

# 3D Object Modeling - Issues and Techniques

- Preliminary Report -  
01 October 2003

Ana-Maria Cretu, Dipl. Eng.  
SITE, University of Ottawa

*This chapter provides an overview of 3D object modeling approaches. It begins by briefly describing the issues in 3D object modeling and their impact on the quality of the resulting model. Then it looks in depth on classical 3D object modeling techniques and what they involve. It discusses advantages and disadvantages and gives possible applications for each described method. The presentation is not meant to be exhaustive and it only provides a background for the study of neural networks in the context of 3D object modeling. The chapter ends up by discussing several neural architectures and examining their use in object modeling.*

## 1. Issues in Object Modeling

The closest definition of “modeling” to the context of 3D graphics and therefore to the purpose of this thesis can be expressed as: the process of giving a “schematic description” of the objects that compose a scene, “that accounts for their known or inferred properties and may be used for further study of their characteristics”[AHD]. This is however, not an easy task. The structure inherent in 3D object scenes is difficult for people to understand, and is also difficult for user interfaces to display and manipulate. The big challenge in 3D graphics is “to make simple modeling easy and complex modeling accessible” [HLL91].

There are several aspects that have to be considered when building an object model, that contribute all together to the quality of the final result. They include the methods to acquire or create the data that describes the object, the purpose of the model, the implementation complexity, its computational cost, its conformance and convenience, and the ease of model manipulation.

### **1.1. Methods to define an object**

There are a numerous ways in which objects can be defined: directly giving the geometric description (e.g. the equation that describes the object), writing programs to create data or using modeling programs, using reconstruction from photographs or using real data in form of clouds of points (a number of points with only their respective (x, y, z) coordinate triplets) obtained from range scanners, optical fringe patterns, mechanical coordinate measuring devices or Computed Tomography (CT) scans. In short, any function or procedure that provides 3D surface data can provide a basis for modeling. However, the way of processing the data might differ for different sources, and thus needs to be taken into account at modeling time.

### **1.2. Purpose of the object model**

There are two main reasons for building 3D objects: “image making” and simulation [BGS97]. While “image making” requires that the model looks good (is realistic, sufficiently complex and conforms to the geometry of the model), simulation requires accurate models. While the first one is described as interactive and approximate, the other one is generally slow, but precise. The future use of an object model is thus an important consideration when choosing a certain modeling approach.

### **1.3. Computational cost**

Another important aspect to be taken into account is, knowing the needs of an application, what price is the user willing to pay in terms of computational costs. The computational cost of a model may be gauged in terms of computer storage space, object construction time, display time, or in application use [BGS97]. The storage space of a model refers to the space required to store the primitives that represent it. Judging the model in terms of object construction time means evaluating the model for the quantity (how much time does it need to be processed) and quality of the source data (how much time is needed to remove errors). The display time refers to the cost of object display, including time and image quality issues. Often, a real-time behavior of an object model is required.

### **1.4. Complexity**

Another question to be answered when evaluating an object model’s quality is how complex the model is. The complexity of a model may be judged as a trade-off between the number of primitives and the inherent complexity of the individual primitives. There are objects that are

composed of many simple forms or objects that are inherently complex. For example, man-made objects and mechanical parts can be seen as a set of simple forms, while biological and other natural structures are inherently complex [BGS97].

### **1.5. Conformance and Convenience**

Other qualities that the users look for in a model are its conformance and convenience. Conformance measures the degree to which the model actually represents the desired objects and provides a realistic, geometrically correct representation for the modeled object. Usually, the quality and the degree of the approximation of the model can be determined only by validation against experimental results. The convenience refers to the ease of performing extensive parametric studies on the described model, in which independent model parameters can be modified, in order to gain global knowledge on the properties of a certain modeled object [PTR03].

### **1.6. Ease of Model Manipulation**

Related to convenience, the user wants to gain the global knowledge of the objects' properties in an easy manner. An object model must be easily transformed in position, size or shape, and offer facilities for evaluating its geometric properties. The ability to manipulate an object depends greatly upon its representations. The most common representations of 3D objects are surface models, solid models and procedural models. The main ideas of the three categories, together with the advantages, disadvantages and most common uses will be further presented.

## **2. Surface Models**

Surface modeling techniques represent surfaces explicitly. Points in 3D space are then characterized as either on or off a given surface. Though surfaces can be joined to create the illusion of a solid object, the solidity is not inherent in the nature of the surfaces' descriptions. The most common surface representations are polygonal models and functional surfaces. Functional surfaces can either be explicit (parametric curves and surfaces) or implicit functions.

### **2.1. Polygonal Techniques**

Polygonal models are one of the most commonly encountered representations in computer graphics. The models are defined as sets of connected polygons, called *meshes*. The polygons are

sized, shaped, and positioned so that they completely tile the required surface at some resolution. Each polygon consists of some connected vertex, edge, and face structure.

An important thing to consider when modeling with polygons is the integrity of the polygons. Nonplanar polygons create holes in the objects, because the plane is twisted and doesn't render. The solution is to split the nonplanar polygon into three-sided polygons (these cannot become nonplanar) or use subdivision technology to smooth the mesh [FLM99]. Subdivision techniques add polygons to a model, in an effort to smooth rough edges.

### *Advantages*

Polygon models are relatively simple to define, manipulate, and display and are the commonest model processed by workstation hardware and commercial graphics software. They are flexible and allow varied mesh density according to the needs of a certain application. They are usually used to provide a low-resolution model, which can then be transformed into a high-resolution using subdivision.

### *Disadvantages*

There are also some limitations of modeling with polygonal meshes. The number of polygons needed to accurately define a complex object may be large. This has implications for memory usage and rendering times [GDM]. Also, smooth curves cannot be easily created with polygons. To create the illusion of smoothness, a large number of polygons needs to be used and numerous points edited [FLM99].

### *Applications*

The ability to work with low-resolution models, create varied density and apply minute details makes polygons the most appropriate technique for character and creature modeling. They are also suited for natural organics and linear models (e.g. furniture, computer equipment, building, city streets) because they do not require precision curves [FLM99].

