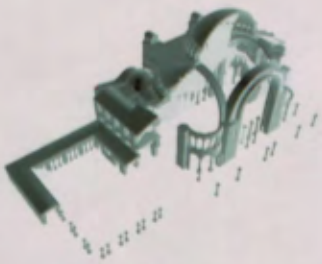
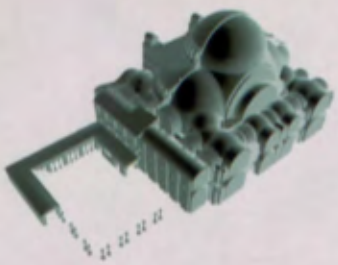
Second Edition

Digital Design Media

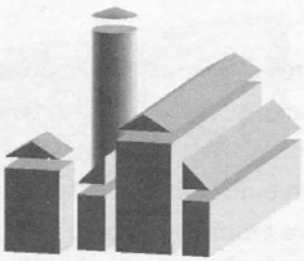




William J. Mitchell
Malcolm McCullough

11

ASSEMBLIES OF SOLIDS



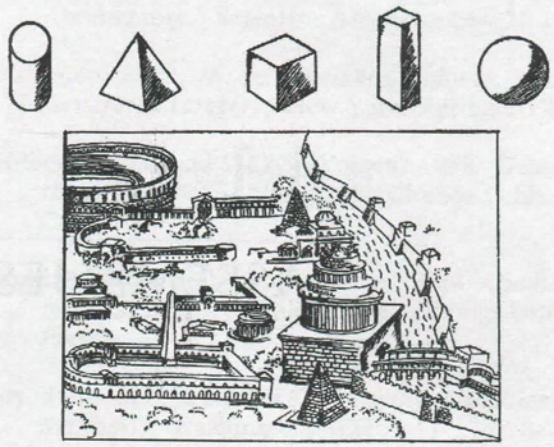
11.1
Solid building blocks

Sometimes designers want to conceive of three-dimensional compositions not abstractly in terms of lines in space, nor more visually as collections of surfaces in light, but spatially, as arrangements of volumes—both solids and enclosed voids (figure 11.1). Indeed, as Le Corbusier pointed out in *Vers une architecture*, this characterizes some architectural styles (figure 11.2). "Egyptian, Greek or Roman architecture," he wrote, "is an architecture of prisms, cubes and cylinders, pyramids or spheres: the Pyramids, the Temple of Luxor, the Parthenon, the Coliseum, Hadrian's Villa." By shaping and arranging blocks of wood or polystyrene an architect can compose directly in volumes, but this is slow and cumbersome. An increasingly attractive alternative is to employ solid-modeling software that provides prisms, cubes, cylinders, spheres, and so on as geometric primitives, together with tools for inserting, deleting, transforming, and combining these.

The displays produced by solid-modeling systems look much like the displays produced by wireframe- or surface-modeling systems (depending upon the way that solids are rendered), but the underlying geometric databases are very different. As a result, there are powerful geometry-editing operations not available in wireframe or surface systems, there are additional data extraction and analysis possibilities, and the process of design exploration with a solid modeler tends to evolve in different ways.

A DIFERENÇA CENTRA-SE NA FORMA COMO A INFORMAÇÃO É ARMAZENADA NA BASE DE DADOS

ESSA ESTRUTURA PERMITE DIFERENTES TIPOS DE EDIÇÃO



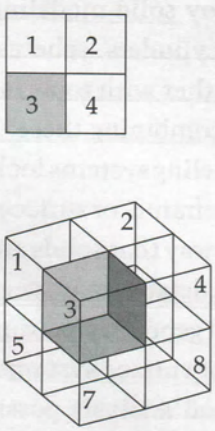
11.2
Volumetric composition according to Le Corbusier

Voxel Representation

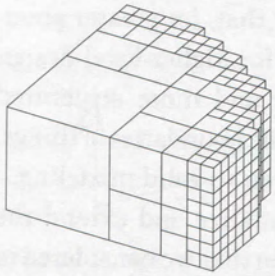
Just as sounds can be represented by one-dimensional arrays of data points and images can be represented as two-dimensional arrays of data points, so compositions of solids can be represented as three-dimensional arrays of data points. For this purpose we employ a cuboid subdivided into cubic voxels (volumetric elements) rather than a rectangle subdivided into square pixels (figure 11.3). We can then represent the forms of solid objects, using one bit of information per voxel, by coding a voxel outside the solid as zero and a voxel inside the solid as one.

As with sampling sounds and sampling images, we need a sufficiently high density of samples to avoid unacceptable aliasing effects. But high sample densities are particularly hard to achieve in this case: whereas halving the interval between sound samples doubles the total number of data points and halving the distance between image samples quadruples the number of data points, halving the distance between solid samples produces an eightfold increase in the number of data points.

However, voxel representations can usually be compressed effectively through use of a technique known as octree encoding (figure 11.4). This is a three-dimensional version of the quadtree technique for compressing bitmapped images, which was discussed in chapter 6. An octree is constructed by first subdividing the voxel array



11.3
Pixels and voxels

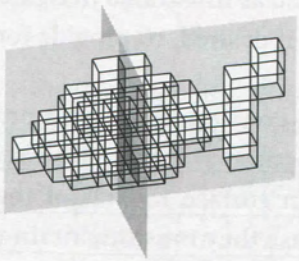


11.4
Octree representation of a quarter cylinder

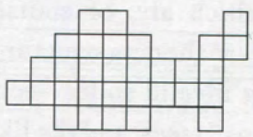
into octants, then further subdividing any nonuniform octants, and so on until there is no need for further subdivision or the level of individual voxels is reached. Each terminal node of the octree can then be labeled with the value of the corresponding volume.

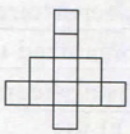
We can generate output from voxel representations in several ways (figure 11.5). Horizontal or vertical slices through the voxel array are one-bit bitmaps that can be displayed as sections. We can also produce hidden-line and shaded images by interpreting the faces of voxels as opaque square surfaces. Raytracing techniques can be adapted to render solids as transparent volumes—a particularly popular technique in medical-imaging applications. And we can employ special devices to produce actual three-dimensional solids. A stereolithography device, for example, operates on a tank of liquid to produce laser-induced solidification at locations corresponding to occupied voxels.

Use of one bit per voxel suffices to distinguish between solid and void (which is enough for many design purposes), but we can introduce more distinctions if we wish. Use of two bits per voxel, for instance, provides for distinction between four different occupancy conditions—different materials, say, or different densities of material. This is useful when we need to represent the internal structures of solids, as geologists do when they represent geological structures, as oceanographers and atmospheric scientists do in their domains, and as medical imagers do when they investigate the internal structure of the human body. As sensing and sampling techniques develop, and as the growing availability of inexpensive memory and processing power increases the feasibility of handling large, high-resolution voxel representations, processing of voxel data for scientific visualization purposes is becoming an increasingly important field.

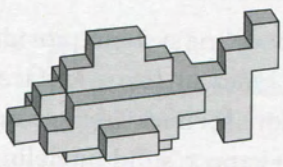


Wireframe





Sections



Shaded surfaces

11.5
Output from a voxel model

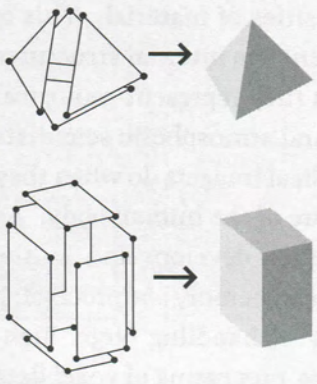
Boundary Representation

For designers' purposes, however, voxel representations suffer from the same sorts of limitations as the bitmapped images that we considered in chapter 6: they are low-level, unstructured, imprecise, and inefficient in use of available

computational resources. We saw that, for greater precision and economy and to provide for higher-level design operations, we could use sparser and more structured representations in terms of lines—the boundaries of things. An analogous approach can be taken to solid modeling.

The basic idea here is to generalize and extend the techniques of surface representation that we considered in chapter 10. Connected pairs of zero-dimensional points (vertices) define finite one-dimensional lines, connected sequences of three or more one-dimensional lines can be used to define two-dimensional closed polygons, and connected assemblies of four or more closed polygons can be used to define closed polyhedral solids (figure 11.6). Thus data structures for boundary representation of polyhedral solids can be structured as illustrated in figure 11.7. These can be generalized, if desired, to provide for curved as well as planar faces.

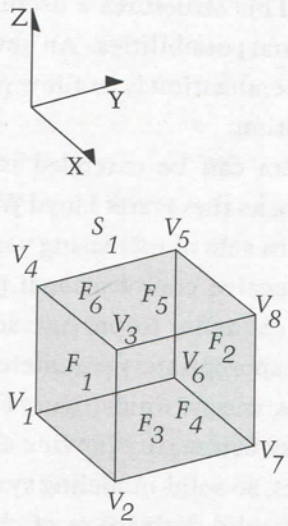
These sorts of data structures consume more memory and are more cumbersome to manipulate than data structures for wireframe or even surface models of the same forms. This is partly because they must maintain a richer network of associations between geometric elements and partly because the associated operations for transforming and combining solids (which are, of course, implemented as operations on values in the data structure) must be prevented from producing invalid solids—self-intersecting ones, ones with "missing" faces, and the like (figure 11.8). The data structures and associated repertoires of operations of solid modelers are usually organized to maintain the topological properties of closed polyhedral solids as specified by Euler's theorem (figure 11.9).



11.6
Surfaces form "watertight" boundaries of closed solids

Vocabularies of Solid Building Blocks

Drafting systems and wireframe-modeling systems provide operations for inserting various types of lines; surface-modeling systems provide operations for inserting various types of surfaces; and, as we might expect, solid-modeling systems provide operations for inserting various types of closed solids. A very simple system might, for example, provide operations for inserting, selecting, and deleting

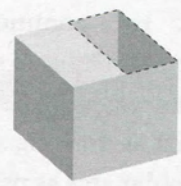


V_1 (0,0,0) E_1 (V_1,V_2)
V_2 (1,0,0) E_2 (V_2,V_3)
V_3 (1,0,1) E_3 (V_3,V_4)
V_4 (0,0,1) E_4 (V_4,V_1)
V_5 (0,1,1) E_5 (V_5,V_6)
V_6 (0,1,0) E_6 (V_6,V_7)
V_7 (1,1,0) E_7 (V_7,V_8)
V_8 (1,1,1) E_8 (V_8,V_5)
E_9 (V_1,V_6)
E_{10}(V_2,V_7)
E_{11}(V_3,V_8)
E_{12}(V_4,V_5)

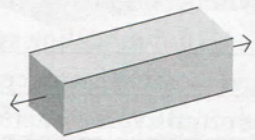
F_1 (E_1, E_2, E_3, E_4)
F_2 (E_5, E_6, E_7, E_8)
F_3 (E_1, E_{10}, E_6, E_9)
F_4 (E_2, E_{11}, E_7, E_{10})
F_5 (E_3, E_{12}, E_8, E_{11})
F_6 (E_4, E_9, E_5, E_{12})

S_1 (F_1, F_2, F_3, F_4, F_5, F_6)

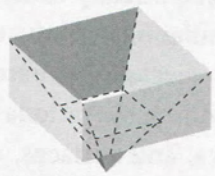
11.7
Boundary representation of a closed polyhedral solid



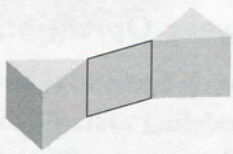
Not closed



Infinite

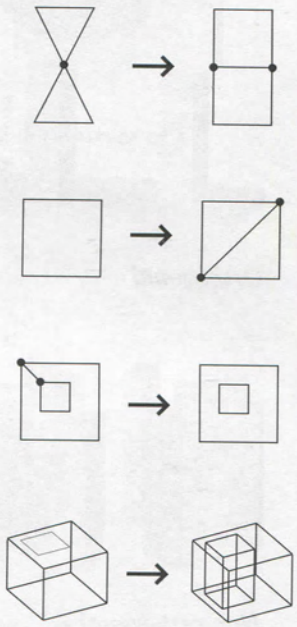


Self-intersecting



Nonmanifold

11.8
Invalid solids

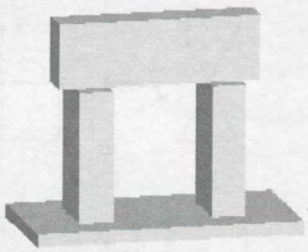


11.9
Euler operators for manipulating vertices, edges, and faces are closed in the polyhedral solids

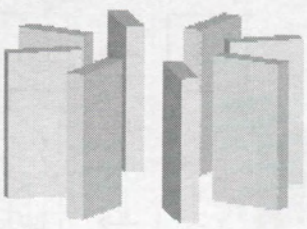
rectangular boxes, of specified dimensions at specified locations and oriented parallel to the axes of the coordinate system (figure 11.10). This structures a useful but very restricted domain of formal possibilities. An obvious and simply implemented generalization is to allow placement of boxes in any orientation.

