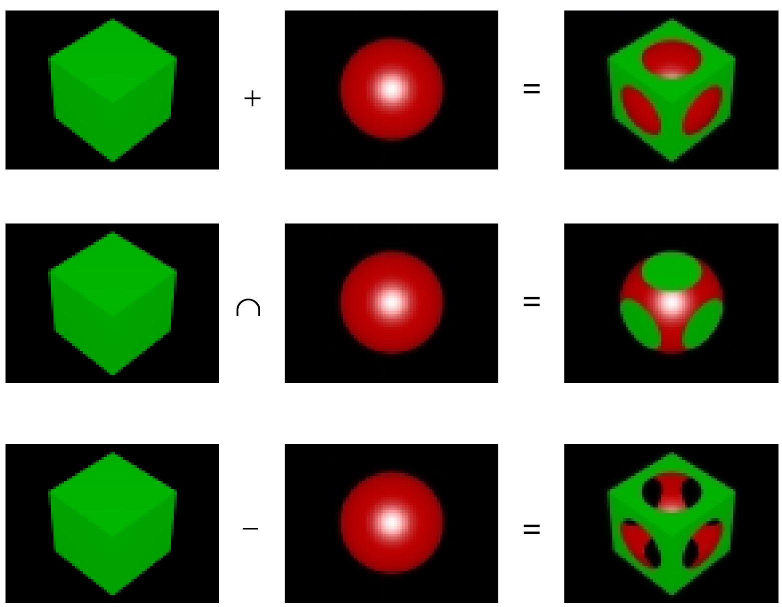


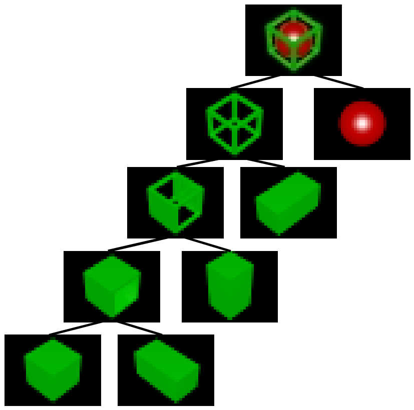
Figure 1: CSG operations

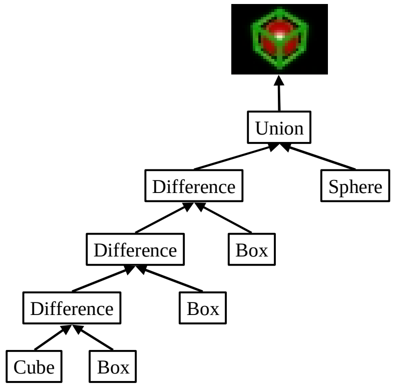
An important aspect of CSG operations is that the operations are not commutative, i.e. the object produced by the operation A – B is normally different from the object produced by B – A. In addition, CSG operations are normally not unique which means that there are a number of different combinations of CSG operations which can be used to create the same object.

CSG has a large number of applications outside computer graphics, in particular in computer aided manufacturing. We will focus only on their applications in computer graphics.

CSG trees

The CSG procedure can be represented as a tree-like structure in which the root of the tree defines an object. The terminal nodes of the tree correspond to geometric primitives while the non-terminal nodes correspond to combination operations (intersection, union and minus) operating on their subtrees.





Ray tracing CSG trees

In a CSG system objects can be represented as a tree structure in which the root of the tree defines the object. The terminal nodes of the tree correspond to the geometric primitives used (i.e. spheres, cylinders and cones). The non-terminal nodes are the combination operations (union, difference and intersection) operating on their subtrees. To ray trace a CSG tree, the tree must be traversed in a depth-first manner starting at the terminal nodes. At the terminal nodes the intersection of the ray with each primitive. The list of ray segments that pass through the solid object must be passed back up the tree. Each list of ray segments will either contain an odd number of intersection points, an even number of intersection points, or an empty list of intersection points. An odd number of intersection points indicates that the viewpoint is inside the solid object while an even number of intersection points indicates that the viewpoint is outside the solid object.

To further improve the efficiency of the ray tracing, CSG trees can be pruned during the intersection calculations. For example, if a subtree of an intersection operation returns an empty intersection list, the other subtree need not be processed. Similarly, if the left subtree of a minus operation returns an empty intersection list, the right subtree need not to be processed. This can significantly speed up the process of ray tracing a CSG tree.

Interior, Exterior and Closure

A problem of CSG representations is caused by the fact that the combination of two solid objects may not necessarily produce another solid object. For example, the intersection of two cubes which share a face yields a plane. To avoid these problems we need to regularize the CSG operations. To achieve this we need to introduce the concepts of interior, exterior and closure.

A point **p** is an interior point of a solid **s** if there exists a radius *r* such that the open ball with center **p** and radius *r* is contained within the solid **s**. The set of all interior points of solid **s** is the interior of **s**, written as **int**(**s**). Based on this definition, the interior of an open ball is the open ball itself. A point **q** is an exterior point of a solid **s** if there exists a radius *r* such that the open ball with center **q** and radius *r* is not contained in **s**. The set of all exterior points of solid **s** is the exterior of solid **s**,written as **ext**(**s**). All points that are neither in the interior nor in the exterior of a solid **s** are the boundary of solid **s**. The boundary of **s** is written as **b**(**s**). Therefore, the union of interior, exterior and boundary of a solid is the whole space. The closure of a solid s is defined to be the union of **s**'s interior and boundary, written as **closure**(**s**). Or, equivalently, the closure of solid **s** comprises all points that are not in the exterior of **s**.

To eliminate these lower dimensional branches, the three set operations are regularized: Compute the result as usual and lower dimensional components may be generated. Compute the interior of the result. This step would remove all lower dimensional components. The result is a solid without its boundary. Compute the closure of the result obtained in the above step. This would add the boundary back.

Let +, ∩ and – be the regularized set union, intersection and difference operators. Let A and B be two solids. Then, A + B, A ∩ B and A – B can be defined mathematically based on the above description:

- A + B = **closure**(**int**(the set union of A and B)
- A ∩ B = **closure**(**int**(the set intersection of A and B)
- A – B = **closure**(**int**(the set difference of A and B)