## **2.2. Parametric Curves and Curved Surfaces**

Since polygons are good mainly at representing non-curved surfaces, considerable effort has been made to determine mathematical formulations for curves and curved surfaces. A *curve* can be defined as “the locus of a point moving with one degree of freedom or the locus of a one-parameter family” [MEM97]. A polynomial parametric curve of degree  $n$  is of the form:

$$p(u) = \sum_{k=0}^n u^k c_k \quad (1)$$

where each  $c_k$  has independent  $x, y, z$  components and  $u$  is the curve parameter. For  $0 \leq u \leq 1$  the above equation defines a curve segment. If a high degree polynomial is chosen, there will be many parameters that can set the desired shape of the curve, but the evaluation of points on the curve will be costly. In addition, as the degree of a polynomial becomes higher, there is more danger that the curve will become rougher [AEP03] and will need more computation time. A too low degree, on the other hand, will give too little flexibility in controlling the shape of the curve. Although they are still approximations of the real curve, parametric curves are more accurate and easier to manipulate than the polygonal models.

The curved surface is the natural extension of a curve. A *surface* is the locus of a point moving with two degrees of freedom and its polynomial parametric representation is:

$$p(u, v) = \sum_{i=0}^n \sum_{j=0}^m u^i v^j c_{ij} \quad (2)$$

A *surface patch* is defined for  $u, v \in [0, 1]$ . Patches are joined along their boundary edges into more complex surfaces. The shape of a patch is derived from control points or tangent vectors. The parametric form of curves and surfaces is the most flexible and robust for computer graphics [AEP03].

There are many formulations of curves and curved surfaces, including cubic Hermite, Bezier, rational Bezier, splines, B-splines, NURBS, beta-splines and thin-plate splines, just to mention a few. Some of them will be further presented.

### 2.2.1. Bezier Curves and Patches

Together with cubic Hermite curves and patches, Bezier curves and patches were among the earliest forms investigated and used in geometric modeling. Their relative simplicity and natural versatility make them good starting points for studying curves and surfaces [MEM97]. The *Bezier curve* for  $n+1$  control points can be described recursively with:

$$p_i^k(u) = (1-u)p_i^{k-1}(u) + up_{i+1}^{k-1}(u), \quad \begin{cases} k = 1..n \\ i = 0, \dots, n-k \end{cases} \quad (3)$$

where  $p_i^k$  are the intermediate control points obtained after the linear interpolation has been applied  $k$  times and  $p_i^0$  are the initial control points. A Bezier curve can also be described with an algebraic formula, called the *Bernstein form*:

$$p(u) = \sum_{i=0}^n B_i^n(u) p_i, \quad B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}, \quad B_i^n(u) \in [0, 1], \quad \text{when } u \in [0, 1], \quad \sum_{i=0}^n B_i^n(u) = 1 \quad (4)$$

### Advantages

The entire Bezier curve will be located in the convex hull of the control points. This is a useful property when computing a bounding area or volume for the curve. The Bezier control points give a good indication of the shape of the corresponding curve [HLP93], which can be adjusted, just by moving these points. Moreover, instead of computing points on a Bezier curves, and then rotating the curve, the control points can be rotated first, and then the points on the curve can be computed. This is much faster than doing the opposite. Bezier curves are the easiest of the curve forms to subdivide [MEM97].

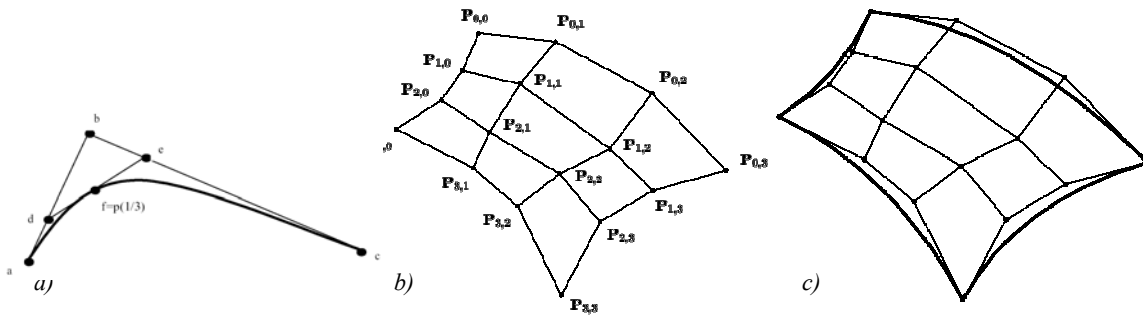


Figure 1. (a) Bezier Curve. (b) Bezier control points and (c) the corresponding patch

### Disadvantages

More control over the curve means increasing the degree of the polynomial and making the evaluation more expensive [HLP93]. Moreover, Bezier points are global in nature. Each point has an effect on the entire parameter interval, such that changes in it induce a change in the entire curve [AEP03, MEM97]. Bezier curves do not have many degrees of freedom and also, not every curve can be described by a Bezier curve [WFH02]. Last but not least, a Bezier curve is not invariant under perspective transformation. Another alternative to solve some of the above-mentioned problems is to use a *rational Bezier curve*.

$$p(u) = \frac{\sum_{i=0}^n w_i B_i^n(u) p_i}{\sum_{i=0}^n w_i B_i^n(u)} \quad (5)$$

For a rational Bezier curve, the same set of control points can produce many different shapes, interpolating the same endpoints and with the same end tangents. Changing one or more of the weights changes the shape of the curve. This means that a rational cubic Bezier curve has four more degrees of freedom than the non-rational cubic counterpart [MEM97]. The rational Bezier curve is invariant under a perspective transformation. If its control points are subjected to this

transformation, then all other points are produced as usual from the transformed control points. The resulting curve is the accurate perspective transformation of the original. Moreover, using the rational Bezier formulation we can accurately represent conic curves, which the non-rational counterpart cannot do. The main justification for using rational curves in practice is their ability to represent conics and circles exactly. Their use is mainly limited for that purpose.

The Bezier curve can be extended to a patch by introducing one more parameter in eq. (3). The surface representations exhibit advantages and disadvantages similar to those of the corresponding curves.

### 2.2.2. Cubic Hermite Curves and Patches

The cubic Hermite curve is defined by starting and ending points ( $p$ ), and starting and ending tangents ( $m$ ) [AMH02], as in Fig. 2.