Vocabularies of polyhedra can be extended indefinitely—just as the Froebel blocks that Frank Lloyd Wright played with as a child came in sets of increasing variety, opening up increasingly extensive compositional possibilities. It is common, for example, to provide simple "pitched roof" forms such as appropriately parameterized triangular wedges, gables, hips, and pyramids (figure 11.11). And, just as drafting systems customarily provide circles and circular arcs as primitives, so solid-modeling systems frequently provide the basic solid derivatives of circles: cylinders, spheres, cones, and doughnuts (figure 11.12).

Solid-modeling systems that rely entirely on creating and locating instances of vocabulary elements are known as primitive instancing systems. They are very effective in contexts where the kit of parts that a designer deploys is, in fact, strictly limited (as is sometimes the case in the manufacturing industry). In most design contexts, though, it is necessary to extend the repertoire of possibilities by providing operations for constructing solids from points, edges, and surfaces, and for combining simple solids to make more complex solids.



Orthogonal

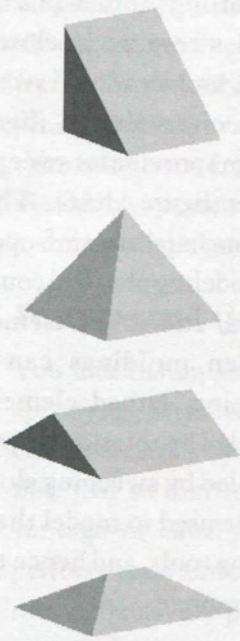


Non-orthogonal

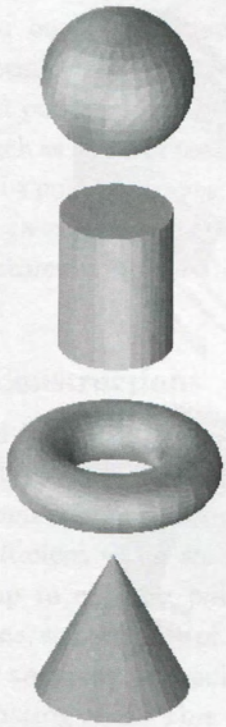
11.10
Assemblies of boxes

Sweep Operations

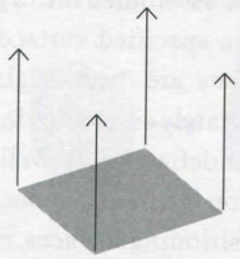
Sweep operations are very commonly employed in solid-modeling systems to create new solids. When a closed polygon is translated along an axis, its zero-dimensional vertices sweep out one-dimensional edge lines, its one-dimensional edges sweep out two-dimensional facets, and its two-dimensional surface sweeps out a three-dimensional volume (figure 11.13). Solids can also be constructed by means of hierarchical sweep operations: a point might be swept to generate a straight line, that line might be swept to generate a square, and that square might be swept to generate a cube (figure 11.14).

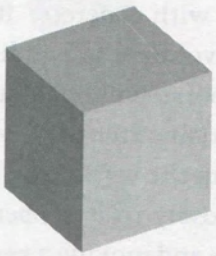


11.11
A vocabulary of simple roof forms

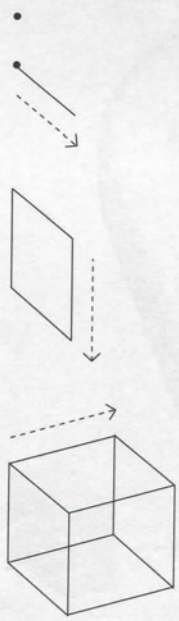


11.12
Solid derivatives of a circle





11.13
A swept polygon



11.14
Hierarchy of swept primitives: point, line, face, cube

11.15
Solids of translation and revolution produced from the same profile

11.16
A solid swept by a polygon moving along an arbitrary path

The usual approach to sweep operations is to provide for drafting profiles in a construction plane, then translational sweeping of closed shapes to produce prismatic forms, and rotational sweeping of open profiles to produce solids of revolution (figure 11.15). More sophisticated systems provide for sweeping closed shapes along arbitrary curves (figure 11.16). These operations have their counterparts in fabrication operations, so they often serve well for modeling physical construction components: extruded, planed, and rolled elements such as steel sections and wooden moldings can be modeled by translational sweeping; lathed elements and turned pottery can be modeled by rotational sweeping; and bent elements can be modeled by sweeping along curves. Sweep operations can also be used to model the envelopes swept out by moving cutting tools, and hence the volumes of material that they will remove.

Skinning and Tweaking Operations

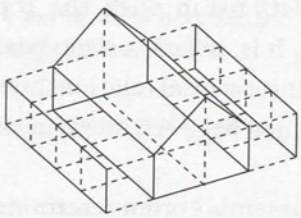
Closed solids can also be constructed in surface-by-surface fashion, as illustrated in figure 11.17. This operation is known as skinning. The user must select the surfaces that are to be assembled into a solid. The software then checks that the specified surfaces do indeed enclose a volume (that they are "watertight"), and if so the surfaces are appropriately associated in the data structure. This method of solid definition is well suited to describing cast construction elements: indeed, the operation of constructing and positioning surfaces, checking for watertightness, and converting the hollow shell into a solid is very closely analogous to building and positioning form work and filling it with concrete. It is also good for describing the exterior volumes of buildings (since these are bounded by waterproof assemblies of surfaces) and the interior volumes of rooms (since these are often bounded by surfaces closed to keep in the warmth).

A closely related operation is known as tweaking—selecting and moving a vertex, control point, edge, or face to adjust the shape of a solid (figure 11.18). This operation must be controlled very carefully (either by the user or by the software), since it can produce inadvertent conversion

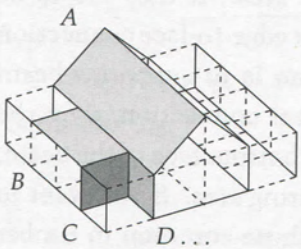
of planar faces into nonplanar ones and conversion of closed solids into self-intersecting objects (figure 11.19). Tweaking vertices and control points is particularly effective for describing shapes such as those of tents, which are controlled at various points by poles and ropes. Objects that are formed by bending, twisting, and other such distortion operations can sometimes be modeled by taking a simple shape and tweaking it.

Features and Geometric Constructions

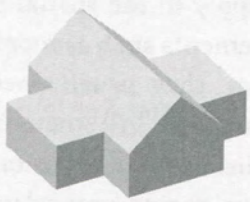
You can instantiate solids and locate them in space by specifying parameter and coordinate values, much as you can insert lines by specifying their end-point coordinates. But, just as drafting is more efficient when we take advantage of capabilities to snap to existing points and perform geometric constructions, so assembly of solids is more effectively performed by snapping new solids into specified relationships with existing ones. This requires definition of the features of the new solid that we want to relate to the environment (other solids), definition of the features of other solids to which they need to be related, and definition of the nature of the relationship.



Wireframe

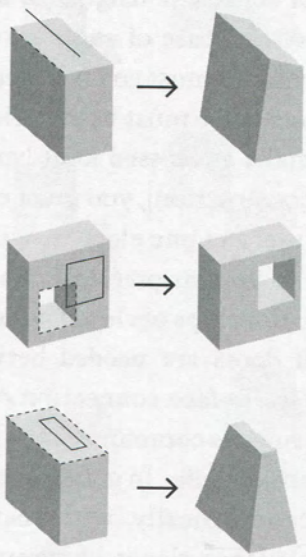


Surfaces

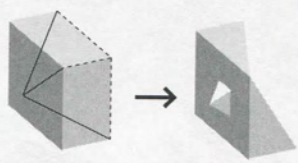


Completed boundary evaluated as a solid

11.17
Skinning



11.18
Tweaking



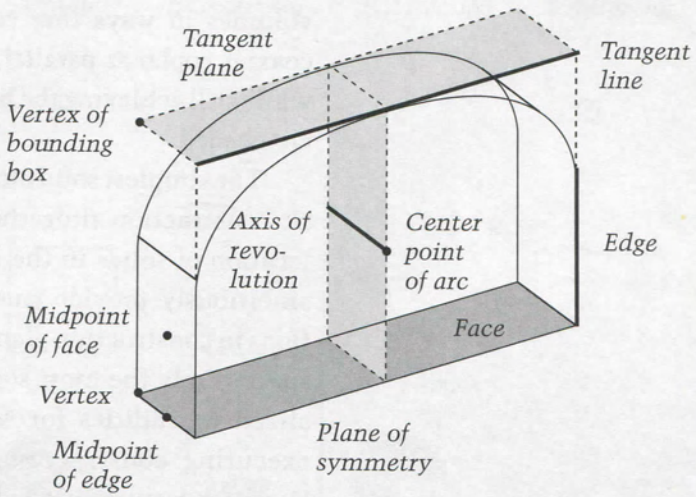
11.19
Invalid object produced by tweaking

The complexity (and interest) of this issue is that solids have a great many definable features (figure 11.20), and the ways to relate solids in terms of these features can be extraordinarily varied. Potentially significant points associated with a solid include, for example, end points and midpoints of edges, center points of arcs, and centers of symmetry. Potentially significant lines include edges of faces, axes of symmetry, and surface normals. Potentially significant surfaces include not only faces, but also tangent planes to curves such as cylinders and spheres, and planes of reflective symmetry. Points may be snapped together; lines may be snapped into collinear, parallel, or perpendicular relationships; the bottom surface of one solid may be snapped into a coplanar relationship with the top surface of another one; and so on. It is, in fact, a nontrivial exercise to enumerate all the definite spatial relationships of design interest that can be formed between one type of solid and another (figure 11.21).

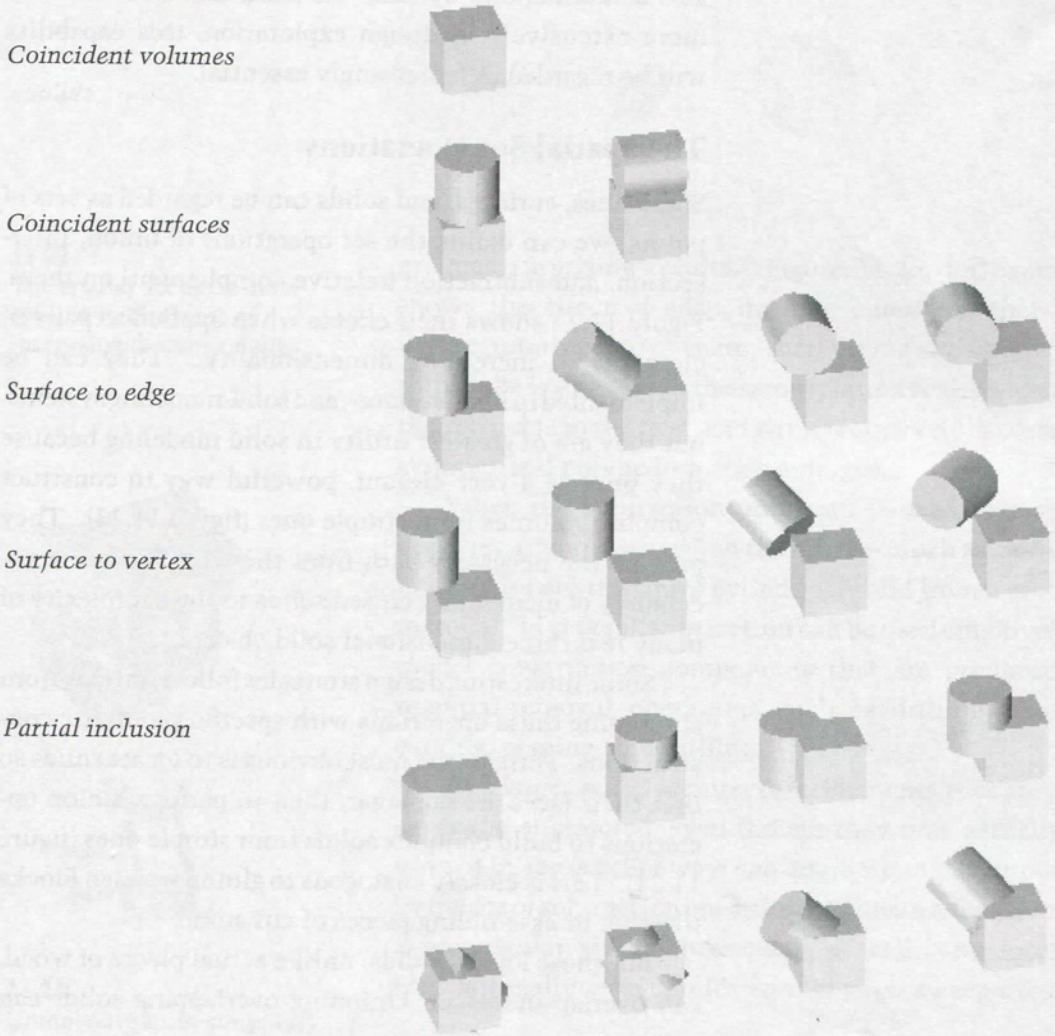
The realities of construction assembly often determine the potential spatial relationships of solid elements. If they are to be glued together, for example, they need face-to-face connection of sufficient area. If they are to be welded, they need edge-to-edge or edge-to-face connection of sufficient length. If a column is to support a beam without use of some sort of shear connection, then the column must go underneath the bottom face of the beam, and there must be sufficient bearing area. If you want to make a recessed joint (such as those common in timber construction), you must cut out a housing so that you can intersect one element with the other.

Similar practical constraints apply to the spatial relationships of closed volumetric elements such as rooms. If doors are needed between them, they usually need face-to-face connection of sufficient area. Alternatively (and less commonly), one might fit inside the other to form an aedicule. In classical composition, rooms were related concentrically, with coaxial axes of symmetry, or with coplanar planes of symmetry. But Frank Lloyd Wright overlapped interior volumes in ways that (in his later work) carefully avoided these classical relationships. And many of Frank Gehry's compositions juxtapose

11.20
Some of the features of a simple solid



11.21
Some of the possible spatial relationships between a cube and a cylinder



volumes in ways that conspicuously avoid concentric, coaxial, coplanar, parallel, and perpendicular relationships while still achieving the basic functional connections that are needed.

TIPOS E SOFISTICAÇÃO DE MODELADORES DE SÓLIDOS

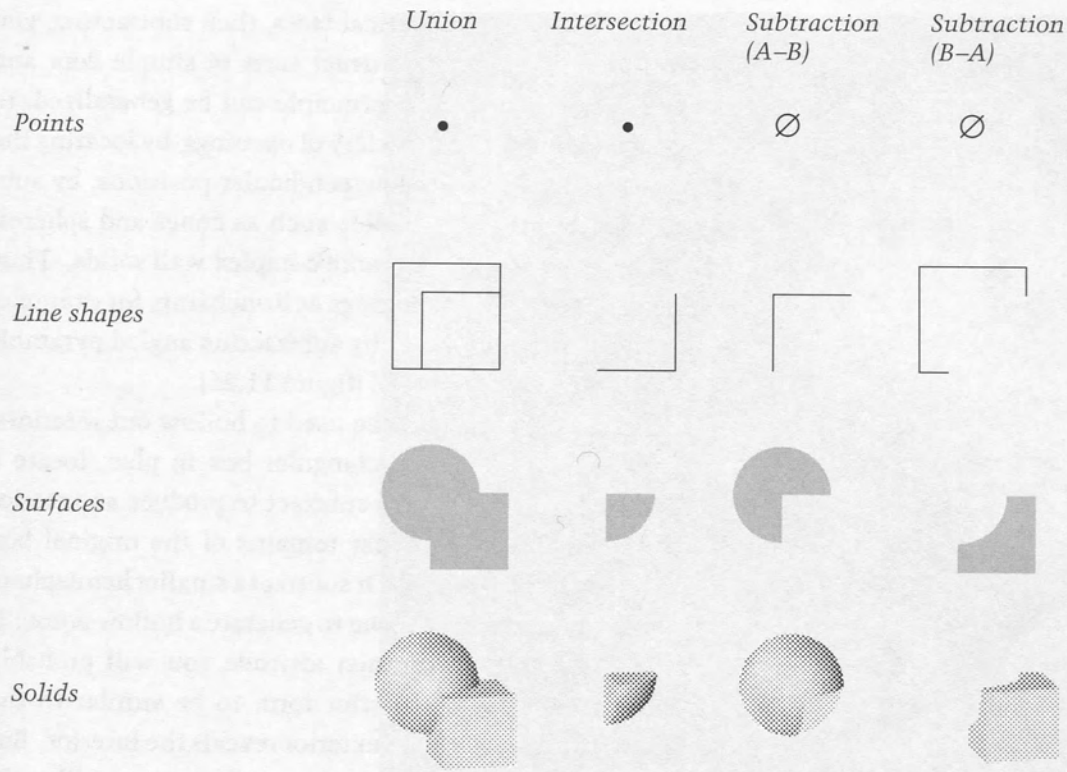
The simplest solid modelers avoid the issue of geometric construction altogether and merely provide for direct location of solids in the coordinate system. Some more ambitiously provide snapping and geometric constructions in construction planes, but not in three-dimensional space. Only the most sophisticated so far provide generalized capabilities for selecting features of solids and executing constructions in terms of these features. However, feature-based editing capabilities allow a designer to specify not only a vocabulary within which to work, but also a rudimentary syntax. As solid modelers are used more extensively in design exploration, this capability will be regarded as increasingly essential.

The Spatial Set Operations

Since lines, surfaces, and solids can be regarded as sets of points, we can define the set operations of union, intersection, and subtraction (relative complement) on them. Figure 11.22 shows their effects when applied to pairs of elements of increasing dimensionality. They can be implemented in line-, surface-, and solid-modeling systems, but they are of greatest utility in solid modeling because they provide a very elegant, powerful way to construct complex volumes from simple ones (figure 11.23). They provide the necessary path from the simplicity of a vocabulary of elementary closed solids to the complexity of many real three-dimensional solid objects.

Some interesting design strategies follow directly from combining these operations with specific geometric constructions. Perhaps the most obvious is to locate solids so that their faces are coplanar, then to perform union operations to build complex solids from simple ones (figure 11.24). This is closely analogous to gluing wooden blocks together or assembling pieces of cut stone.

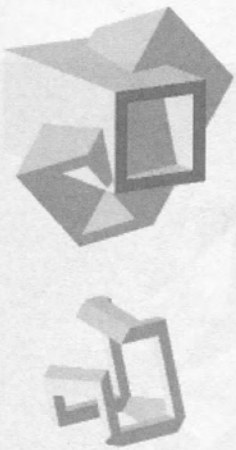
But these logical solids, unlike actual pieces of wood, can overlap in space. Unioning overlapping solids can



11.22
The spatial set operations applied to elements of increasing dimensionality

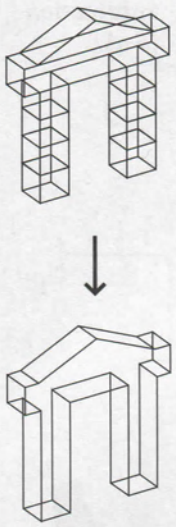
produce surprising results. Figure 11.25, for example, shows the effect of snapping two cubes together at a vertex, rotating one of them, then taking the union. If a third cube is unioned in the same relation to the second as the second is to the first, and so on recursively, a complex symmetrical polyhedron soon emerges.

When the subtraction operation is used, one shape becomes a "cutting tool" on the other—much as a drill bit is a tool for subtracting a cylindrical solid from a piece of material. In general, subtraction can be used effectively to model construction components that are produced by material-removal operations such as drilling, sawing, carving, planing, and milling.



11.23
Complexity from simplicity

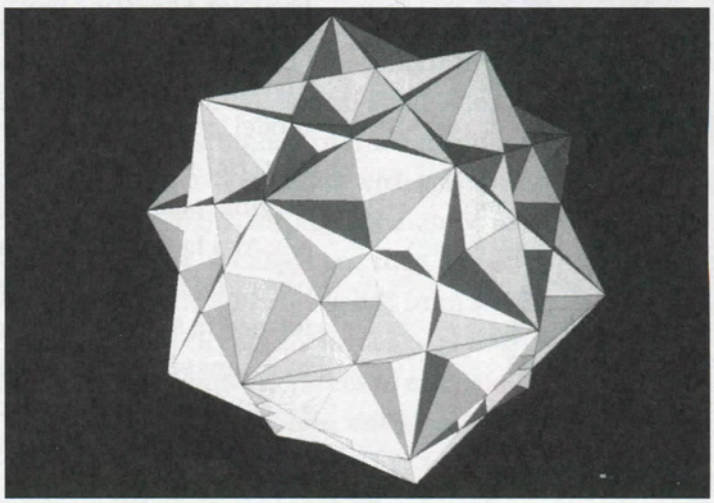
At a larger scale, architectural elements that are *conceptually* subtracted (even though they may actually be formed in some other way) can appropriately be modeled by means of subtraction operations. Consider, for example, a rectangular solid representing a wall. By locating translationally swept solids so that their sweep axes are