The cardinal form of a Hermite curve [FRN02] is:

$$p(u) = p_0 H_0^3(u) + m_0 H_1^3(u) + m_1 H_2^3(u) + p_1 H_3^3(u) \quad (6)$$

where the cubic Hermite polynomials are given by:

$$H_0^3(u) = B_0^3(u) + B_1^3(u), H_1^3(u) = 1/3 B_1^3(u), H_2^3(u) = -1/3 B_2^3(u), H_3^3(u) = B_2^3(u) + B_3^3(u) \quad (7)$$

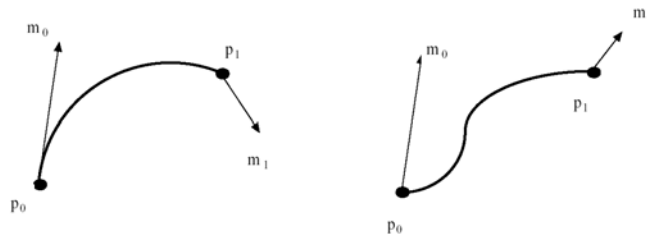


Figure 2. (a) Hermite curves (redrawn from [AMH02])

The curve interpolates all its control points and is easy to subdivide [MEM97]. It is invariant under rotation, scaling and translation, but is not invariant under perspective projection [FDF96]. When interpolating between more than two points using Hermite curves, a mean to control the shared tangents is needed. *Kochanek-Bartels curves* [KBT84] give the user three intuitive parameters to control shared tangents – bias (influences the direction of the tangent), tension (modifies the length of the tangent vector) and continuity (controls behaviour at the joints).

Like the Bezier curve, a Hermite curve can be extended to a Hermite patch by adding one more parameter in eq. (6).

### 2.2.3. Splines

The other major category of curves and surfaces used in computer graphics are splines. Though many different formulations of spline exist (e.g. natural B-splines, beta-splines [FDF96, MEM97], thin-plate splines [WSS, TBS99]), the most commonly used are B-splines and NURBS (Non-Uniform Rational B-splines). A *b-spline curve* is a generalization of a Bezier curve. Given  $n + 1$  control points  $c_0, c_1, \dots, c_n$  and a knot vector  $U = \{ u_0, u_1, \dots, u_m \}$ , the B-spline curve of degree  $p$  defined by these control points and knot vector  $U$  [MEM97] is:

$$p(u) = \sum_{i=0}^n N_{i,p}(u) c_i N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u \leq u_{i+1} \\ 0 & \text{otherwise} \end{cases}, \quad N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+1} - u}{u_{i+1} - u_{i+1}} N_{i+1,p-1}(u) \quad (8)$$

where  $N_{i,p}(u)$  are called B-spline basis functions of degree  $p$ .

#### *Advantages*

B-spline curves satisfy all the important properties that Bezier curves have. However, they provide more control flexibility than Bezier curves can do. They are defined locally and changes in one of the points of the associated control polygon affect the curve locally. It is also possible to add new control points without increasing the polynomial degree, and even to change the order of continuity between neighbouring spline segments by choosing the control points appropriately [HLP93]. A valuable property is that they are invariant under affine transformations [MEM97]. Models generated with splines theoretically offer infinite resolution [FLM99].

#### *Disadvantages*

B-spline curves require more information (the degree of the curve and a knot vector) and a more complex theory than Bezier curves. From computer graphics perspective, B-splines cannot be used to model details. They cannot accurately represent small bumps, wrinkles, detail branching or complex objects with many protrusions [FLM99]. Moreover, they cannot represent conic sections (circles, ellipses, spheres, cylinders).

#### *Applications*

While not suitable for modeling details and protrusions, splines are to be used when exact curves are needed to describe the physical surface of an object or for prototyping low-detail models.

A *NURBS (Non-Uniform Rational B-Spline) curve* of degree  $m$  is given by:

$$p(u) = \frac{\sum_{i=0}^n w_i N_i^m(u) p_i}{\sum_{i=0}^n w_i N_i^m(u)}, \quad N_i^m(u) = \frac{u - u_i}{u_{i+m} - u_i} N_i^{m-1}(u) + \frac{u_{i+m+1} - u}{u_{i+m+1} - u_{i+1}} N_{i+1}^{m-1}(u), \quad m \geq 1, i = 0..n \quad (9)$$

where

$$N_i^0(u) = \begin{cases} 1 & \text{for } u_i \leq u \leq u_{i+1} \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

is the normalized B-Spline basis function defined recursively,  $p_i$  are control points for  $i = 0..n$ ,  $w_i$  are weights and  $U = \{ u_0, u_1, \dots, u_m \}$  is the knot vector. The NURBS curve can also be extended to a NURBS surface.

#### *Advantages*

The NURBS curves and surfaces can be easily editable by adjusting the knots. The resulting models are efficient in terms of memory use. Very little memory is needed to store quite complex curve structures [GDM]. NURBS provide theoretically infinite resolution. They allow addition of points to the curve without altering its shape. Unlike parametric curves and natural B-splines, they are able to match conic sections with a minimum of values. The perspective division embedded in the construction of NURBS curves ensures that they are handled correctly in perspective views [AEP03].

#### *Disadvantages*

Like B-splines, NURBS are not good for modelling details, holes and branches. A great number of steps is required between constructing a curve network and actually getting the visual feedback of a renderable object. It is difficult to see which control point will deform which curve while adjusting control points [FLM99].

#### *Applications*

NURBS are well suited to exact curves modelling and as a starting point for polygonal modeling.

### **2.3. Implicit Surfaces**

Instead of using parameters such as  $(u, v)$  to explicitly describe a point on the surface, an implicit function of the form:

$$f(x, y, z) = f(p) = 0 \quad (11)$$

can be used. In other words, a point  $p$  is on the implicit surface if the result is 0 when the coordinates of the point are inserted into the implicit function  $f$ .

### *Advantages*

A main advantage of implicit representation over its parametric counterpart is that they comprise more information about their interior and exterior. Implicit surfaces are inherently manifold. They can be collided and deformed [YTG99]. Moreover for the same polynomial of degree  $n$ , implicit algebraic surfaces have more degrees of freedom, compared with rational parametric surface of the same degree. Hence, implicit algebraic surfaces are more flexible to approximate a complicated structure with fewer number of pieces or to achieve a higher order of smoothness [BCX95].

### *Disadvantages*

Despite their many advantages, implicit surfaces are difficult to render efficiently. Today's real time graphics systems are heavily optimized for rendering triangles, therefore an implicit surface should be converted to a mesh of triangles before being rendered [KSA01]. The representation being multivalued may cause the real zero contour surface to have multiple sheet, self-interactions and several other undesirable singularities [YTG99]. Their use is limited by the difficulty in identifying specific locations on a surface.