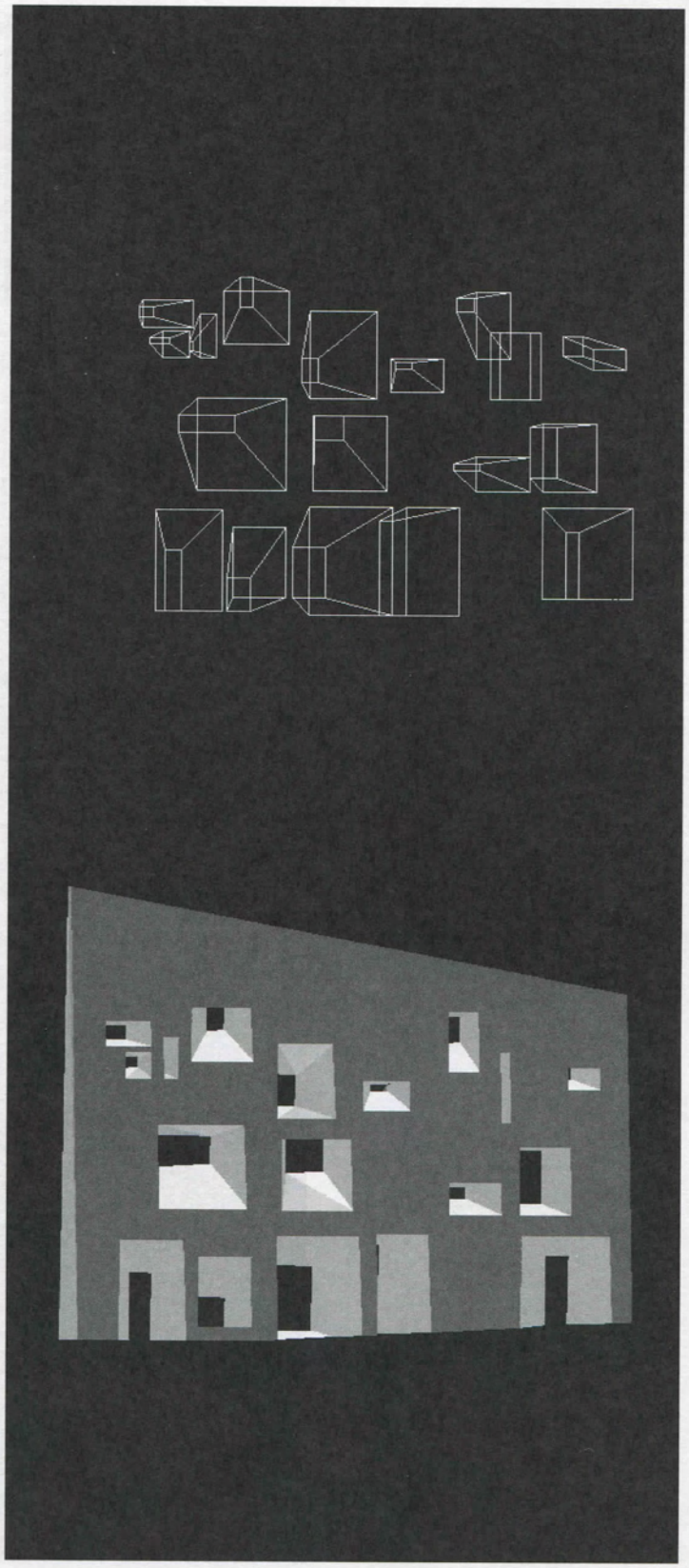
11.24
Gluing

perpendicular to the vertical faces, then subtracting, you can quickly cut out the usual sorts of simple door and window openings. The principle can be generalized, to produce a much richer variety of openings, by locating the subtracted solids in nonperpendicular positions, by subtracting nonprismatic solids such as cones and spheres, and by subtracting from more complex wall solids. Thus the famous window openings at Ronchamp, for example, can quickly be produced by subtracting angled pyramids from a wedge-shaped wall (figure 11.26).

Subtraction can also be used to hollow out interiors. For example, locate a rectangular box in plan, locate a smaller box inside it, and subtract to produce an interior room (figure 11.27). What remains of the original box becomes the wall *poché*. Or subtract a smaller hemisphere from a concentric larger one to generate a hollow dome. If you take a strict modernist attitude you will probably want the subtracted interior form to be similar to the exterior form, so that the exterior reveals the interior. But if you think more like a baroque architect you will probably want to subtract a dissimilar interior form—leaving a complex *poché* to mediate the differences between interior and exterior. Sir Christopher Wren's dome for Saint Paul's Cathedral in London, for example, has one shape on the exterior, a very different shape on the interior, and a vast *poché* volume taking up the difference (figure 11.28).



11.25
Complex solid produced by copying, rotating, and unioning cubes



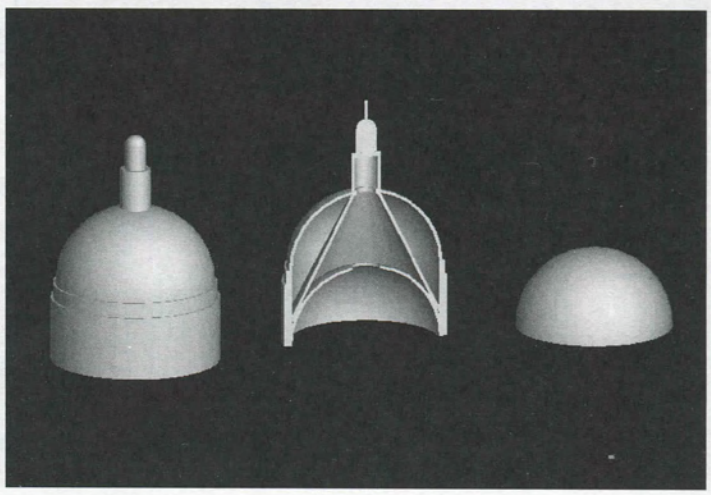
11.26
Solid model of window forms in south wall at Ronchamp

11.27
Interior volumes hollowed out by copying, scaling, and subtracting

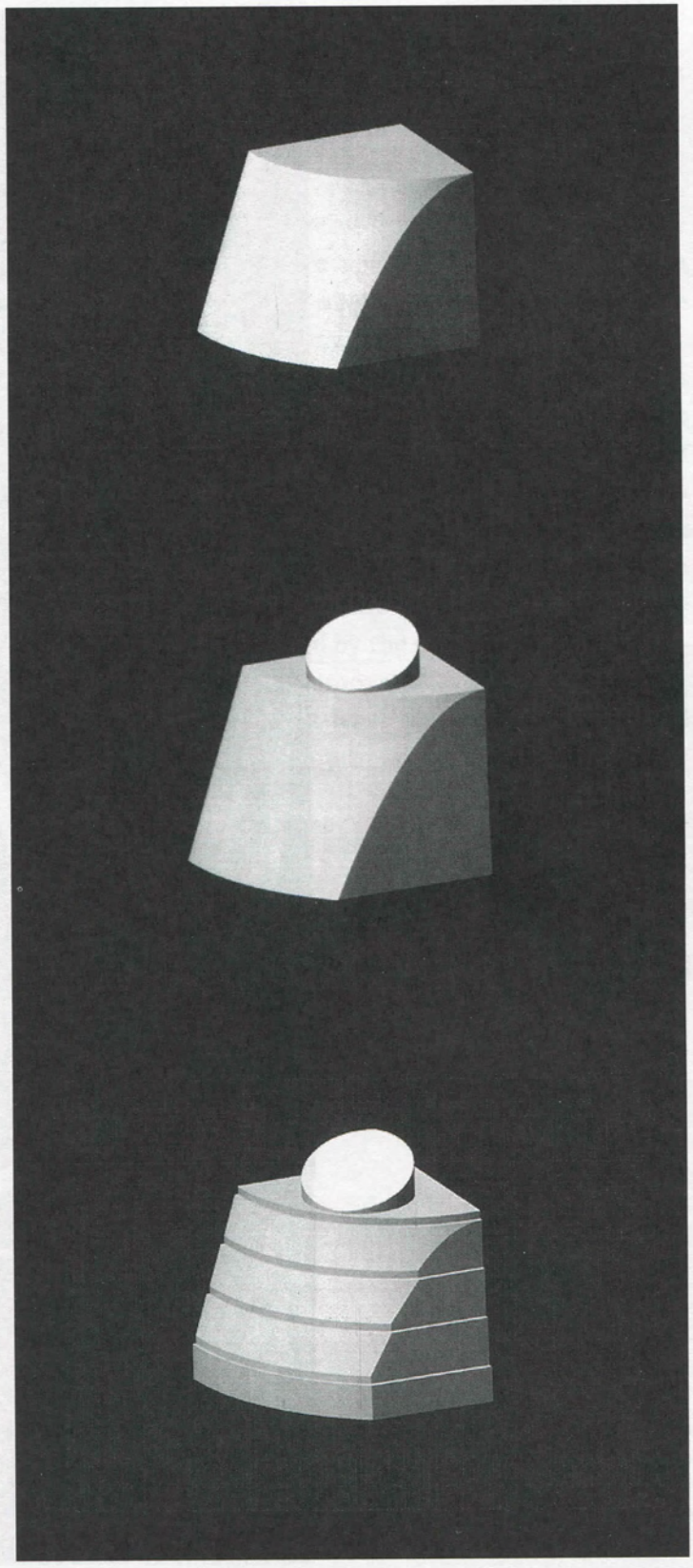
Wren, the great mathematician, would certainly have enjoyed the possibility of exploring geometric possibilities with a solid modeler.

Now imagine cutting a prismatic shape from the center of a rectangular block of polystyrene with a hot wire, cutting another prismatic shape from another direction, and removing the resulting core from the interior of the block. This core is formed by the intersection of the two prismatic shapes. The stonemason's strategy of projecting prisms from profiles drawn on the surfaces of a block, then removing everything except the intersection, illustrates the same idea.

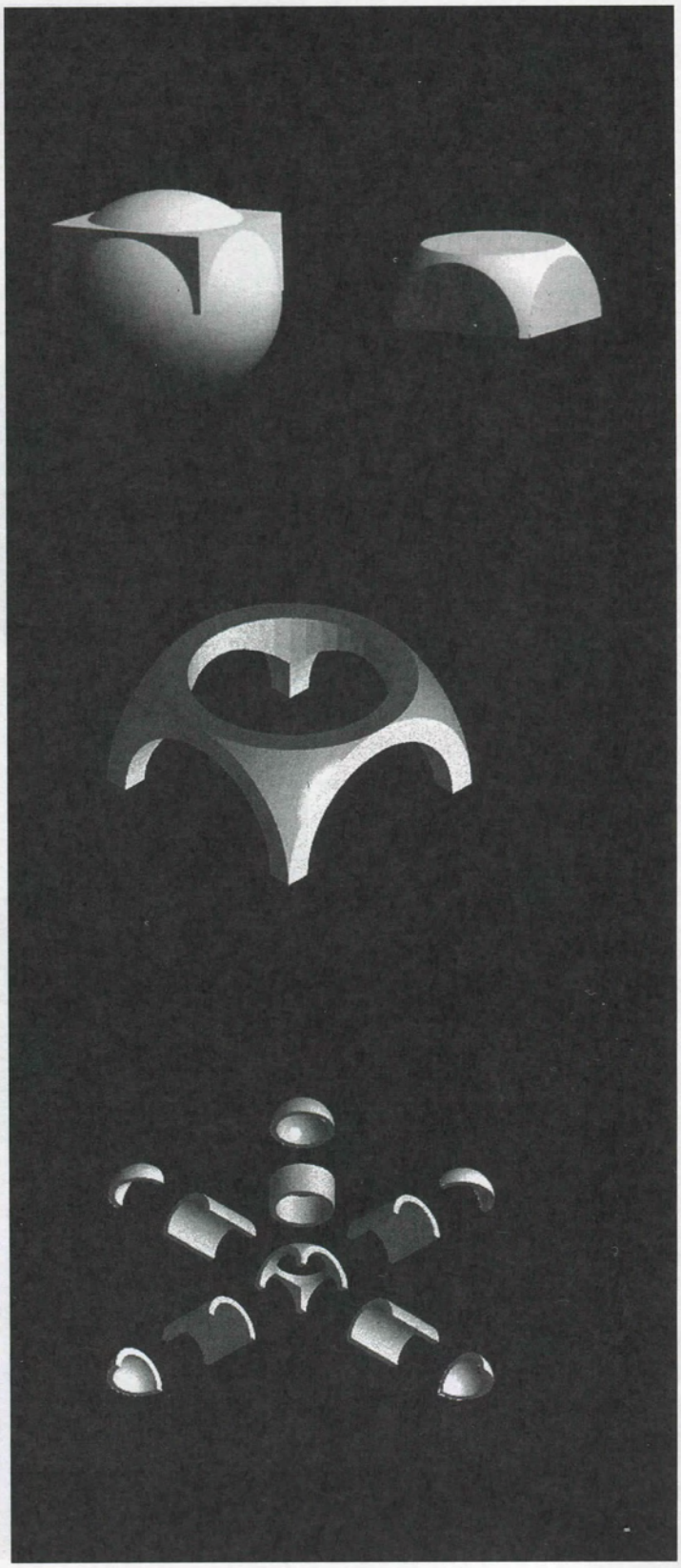
Strategies for volumetric design by intersection generalize this idea. You can construct the basic form of Helmut Jahn's State of Illinois Center in Chicago, for example, by fitting a cone (resulting from setback requirements) into the corner of a rectangular box (defined by the Chicago street grid) and then intersecting (figure 11.29). Similarly, you can construct the form of pendentives making the transition between a square plan and a hemispherical dome by fitting a rectangular box into the equator circle of a sphere and then intersecting (figure 11.30). This construction can be generalized for production of column capitals and many other kinds of transitional solids: the plan shape can be not just a square but any polygon, and the intersected solid can be almost anything that is nonprismatic—an elliptical spheroid, a cone, a pyramid, or whatever.