## **2.3.1. Quadrics and Superquadrics**

Quadrics and superquadrics are the most important case of implicit functions. The class includes ellipsoids, paraboloids, and hyperboloids [HSF90]. A quadric is described by the equation:

$$q(x, y, z) = a_{11}x^2 + 2a_{12}xy + a_{22}y^2 + a_{33}z^2 + 2a_{23}yz + 2a_{13}xz + b_1x + b_2y + b_3z + c = 0 \quad (12)$$

Quadrics are a convenient representation for spheres, ellipsoids and cylinders. They are used in molecular modeling and solid modeling systems [FDF96].

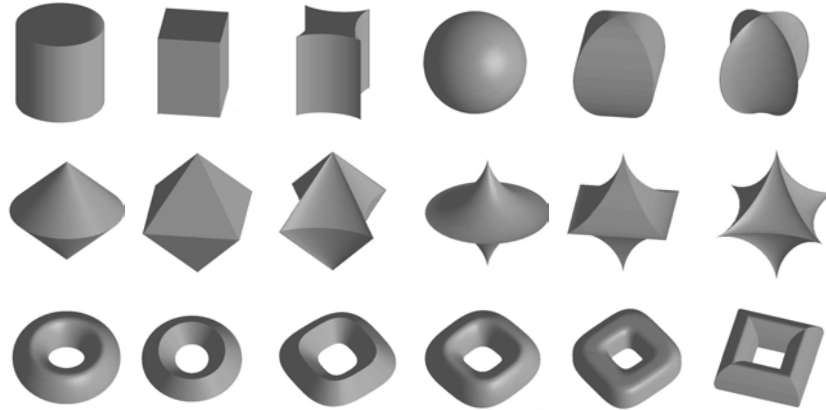
The *superquadrics* are a mathematical generalization of quadrics, which include an interesting class of shapes within a single framework: spheres, ellipsoids, and objects that arbitrarily closely look like prisms, cylinders, stars or toruses (Fig. 3). A superquadric is described by the following implicit function:

$$\left( \left( \frac{x}{a_1} \right)^{2/\epsilon_2} + \left( \frac{y}{a_2} \right)^{2/\epsilon_2} \right)^{\epsilon_2/\epsilon_1} + \left( \frac{z}{a_3} \right)^{2/\epsilon_1} - 1 = 0 \quad (13)$$

where  $a_1, a_2, a_3$  define the superquadric size in  $x, y, z$  coordinates and  $\varepsilon_1$  and  $\varepsilon_2$  are the squareness parameters in latitude and longitude plane, respectively.

### *Advantages*

The shape of the superquadrics is controlled by simple parameters ( $\varepsilon_1$  and  $\varepsilon_2$ ). Their modeling capabilities can be further enhanced by deforming them in different ways, by tapering, bending or twisting (presented in 3.4) [BAH84, SBT90].



**Figure 3. Superquadrics – Superellipsoids and supertoruses**

### *Disadvantages*

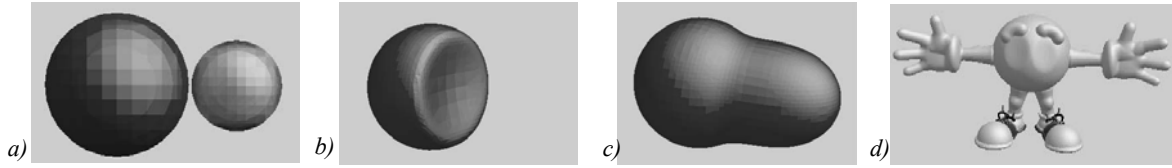
However, superquadrics lack the descriptive ability to effectively capture local surface shape changes on freeform objects [CFT01]. They have problems in modeling linear objects, mechanical organics and models that require close-ups. The representation is nonunique, which might cause problems in object matching with a database of models [SBT90].

### *Applications*

Superquadrics are well suited for modeling complex organics and their capability of being easily deformed makes them a good choice for animation. They have been used to recognize a large class of objects [30, SBT90] and to build composite parts using unions of several superquadrics.

## **2.3.2. Metaballs**

Another case of implicit functions is the metaballs. *Metaballs* consist of a series of components, spherical in shape, each associated with a size and an “attractive” or a “repulsive” strength. The final shape of the metaball is computed based on the position, size, and strength of each component. The center of the metaball has the maximum density value, and the density decreases outwards. Metaballs are spheres that blend together to form a single object [FLM99].



**Figure 4. (a) Metaballs. (b,c) Negative and positive metaball influence. (d) Metaball model (reproduced from [FLM99])**

### *Advantages*

Metaballs editing is easy because of the ability to modify the spherical shape of the metaball (they stretch, shear, bend and twist). They are well suited for modeling complex, highly detailed organics. Metaball modeling is one of the fastest modeling methods available. It is at the same time well suited for animation, in modeling water effects and liquid metals [FLM99].

### *Disadvantages*

The determination of the correct placement of the metaballs is a challenge. Quite often the metaballs need to be separated by a gap to produce the proper effect. Metaballs models can result in disorganized meshes and models tend to look as if they are made of spheres. Moreover, small details are very difficult to obtain with metaballs.

### *Applications*

While well suited for detailed creatures and characters, natural organics and rapid prototyping, metaballs are not to be used for linear models, mechanical organics or models that require close-ups [FLM99].

## **3. Solid Models**

Solid modeling describes the volume of space occupied by a solid. It differs from surface modeling in several ways. Solid objects are, unlike surfaces, closed (contain their boundary). While it is possible to decompose a solid object into smaller patches, it is not possible to convert a surface into a 'real' 3D object. Moreover, unlike surfaces, a solid object has an inside and an outside. The solid representation schemes can be categorized into implicit, enumerative, boundary schemes [SFH02] and deformational schemes.

### **3.1. Implicit Schemes**

*Implicit* representations give rules for testing which points belong to an object and which do not. The most important representations comprised in this category are the pure primitive instancing schemes, constructive solid geometry and sweep presentations.

### 3.1.1. Pure Primitive Instancing

In a pure primitive instancing scheme each object family is called a *generic primitive*, and individual objects within a family are called *primitive instances* [RGA80]. A generic primitive is a 3D solid shape specific to the application area [MEM97] and represented by fixed-length tuple of values [RGA80], such as ('PARALLELEPIPED',  $l$ ,  $w$ ), where 'PARALLELEPIPED' represents the name of the family and  $l$  is the length of the sides and  $w$  the width. The primitive instance is a linear transformation of an existing generic primitive. The instances are not usually combined to form more complex shapes, though no theoretical barriers prevent it.

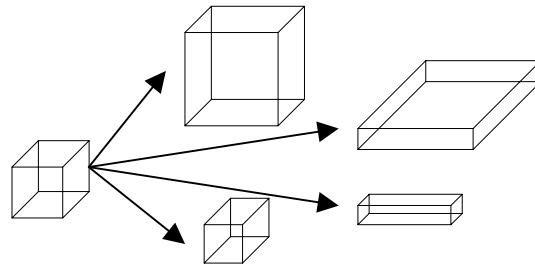


Figure 5. Instances of the unit cube (redrawn from [MEM97])

#### *Advantages*

Pure primitive instancing schemes are easy to validate, unambiguous, unique, concise, and easy to use.

#### *Disadvantages*

One disadvantage is the lack of means for combining instances to create new and more complex objects. Furthermore, it is difficult to write algorithms for computing properties of represented solids. A considerable amount of family-specific knowledge must be built into the algorithms, and therefore each object family must be treated as a special case, allowing no uniform overall treatment [RGA80].

#### *Applications*

Often used for relatively complex objects, in manufacturing world, such as gears and bolts, that are tedious to define in terms of boolean operations [FDF96].

### 3.1.2. Constructive Solid Geometry

*Constructive Solid Geometry (CSG)* is a set of modeling methods to define complex solid using closed primitives, rigid body motions and set operations [RGA80, SFH02]. Rigid body motions are described in terms of TRPY convention by:

$$\begin{aligned}
X &= a_1 x \cos \theta \cos \varphi + a_1 y (\cos \theta \sin \varphi \sin \psi - \sin \theta \cos \psi) + a_1 z (\cos \theta \sin \varphi \cos \psi + \sin \theta \sin \psi) + x_T \\
Y &= a_2 x \sin \theta \cos \varphi + a_2 y (\sin \theta \sin \varphi \sin \psi + \cos \theta \cos \psi) + a_2 z (\sin \theta \sin \varphi \cos \psi - \cos \theta \sin \psi) + y_T \\
Z &= -a_3 x \sin \varphi + a_3 y \cos \varphi \sin \psi + a_3 z \cos \varphi \cos \psi + z_T
\end{aligned}
\tag{14}$$

where  $X, Y, Z$  are the coordinates of the points in the transformed solid,  $x, y, z$  the coordinates of points in the original one,  $\theta$  defines the rotation around  $z$ ,  $\varphi$  the rotation around  $y$ ,  $\psi$  the rotation around  $x$ ,  $x_T, y_T, z_T$  describe the translation and  $a_1, a_2, a_3$  define the scaling factor along  $x, y, z$  respectively. The equations above permit the determination of the transformed solid coordinates for pure transformations and combinations of transformations as well.

The set operations used are union, difference and intersection. A regularized union, intersection or difference ensures that the new object is a closed one, which is a must in solid modeling.

The CGS representation is an ordered binary tree with nonterminal nodes operators and terminal nodes representing either primitive or transformation leaves which contain the arguments of rigid motions. The operators can either be rigid motions or regularized union, intersection, or difference.

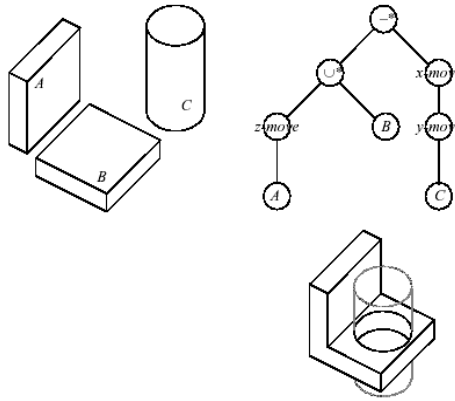


Figure 6. A CGS model and its tree structure [PSM]

An object therefore exists as a tree structure (Fig. 6), which is evaluated during rendering [BGS97].

### Advantages

The CSG representation is unambiguous and concise. The set operations are computationally convenient and the rigid motion parameters offer a natural control of the solid's shape, position and orientation. The relatively simple and robust data structures and the elegant algorithms contributed to the popularity of CSG in computer modeling [SFH02].

### *Disadvantages*

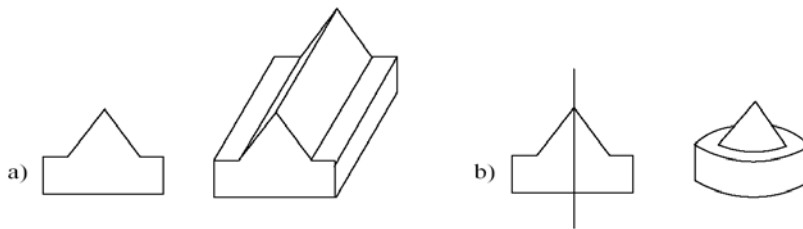
The objects that can be modeled with CSG are fairly limited compared to other methods. The lack of explicit representation and parameterization of the interior of the solid object is another drawback, actually associated with all implicit models. Comparison of solids is also problematic, since the CSG representation is non-unique. Moreover, the display is usually expensive.

### *Applications*

CSG is used in those applications where simple geometric objects are desired, or where mathematical accuracy is important.

## **3.1.3. Sweep Presentation**

Sweep presentations are constructed by moving a curve, surface, or solid along some path. The locus of points generated by this process defines a new 2D or 3D object. A translational sweep (extrusion) is defined by moving a planar curve or planar shape along a straight line normal to the plane of the curve, the former generating a surface and the latter a solid (Fig.7a). A rotational sweep is defined by rotating a planar curve or shape with finite area about an axis (Fig. 7b). A general sweep is one whose generating shape follows some arbitrary curved path, and which itself may change size, shape and orientation [MEM97].



**Figure 7. (a) Translational sweep presentation, (b) Rotational sweep presentation**

### *Advantages*

Sweep representations are unambiguous representations, simple to understand and use. Only a small data set is needed to specify the shape. They are mathematically concise. Occlusions can be handled by storing a sweep presentation of the free space instead of the solid objects.