11.28
Exterior volume, *poché*, and interior volume in a solid model of the dome of Saint Paul's Cathedral



11.29
Basic form of the State of Illinois Center modeled by intersection and union

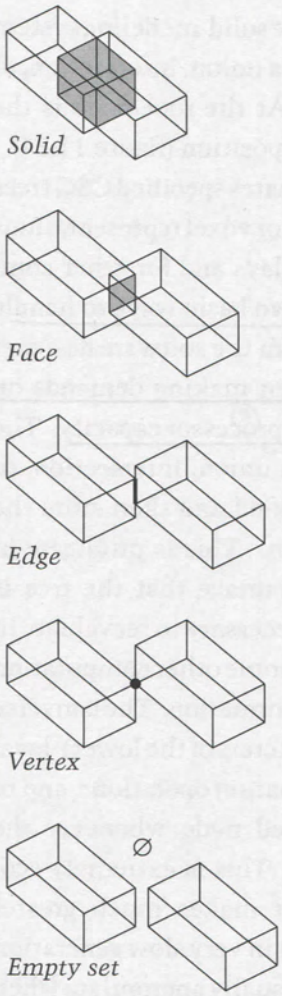


11.30
Pendentive and vault forms modeled by intersection and subtraction

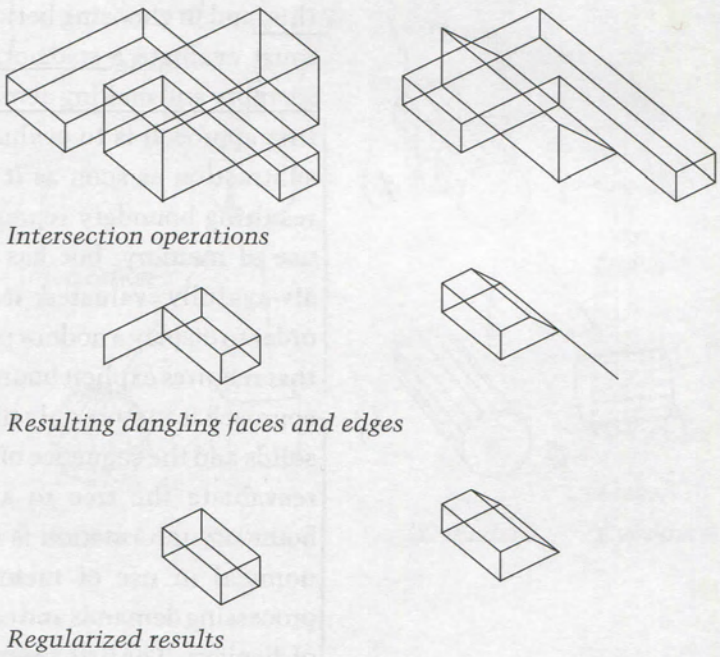
Regularizing the Spatial Set Operations

A difficulty with the spatial set operations on solids is that they are not closed—they do not necessarily yield solids. In general, the intersection of two solids may yield a solid, a face, an edge, a vertex, or the empty set (figure 11.31). Some solid-modeling systems leave it to the user to make sure that operations are specified so as to produce nondegenerate solids, but a better approach is to eliminate the problem by regularizing the spatial set operations.

The basic idea is to partition a solid (considered as a point set) into interior points and boundary points (figure 11.32). Boundary points are defined as those whose distance from the solid and the solid's complement are zero. In general, boundaries may include dangling and floating edges and the like—the undesirable sorts of things that can result from ill-specified spatial set operations. These blemishes can be removed by the operation of regularization, which amounts to removal of every boundary point that is not adjacent to at least one interior point. The regularized union, intersection, and subtraction operations



11.31
Possible results of intersection operations



11.32
Regularization

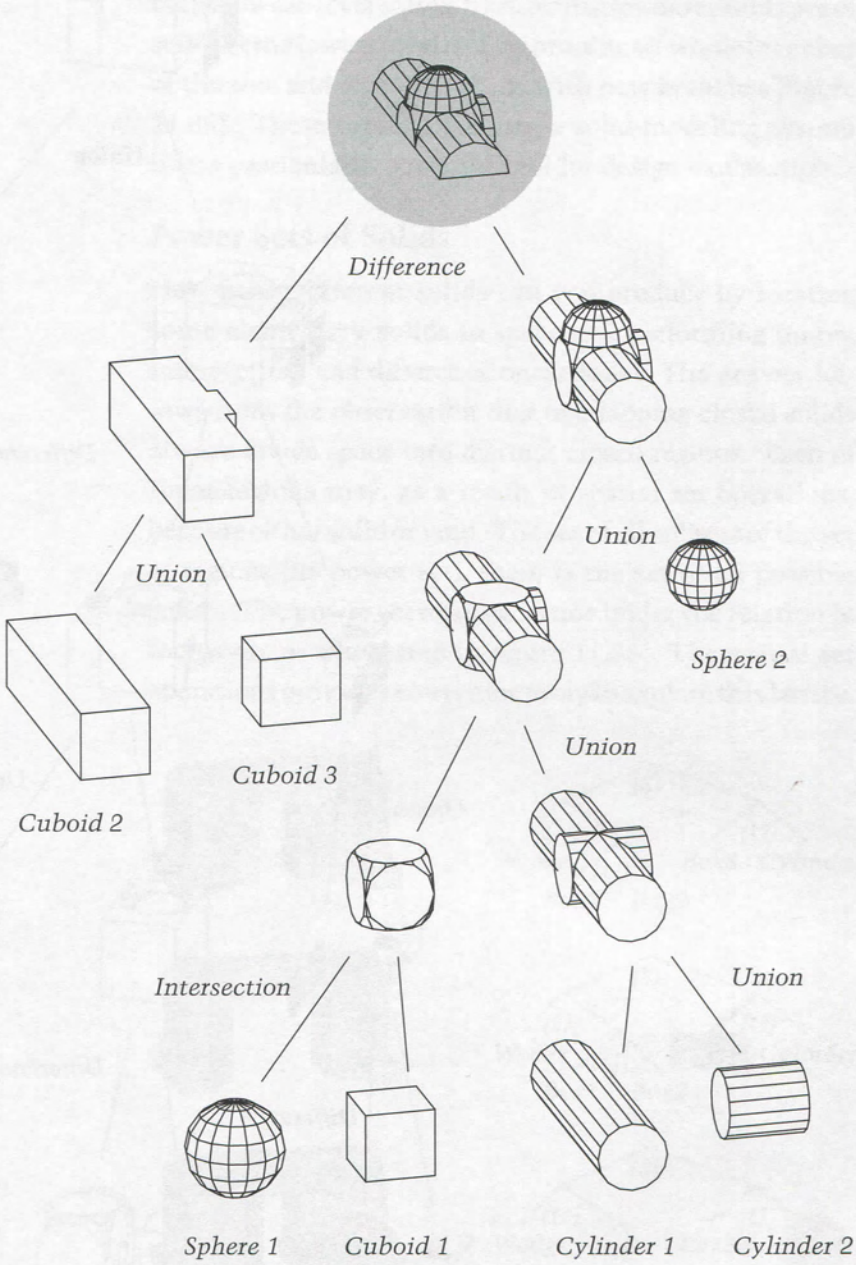
can then be defined so that they apply to regularized solids, and are closed in the regularized solids.

A related problem results from the round-off errors inherent in floating-point arithmetic. Faces that appear to be coplanar may overlap to produce intersection slivers and the like. Careful dimensional control, by snapping to grids and so on, is the best way to eliminate this possibility.

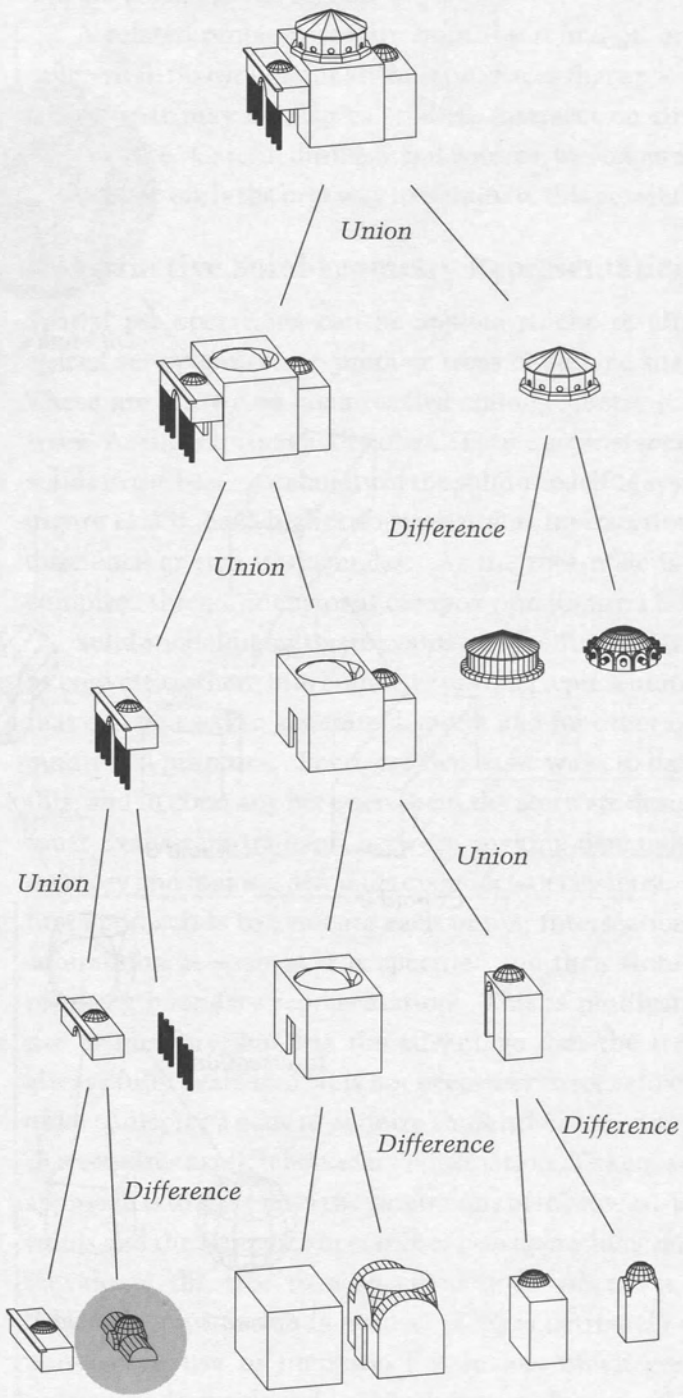
Constructive Solid-geometry Representations

Spatial set operations can be applied to the results of spatial set operations to produce trees of derived shapes. These are known as constructive solid-geometry (CSG) trees. At the terminal nodes of a CSG tree are instances of solids in the basic vocabulary of the solid-modeling system (figure 11.33). Each higher node is a union, intersection, or difference of two lower nodes. At the root node is the complete three-dimensional composition (figure 11.34).