### *Disadvantages*

Their use is restricted to objects with rotational or translational symmetry. Moreover, the sweep presentation is not unique. This problem can be solved using *generalized cylinders*, a combination of sweep presentation and contours. This definition ensures that the representation is unique over a class of transformations [RGA80].

### *Applications*

Sweep presentations can accurately represent a large class of engineering and manufacturing objects and processes. They proved to be practical and efficient for modeling constant cross-section mechanical parts and for simulating and analyzing material-removal operations in manufacturing. They can also be used to detect potential interference between parts and mechanisms [MEM97]. A moving object collides with a fixed object if the swept volume of the moving object intersects the fixed object.

## **3.2. Enumerative Schemes**

A more direct way to define which points belong to the solid and which do not is to enumerate the points by an explicit parametric rule.

### **3.2.1. Parametric Solids**

The simplest and most direct mathematical approach to model a solid is by using continuous, three-parameter, single-valued functions of the form  $x=x(u,v,w)$ ,  $y=y(u,v,w)$ ,  $z=z(u,v,w)$ , where  $u, v, w \in [0,1]$ . These functions define the coordinates of the set of points comprising the interior and exterior of the solid. This representation is called *parametric solid*, or *hyperpatch* [MEM97]. It inherits the advantages and disadvantages of parametric curves and surfaces. It is unambiguous, unique and concise.

### **3.2.2. Space-Partitioning Solids**

An object can be decomposed in parts until its parts are describable. The solid will be thus a union of cells into which the object is divided. This process is known as *cell decomposition*. Space-partitioning representation is a special case of cell decomposition, where the shape of the cell is a cube or parallelepiped (*voxel*) and the cells are fixed in a spatial grid. As the size of the cube decreases, this method approaches the representation of a solid body as a set of contiguous points in space [MEM97].

#### **Voxels**

The simplest and convenient way to represent a solid is as an ordered list of voxels occupied by the solid, called *spatial array* [RGA80].

#### *Advantages*

A spatial array is an easy to validate, unambiguous and unique representation. The complexity of modeled objects is the same, in that the representation uses the same amount of

computer memory no matter what physical size it is representing. Set operations are easy to implement and the representation is affine invariant.

#### *Disadvantages*

However, the voxel representations can be quite verbose and require large amounts of data storage. The representation is not very accurate and is not concise.

The display is usually extensive.



Figure 8. An object and its voxel representation (reproduced from [OCR])

#### *Applications*

Spatial arrays have been successfully used in certain architectural applications where buildings are sufficiently modular or in tomography where irregular biological objects are modeled approximately by polyhedra [RGA80].

#### **Octrees**

*Octrees* are an approach to 3D object representation based on the recursive subdivision of an object array into octants.

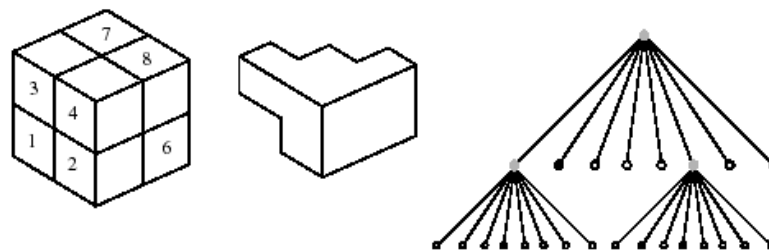


Figure 9. A model and an octree representation (reproduced from [PSM])

This process is represented by a tree of degree 8, in which the root node represents the entire object with octants labeled. The leaf nodes correspond to those cubes of the array for which no further subdivision is necessary [SHS84]. Leaf nodes are black ('full') or white ('empty'), depending on whether their corresponding cubes are entirely within or outside of the object, respectively. All nonleaf nodes are gray.

#### *Advantages*

The octree representation is easy to validate. Set operations are easy to implement due to the simplicity of the tree structure. The complexity of modeled objects is the same, in that the

representation uses the same amount of computer memory no matter what physical size it is representing. Because usually the cubes are larger in size than the voxels, the octree is an economical hierarchical representation. Octrees are able to provide a representation for many types of objects and their precision is determined only by the size of the smallest cube [FDF96]. If all leaves of a particular node are outside the field of view of the viewpoint, that node can be discarded from rendering. This is a fast and effective way of speeding up the rendering engine.

*Disadvantages*

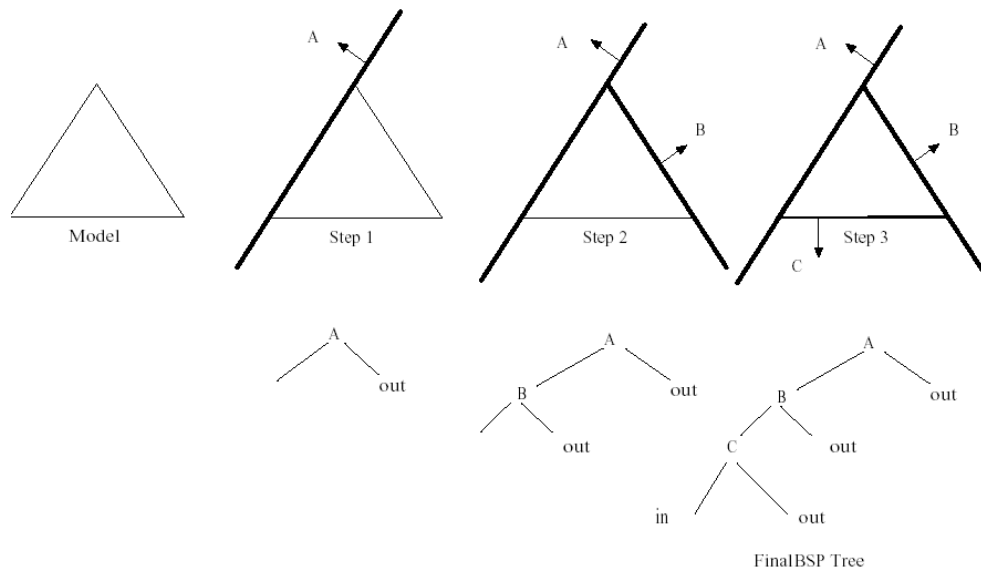
The octree representation is neither concise, nor affine invariant. The display is expensive and high accuracy implies high storage. Using an octree representation, an object can only be approximated, and not fully represented. This limitation restricts the usability of octrees in the existing graphics systems [FDF96].