Solid-modeling software evaluates specified CSG trees by converting them into boundary or voxel representations that can be used to generate displays and for other computational purposes. There are two basic ways to handle this, and in choosing between them the software designer must evaluate a trade-off between making demands on memory and making demands on processor capacity. The first approach is to evaluate each union, intersection, or subtraction as soon as it is specified and then store the resulting boundary representation. This is profligate in use of memory, but has the advantage that the tree is always fully evaluated: it is not necessary to reevaluate in order to display a node or perform some other computation that requires explicit boundary information. The converse approach is to store only the parameters of the lowest-level solids and the sequence of combination operations, and to reevaluate the tree to a specified node whenever the boundary information is needed. This is extremely economical in use of memory, but makes much greater processing demands and can result in very slow generation of displays. The first approach is usually appropriate when processor speed constrains performance more than memory limitations, and the second tends to be appropriate when the converse is true.



11.33
Lower branch of a CSG tree showing modeling of a ceiling void from geometric primitives



11.34
Root of a CSG tree showing complete assembly (lower branch from previous figure attaches at the node marked with a circle)

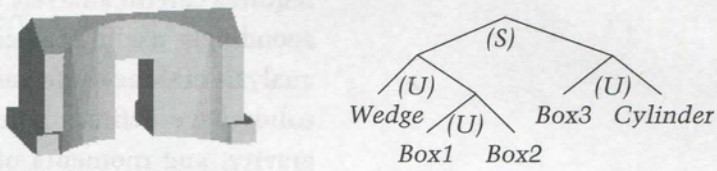
From a designer's viewpoint, however, the evaluate-as-needed approach has an additional advantage. It permits fluid variation of a design by manipulating the parameters of the lowest-level solids, by substituting different types of solids at the lowest level, and by pruning off whole branches of the tree and replacing them with new branches (figure 11.35). These capabilities turn a solid-modeling system into a particularly powerful tool for design exploration.

Power Sets of Solids

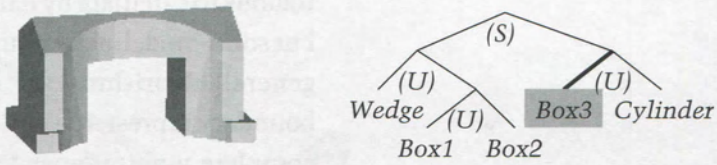
How many different solids can you produce by locating some elementary solids in space and performing union, intersection, and difference operations? The answer follows from the observation that overlapping closed solids always divide space into distinct closed regions. Each of these regions may, as a result of spatial set operations, become either solid or void. The set of all subsets of the set of regions (its power set), then, is the set of all possible solids. The power set forms a lattice under the relation of inclusion, as illustrated in figure 11.36. The spatial set operations provide a convenient way to explore this lattice.

11.35
CSG tree editing

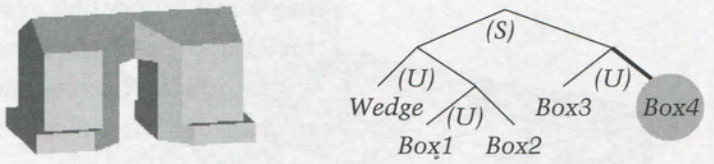
CSG model



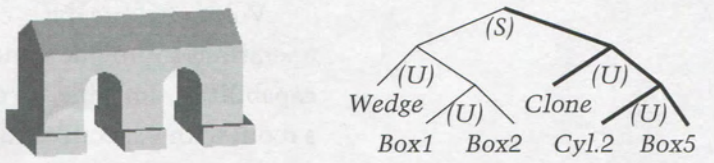
Primitive variation

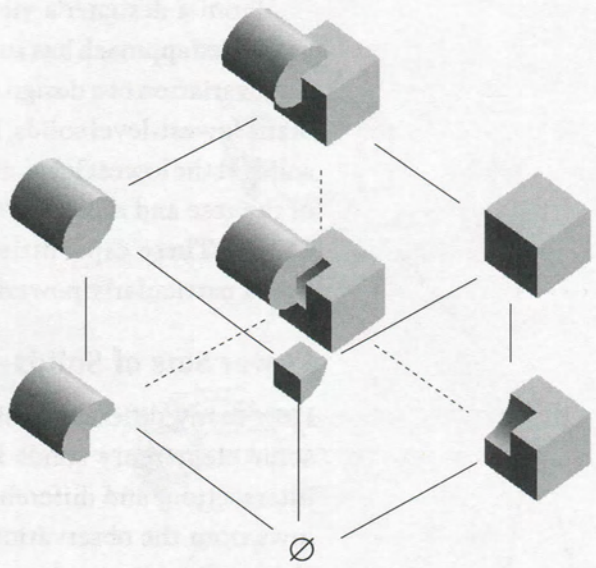


Primitive substitution



Branch substitution





11.36
Power set of the closed regions formed by two overlapping solids

Volumetric and Engineering Analysis

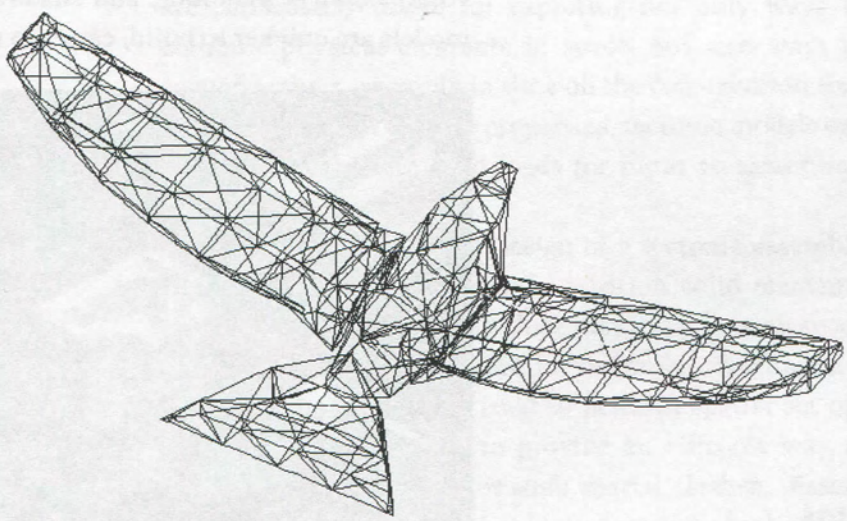
Just as polygon-modeling systems are particularly useful for urban design, space planning, and other work that requires careful analysis of areas, solid models are correspondingly useful for design work that requires careful analysis of three-dimensional forms. In particular, closed solids have definite volumes, surface areas, centers of gravity, and moments of inertia. These properties are tedious to calculate by hand for any but the simplest cases, but solid-modeling systems can be equipped with efficient, general algorithms for deriving them from voxel or boundary representations. (This is not possible with less complete wireframe and surface models.) Thus you can use a solid-modeling system to measure the amount of material to be cast in a form, to measure the volume of an auditorium for heating and cooling or acoustic analysis, or to measure the volume of a building for urban design analysis.

Volumetric analysis can be combined with spatial set operations to provide some very powerful problem-solving capabilities. Imagine, for example, that you need to design a room with a specified volume. Instead of choosing some

simple shape (such as a rectangular box) to make the volume calculations easy, you might write a procedure to push two complex solids together along an axis, generate their intersection at each step, and calculate the volume at each step. If you did not like the shapes of appropriate volumes that resulted from this, you could try pushing them together along another axis.

Furthermore, the data structures of solid-modeling systems can be extended to provide for association of material properties such as density with solids. Associated algorithms can then derive additional properties. From volume and density, for example, the mass of a solid can be calculated. And if you know the location of the center of gravity, you know where this mass acts.

For detailed analysis of engineering properties, solids may be broken up into small pieces, known as finite elements, as illustrated in figure 11.37. Advanced solid-modeling systems provide algorithms for automatically constructing finite-element meshes from boundary models. Once this has been accomplished, finite-element analysis procedures can be used to produce detailed and accurate analyses of structural properties, thermal properties, and so on.



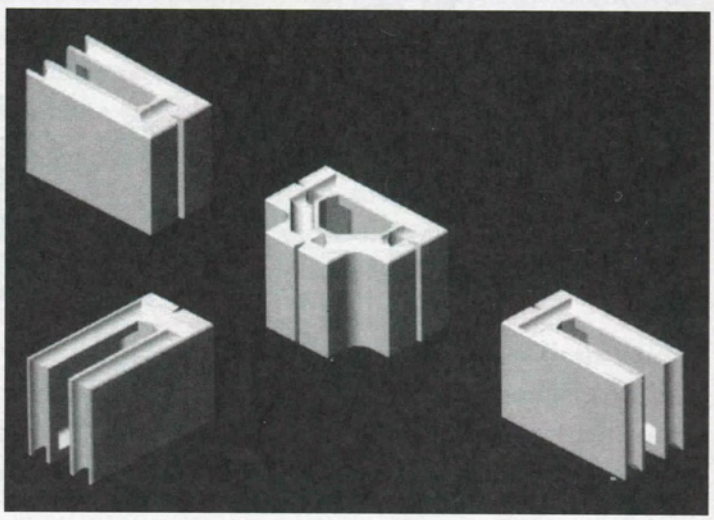
11.37
A finite element mesh

Assemblies

In the same way that a three-dimensional physical model can be assembled from wooden or plastic components, a digital model of a building can be assembled from discrete solids (figure 11.38). Such a model is the three-dimensional equivalent of the polygon maps that we considered in chapter 8: it exhaustively and unambiguously describes the occupancy of space by subdividing space into bounded pieces.

Before constructing a solid model of a building you must decide how to use solids to represent architectural elements. If you are making an exterior massing model, for example, the solids in the model will stand for major volumetric elements—much as blocks of wood or polystyrene stand for these elements in physical massing models. You may want to use such models not only to generate images, but also to analyze basic geometric properties of massing alternatives: you can compute volumes and surface areas, cut horizontal slices to reveal floor plates, and section vertically to study profiles.

For urban design purposes you can assemble simple exterior massing models of buildings into three-dimensional models of urban fabric. You can include in the model not only actual building volumes, but also notional volumes such as allowable building envelopes, air-rights volumes, view pyramids, and shadow volumes. These models are quicker to build, easier to modify and update,



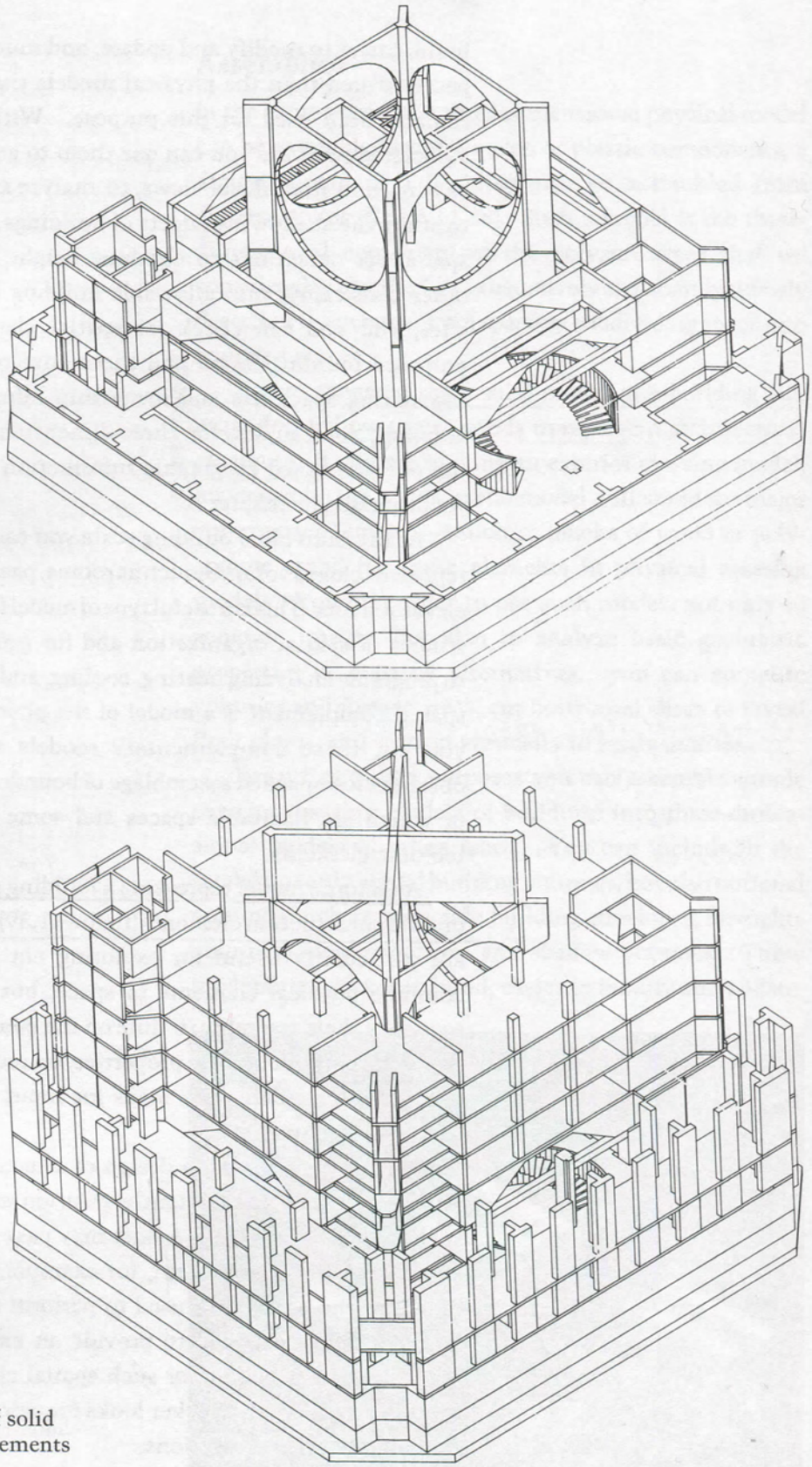
11.38
Components of a solid assembly

build, easier to modify and update, and much more compactly stored than the physical models that have traditionally been used for this purpose. With appropriate associated software you can use them to generate aerial, skyline, and street-level views, to analyze sightlines, and to study the shadowing effects of buildings. You can use spatial set operations to combine height, setback, and other constraints into allowable building envelopes for sites, and you can check for spatial clashes between proposed building forms and these envelopes. And by associating space use and ownership information with volumes, you can develop three-dimensional versions of the polygon-based geographic information systems that we discussed in chapter 8.

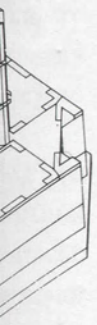
At the individual building scale you can use solids to represent closed volumes such as rooms, passageways, and elevator shafts. This is a useful type of model for preliminary studies of spatial organization and for generating input to programs analyzing heating, cooling, and acoustics. Its spatial complement is a model of the physical fabric. If you put these complementary models together, you obtain a close-packed assemblage of bounded forms, some of which are habitable spaces and some of which are tectonic elements.

A tectonic model represents a building as a collection of solid construction elements (figure 11.39). Such models are particularly useful for exploring not only ways to combine physical elements in space, but also ways to sequence their assembly in time on the construction site. By taking account of mass properties, tectonic models can be used to calculate dead loads for input to structural-analysis programs.

One of the hazards in design of a tectonic assembly is that you may inadvertently position solid elements such that they intersect. A duct may pass through space already occupied by a beam, for example. Fortunately, however, the procedures used to perform spatial set operations can be adapted to provide an efficient way of automatically checking for such spatial clashes. Essentially, a spatial clash checker looks for pairs of solids that have nonempty intersections.



11.39
An assembly of solid construction elements



The task of modifying an assembly of solids to reflect design changes can become laborious and frustrating, since changes in the position or dimensions of one solid element may propagate long chains of necessary adjustments to other elements. If columns are moved further apart, then you need to lengthen the beam that they support, then the slab supported by the beam must be correspondingly lengthened, and so on. This process of adjustment can be automated if component solids are defined as parametric objects and their relationships are described by formulae that relate parameters so that, in effect, the whole assembly is programmed to behave appropriately in response to changes in dimensions or locations of parts. Some advanced modelers provide for this. Specifying an appropriate structure of relationships for a complex three-dimensional assembly may, however, prove to be a very difficult problem.

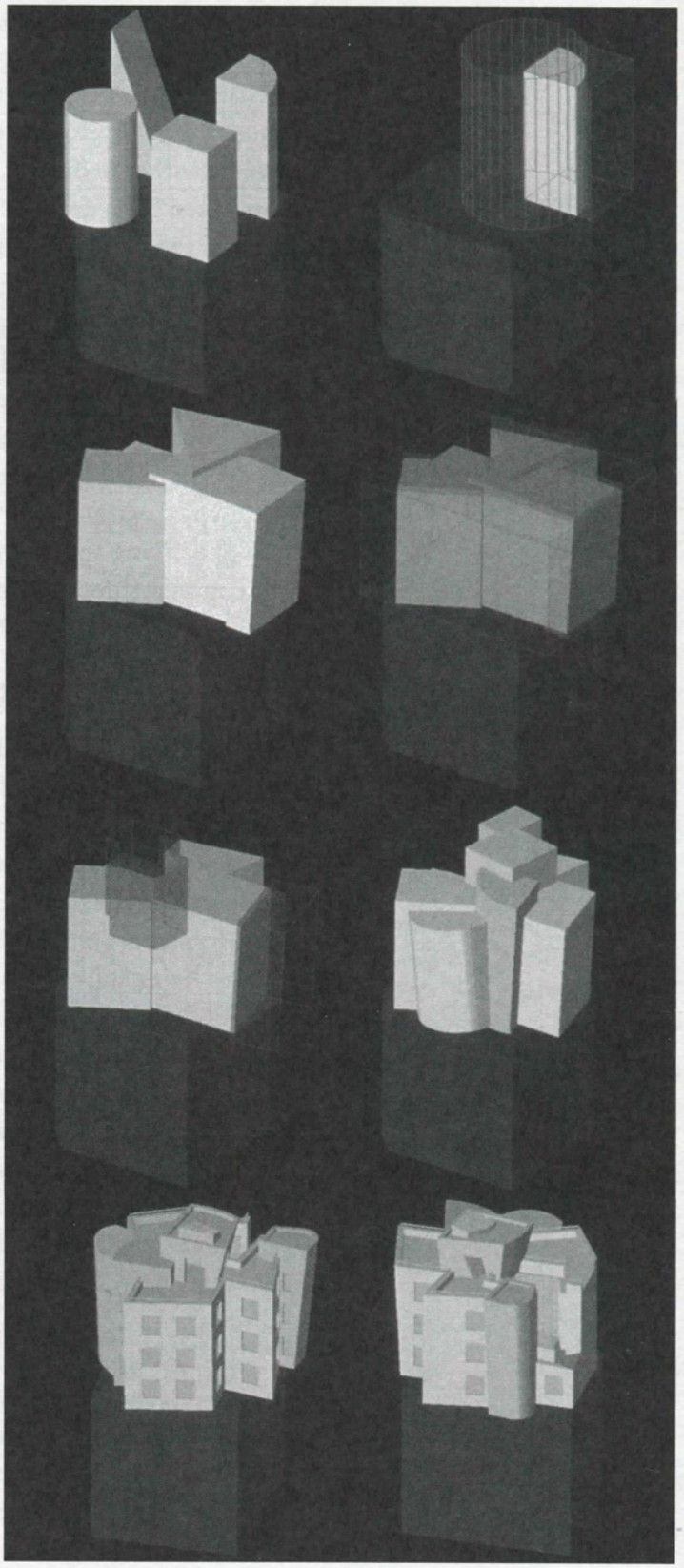
Nonmanifold Assemblies

For some purposes, solid-assembly models may be too realistic. A designer might, for example, want to include freestanding wireframes and floating surfaces (as well as closed solids) in an assembly model to serve as a construction skeleton (figure 11.40). Furthermore, it may be useful to have operations that combine elements of different types—slicing solids with surfaces, extracting medial axes from solids, and so on. The data structure of a typical solid-assembly modeler is, unfortunately, not designed to allow this.

More technically, solid modelers are usually based on the assumption that solids have enclosing shells of surfaces and that these enclosing surfaces are two-manifolds (figure 11.41). Such shells always obey Euler's polyhedron law, which may be stated:

$$V - E + F - R = 2\,(S - H)$$

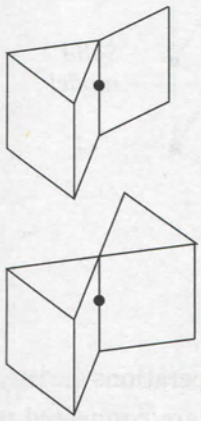
where V, E, and F are, respectively, the numbers of vertices, edges, and faces, H is the number of holes, and R is the number of rings. This means, it turns out, that each edge must be incident at exactly two vertices, and each face



11.40
Constructions and stages in the evolution of a solid model

must be incident at exactly two edges. The data structures of solid modelers are designed to accommodate objects that obey this law, and operators that manipulate those data structures are designed to preserve consistency with it. This excludes stand-alone and dangling faces and edges (the explicit intention of the regularized spatial set operators)—an appropriate exclusion if the intention is to model a world of solid objects, but not if the intention is to model a designer's nonmanifold world in which such abstractions play an important role.