*Applications*

Octrees are excellent at modeling 3D spaces, which contain volumes that are highly connected. They can also be extended to model arbitrarily large and growing surfaces, which do not have a priori bounds [LDT91].

**Binary Space-Partitioning Trees**

A *binary space-partitioning (BSP)* tree is a recursive, hierarchical subdivision of the object space. Each subspace is partitioned by hyperplanes that intersects the interior of that subspace and the two resulting subspaces are labeled “in and “out” [FDF96]. These two subspaces can be further partitioned, as depicted in Fig. 10.



**Figure 10. A model and the steps in the BSP tree construction**

### *Advantages*

The BSP representation is an easy to validate, unambiguous, simple and elegant representation. It is affine invariant and set operations are easy to be performed. The display is efficient.

### *Disadvantages*

On the other hand, the BSP tree is a non-unique and not concise representation, and its validity is computationally expensive to establish. The representation does not ensure that the result is a bounded solid (e.g. the 3D BSP tree consisting of a single internal node, with “in” and “out” children defines an object that is a half-space bounded by only one plane [FDF96]). In order to add or remove cells from the described solid, a part of BSP tree needs to be reconstructed. This property limits the use of BSP trees to static models.

### *Applications*

BSP trees, as the octrees, are an efficient scheme to be used when a fast rendering is needed. If all leaves of a particular node are outside the field of view of the viewpoint, that node can be discarded from rendering. This is a fast and effective way of speeding up the rendering engine. The tree can also be used for a number of other effects, such as shadows.

## **3.3. Boundary Schemes**

Another idea to represent a solid object is to describe the surfaces enclosing that solid. Boundary models are complete representations of a solid as an organized collection of surfaces. The solid is thus a union of faces (surfaces), bounded by edges (curves), which in turn are bounded by vertices (points) [MEM97], as presented in Fig. 11.

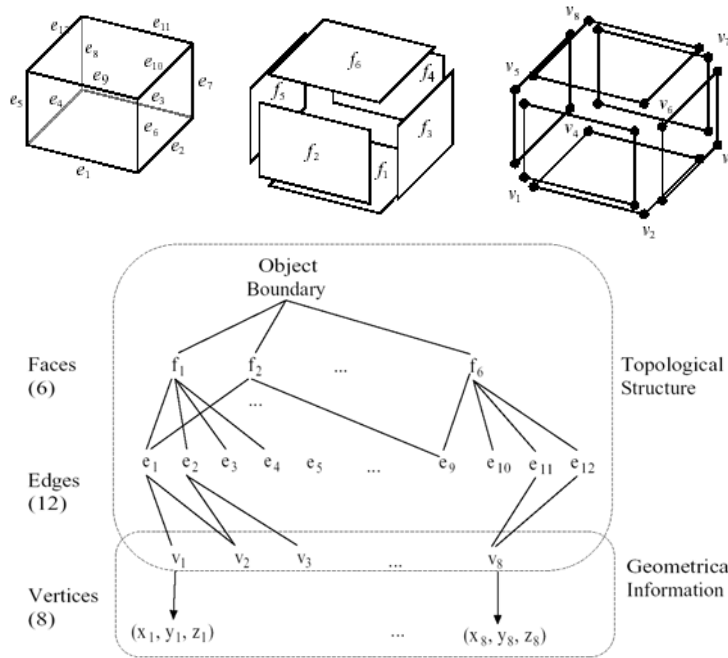
### *Advantages*

The boundary representation is unambiguous. It is as powerful as, or even more powerful than CSG schemes, given that the appropriate primitive surfaces are chosen. The main virtue of boundary representations lies in the ready availability of representations for faces, edges, and the relations between them. Moreover, boundary representation can be much smaller than most 3D cellular representations of the same object.

### *Disadvantages*

Boundary representations are generally not unique and are not concise (usually an order of magnitude longer than corresponding CSG representations). They are not trivial to construct and maintain. There are two types of conditions for validity that a boundary scheme has to

satisfy: topological (realizable and orientable 2-manifolds) and geometrical. Geometrical conditions are usually difficult and computationally expensive to check.



**Figure 11. A cube with marked faces, edges and vertices (reproduced from [PSM]) and its boundary representation**

### Applications

The data on faces, edges and relations between them are important for generating line drawings and graphic displays and for supporting graphic interaction [RGA80]. Thus it is not surprising that boundary representations have been used extensively in computer graphics.

## 3.4. Deformational Schemes

Deformations can be seen as an extension of affine transformations and set operations for solid object modeling. There are two categories of deformations. A *locally specified deformation* modifies only a set of points corresponding to a sub-region of the object surface (the tangent space of the solid is modified [BAH84]). A *globally specified deformation*, in contrast, affects an object as a whole, by explicitly modifying the global coordinates of all solid's points in space [BAH84]. Such transformations include global tapering, global twisting and global bending. Only the latter will be further detailed. Deformations can also be combined in the form of hierarchical structures. Thus they increase the range and complexity of solids that can be modeled.

### Global Linear Tapering

The *tapering* deformation can be thought of as a differential scaling, which only changes the length of two components (say  $y$  and  $z$ ), without changing the length of the third ( $x$ ). Thus, the global tapering along  $x$  is given by:

$$\begin{aligned} X &= x \\ Y &= f_y(x)y \\ Z &= f_z(x)z \end{aligned} \tag{15}$$

where  $X, Y, Z$  are the coordinates of the points in the deformed solid,  $x, y, z$  the coordinates of points in the original solid and  $f_y(x)$  and  $f_z(x)$  are the tapering functions along  $y$  and  $z$  axes. The tapering functions are piecewise linear functions which decrease as  $x$  increases [BAH84]. Tapering along  $y$  and  $z$  are obtained in the same manner.