Two-manifold shell

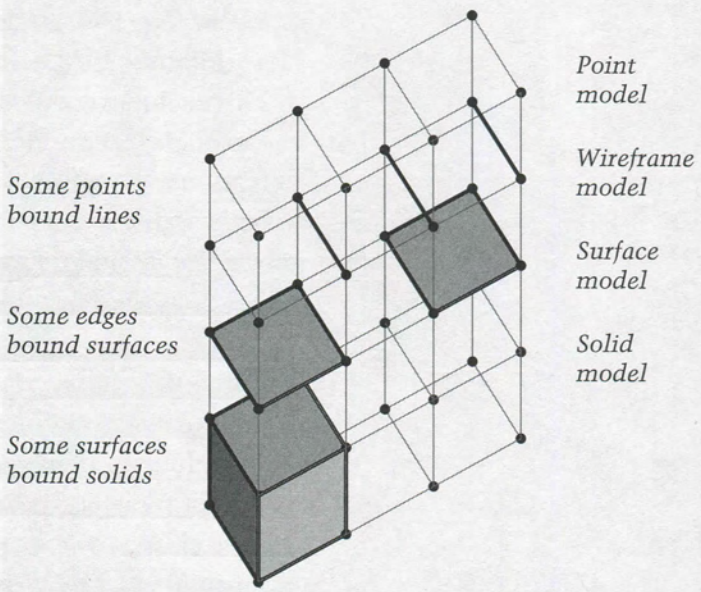


Nonmanifold assemblies of surfaces

11.41
Manifold and non-manifold shells

Nonmanifold geometric modelers, then, are systems that provide for assemblies of vertices, edges, faces, and solids into configurations that do not satisfy Euler's law. They include in their repertoires not only the usual operators for transforming and combining elements in each of these classes, but also operators that are not closed in one or another of the classes. These include operators that assemble edges to produce faces, that assemble faces to produce solids (the skinning operation discussed earlier), and that reduce solids to more abstract representations (which do not obey Euler's law) by pulling off faces, performing medial axis transforms, and so on. Thus they provide environments for incremental transformation of an abstract three-dimensional *parti*—a skeleton of lines and freestanding faces—into a complete, consistent solid-assembly model.

A full-featured, design-oriented geometric modeler might support four submodels: a point model, a wireframe model, a surface model, and a solid model. In the data structure, entities of lower-level models are associated to define entities of higher-level models: points bound lines, lines bound surfaces, and surfaces bound solids. Models at each level are regularized: the wireframe has no isolated points, the surface model has no isolated or dangling lines, the solid model has no isolated or dangling faces, and regularized union, intersection, and subtraction operators are used at each level. However, there may be points in the point model that do not bound lines in the wireframe, lines in the wireframe that do not bound surfaces in the surface model, and surfaces in the surface model that do not bound



11.42
Entities in a full-featured geometric modeler

solids in the solid model. Association operations are used to create higher-level entities: points are connected to make lines, lines are connected to make surface facets, and solids are skinned by surfaces. Conversely, cutting operations are used to create lower-level entities: solids are cut by surfaces to make surfaces or by lines to make lines, surfaces are cut by surfaces to make lines or by lines to make points, and lines are cut by lines to make points.

Producing Graphic Output

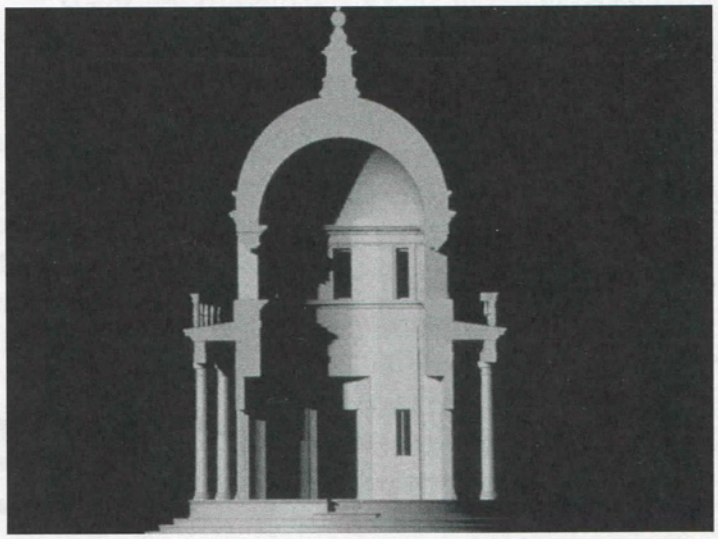
Solid-assembly models implicitly embody surface and wireframe models, so all of the types of renderings that can be produced from these less spatially complete models can be produced from solid-assembly models as well. Some additional types of graphic output are also made possible by the additional spatial information that a solid-assembly model contains.

First, you can cut arbitrary sections—at any location and angle and to complex section surfaces as well as to planes. Sectioning to a vertical plane to produce a traditional architectural section, to a horizontal plane to produce a plan, or to an inclined plane, is accomplished by subtracting a half space or a very large box. A thin slice can

be cut out by intersecting the model with a slab-shaped object. Sometimes it is useful to intersect with more complex shapes to produce cylindrical cores, and so on.

Sectioning by intersection with a plane, rather than by subtraction or by intersection with a thin solid, is a good example of the use of a nonmanifold modeler. The result of sectioning a solid-assembly model by intersection with a plane is a set of two-dimensional *poché* polygons. These might be kept on a separate layer, since they are not really part of the basic geometric model, or they might be transferred to a two-dimensional drafting system for use in production of a section drawing.

Sectioned three-dimensional models may be shown in perspective or axonometric projection, or they may be projected orthographically onto planes parallel to the section planes to produce more traditional plan and section drawings. Section-cutting software can be extended to keep track of the new faces generated by the cut so that these faces can be shaded or outlined to show the *poché*. Many different rendering techniques may be used, depending on what you want to emphasize: among the options are wireframe with *poché*, hidden line with *poché*, hidden line with cast shadows, shaded, and shaded with cast shadows (figure 11.43).



11.43
A sectional perspective

Automated Production of Physical Models

The CAD/CAM techniques that have been developed for use in the manufacturing industry can sometimes be adapted effectively for production of physical topographic and architectural models from numerical data describing solids. If a model is broken down into planar surface facets, for example, then a computer-controlled laser cutter can be used to cut the facets from thin sheet material: finely detailed wooden models of buildings and contoured topographic surfaces can be produced in this way. Alternatively, the computer-controlled milling machines that now find wide application in the manufacturing industry can be employed to produce complex solid parts in metal or high-density foam. Stereolithography is perhaps the most versatile technique, and despite its technical complexity and high cost it has rapidly found a niche in medical imaging and mechanical parts design. A stereolithography system employs a computer-controlled laser to solidify liquid resin, in layer-by-layer fashion, to build up a solid.

All of these techniques make use of complex, expensive machinery that most designers are unlikely to have in-house. They will increasingly be made available by model-making and prototyping service bureaus, however. There are also some inexpensive alternatives. One effective way to streamline model production is to print facet shapes with a laser printer, mount them on cardboard, and cut them out with a matte knife.

Uses and Limitations of Solid Models

Solid models of parts and solid-assembly models have a higher level of geometric completeness than corresponding bitmapped images, drafted drawings, wireframe models, and surface models. This is the source of both their advantages and disadvantages.

In contexts where completeness and consistency of geometric representation are crucial, where it is necessary to integrate a wide range of applications around a geometric model, or where designers want to work in a directly sculptural way (rather than rely on abstractions like plans

and sections), solid and solid-assembly models are particularly appropriate. But, since they must store more coordinate information and keep track of more topological relationships, solid models of artifacts tend to be much larger and more complex than less complete types of representations. This means that they make heavier demands on memory, computational capacity, and software engineering technique. A designer must consider whether the advantages of greater completeness justify the higher associated costs.

There is also a more subtle issue of representational economy. At an early stage in a design process a designer is usually interested in rapid, unencumbered exploration of ideas. Ambiguities do not cause major problems and may even become sources of creative ideas. Many inconsistencies can safely be ignored on the assumption that they will be dealt with later if the idea turns out to be a good one, but they are not worth attention if the idea is to be abandoned anyway. In this context sparse, economical representations that are easy to manipulate and do not mire the designer in demands for detail usually work better than representations that emphasize completeness and consistency. Later, when the focus shifts to resolving problems, working out details, analyzing cost and performance precisely, and producing complete documentation, abstract representations become less appropriate and techniques like solid modeling become more attractive.

The practical usefulness of solid-modeling technology has grown with the availability of computing power and the sophistication of available software engineering techniques, and this trend will continue. Prototype solid modelers emerged in the 1970s, but the commercially available systems that followed in the 1970s and 1980s were limited, slow, expensive, and often unreliable. By the end of the 1980s, however, robust and effective solid modelers were available on inexpensive personal computers and workstations, and they were becoming increasingly popular. As solid-modeling software exploits the capacities of increasingly powerful computers, it will be bound by fewer limitations on the topologies and geometries that

it can process, it will be capable of handling larger and more complex projects, and it will increasingly emphasize real-time geometric transformation and spatial set operations, stereo and virtual reality interfaces, and other features that support swift, fluid manipulation of designs.

Suggested Readings

Foley, James D., Andries van Dam, Steven K. Feiner, and John F. Hughes. 1990. "Solid Modeling." In *Computer Graphics: Principles and Practice.* Reading, Mass.: Addison-Wesley.

Goult, R. J. 1987. "Finite Element Analysis." In Joe Rooney and Philip Steadman (eds.), *Principles of Computer-Aided Design.* London: Pitman.

Hoffmann, Christoph M. 1989. *Geometric and Solid Modeling: An Introduction.* San Mateo: Morgan Kaufmann.

Jared, G. E. M., and J. R. Dodsworth. 1987. "Solid Modeling." In Joe Rooney and Philip Steadman (eds.), *Principles of Computer-Aided Design.* London: Pitman.

Kalay, Yehuda E. 1989. *Modeling Objects and Environments.* New York: John Wiley.

Koenderink, Jan J. 1990. *Solid Shape.* Cambridge: The MIT Press.

Mantyla, Martti. 1988. *An Introduction to Solid Modeling.* Rockville, Md.: Computer Science Press.

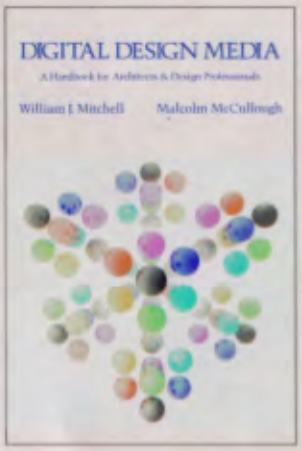
Mortensen, Michael E. 1985. *Geometric Modeling.* New York: John Wiley.

Mullineux, Glen. 1986. *CAD: Computational Concepts and Methods.* New York: Macmillan.

Raper, Jonathan (ed.). 1989. *Three Dimensional Applications in Geographical Information Systems.* London: Taylor & Francis.

Weiler, Kevin J. 1986. "Topological Structures for Geometric Modeling." PhD. thesis, Rensselaer Polytechnic Institute.

Yessios, Chris. 1987. "The Computability of Void Architectural Modeling." In Yehuda E. Kalay (ed.), *The Computability of Design.* New York: John Wiley.



In *Digital Design Media, Second Edition,* architects and related design professionals will find a complete conceptual guide to the multidimensional world of computer-aided design. In contrast to the many books that describe how to use particular programs (and which therefore go out of date very quickly), *Digital Design Media* constructs a lasting theoretical framework, which will make it easier to understand a great number of programs—existing and future—as a whole. Clear structure, numerous historical references, and hundreds of illustrations make this framework both accessible to the nontechnical professional and broadening for the experienced computer-aided designer.

The book will be especially valuable to anyone who is ready to expend their work in CAD beyond production drafting systems. The new second edition adds chapters on emerging technologies, such as the Internet, but the book's original content is as valid as ever. Thousands of design students and practitioners have made this book a standard.

William J. Mitchell is Dean of the School of Architecture and Planning at the Massachusetts Institute of Technology. His other books include *Computer-Aided Architectural Design* and *The Art of Computer Graphics Programming* (with R.S. Liggett and Thomas Kvan).

Malcolm McCullough is Associate Professor of Architecture at the Harvard University School of Design.

ISBN 0-442-01934-3
90000
9 780442 019341

VAN NOSTRAND REINHOLD
115 Fifth Avenue, New York, NY 10003