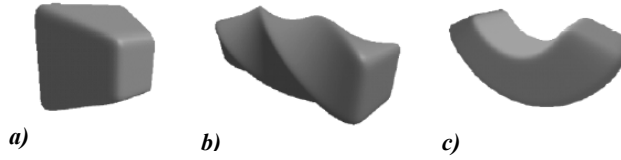


Figure 12. (a) Tapering, (b) Twisting, (c) Bending (reproduced from [SHJ00])

### Global Axial Twisting

A *twist* can be approximated as a differential rotation. Only two components are rotated as a function of height, while leaving the third one unaltered. The global twist around  $x$  is produced by the following set of equations:

$$\begin{aligned} X &= x \\ \theta &= f(x) \\ Y &= y \cos \theta - z \sin \theta \\ Z &= y \sin \theta + z \cos \theta \end{aligned} \tag{16}$$

where  $X, Y, Z, x, y, z$  are defined as before, and the twist proceeds along  $x$  at a rate of the derivative of  $f$ ,  $f'(x)$  radians per unit length [BAH84]. Twisting along  $y$  and  $z$  are obtained in the same manner.

### Global Linear Bending

Another deformation consists in simultaneously translating and rotating two components around the third. This deformation is called bending. A bend along  $x$  is described by eq. (17) [SHJ00] where  $X, Y, Z, x, y, z$  are defined as before, and  $k$  is the bending parameter ( $1/k$  gives the curvature of the bending). The bending along  $y$  and  $z$  are achieved in the same manner as for  $x$ .

$$\begin{aligned}
\theta &= ky \\
X &= x \\
Y &= -\sin\theta\left(z - \frac{1}{k}\right) \\
Z &= \cos\theta\left(z - \frac{1}{k}\right) + \frac{1}{k}
\end{aligned}
\tag{17}$$

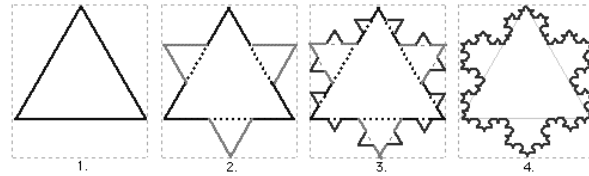
## 4. Procedural Models

Some irregular or complex objects with dynamic topologies, or objects that have no solid surface are difficult to be represented as a set of surface primitives or polyhedral models. Surface and polygonal models usually need explicitly storing of all the requisite data [BGS97]. Moreover, they have difficulty in combining computer graphics with physical laws. Although one can build and animate polygonal models of real-world objects, it is far more difficult to make these graphical objects act as solids and not penetrate one another [AEP03]. Thus, researchers in the domain looked for alternatives. One of the proposed solutions is procedural modeling.

Procedural models are generative processes that describe objects that can interact with external events to modify themselves [FDF96]. Each procedural model of a 3D object is described in terms of components and a procedure (algorithm) that shows how to generate the object and how to control its shape using these components. The necessary data is produced only when requested. The user can control how many primitives are produced and at which point in the generation process. Equally important is that procedural graphics provide an object-oriented approach to building models – an approach that should be of increasing importance in the future [AEP03]. The most common procedural models are fractals, graftals and particle systems.

### 4.1. Fractals and Graftals

The term “fractal” denotes “A geometric pattern that is repeated at ever smaller scales to produce irregular shapes and surfaces that cannot be represented by classical geometry. Fractals are used especially in computer modeling of irregular patterns and structures in nature”[AHD]. The concept is best illustrated using the ‘von Koch snowflake’ (Fig. 13). The snowflake is built by starting with an equilateral triangle, removing the inner third of each side, building another equilateral triangle at the location where the side was removed, and then repeating the process indefinitely [WSS].



**Figure 13. Von Koch Snowflake**

*Graftals* use deterministic processes to model more repetitive patterns [BGS97]. Parallel graph grammar languages (L-grammars), called by Smith [SRA84] ‘graftals’, were first used to describe the structure of certain plants. The shape of the plant is defined by a string of symbols constructed by a graftal grammar (Fig.14). A graftal grammar consists of an alphabet of symbols, a set of production rules and an axiom from which to begin construction.

A typical example is the grammar  $\{A,B, [, ], (, )\}$  and the two production rules [FDF96]:

$$A \rightarrow AA$$

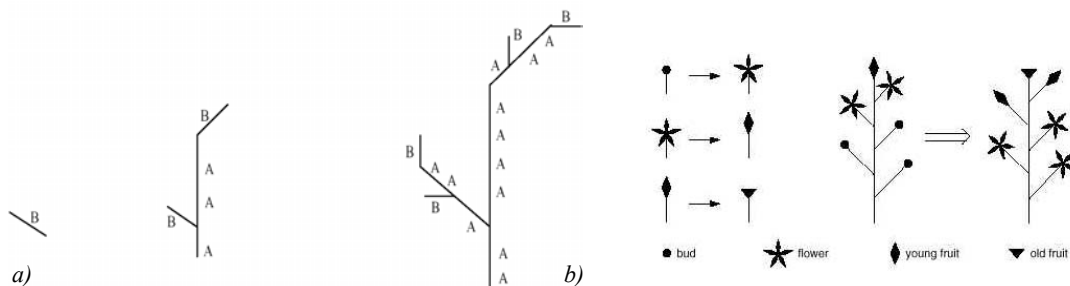
$$B \rightarrow A[B]AA(B)$$

The graftal is built by recursively feeding the axiom through the production rules. Each character of the input string is checked against the rule list to determine which character or string to replace it with in the output string. In this example, a 'A' in the input string becomes 'AA' in the output string. The axiom is applied in the same manner for 'B':

$$1st\ Recursion: A[B]AA(B)$$

$$2nd\ Recursion: AA[A[B]AA(B)]AAAA(A[B]AA(B))$$

This string is drawn as an image where each symbol is assigned a graphical operation; the square brackets become a left branch and parentheses a right branch (Fig 18a).



**Figure 14. (a) Tree representation of the first three words of the grammar (redrawn from [FDF96]). (b) Representation of a plant (reproduced from [PHM])**

### *Advantages*

Fractals and graftals create surfaces using an implicit model that produces data when requested. They save on storage space and object construction time and allow a terse

representation of a complex model. Graftals share with fractals the property that they offer infinite detail.

#### *Disadvantages*

Rendering can be difficult and slow.

#### *Applications*

*Fractal models* have been used to model natural objects that exhibit self-similarities: mountain landscapes with rivers [PHM], rocks, clouds and coastlines, planets and virtual universes [MFS01], and in the architectural domain [SNC00]. The plants models are the most spectacular examples of *graftals* modeling. However, the method has been used in other applications as well, such as architecture (city and street modeling) [PHM01]. Actually in any domain in which the objects being modelled exhibit sufficient regularity, there may be an opportunity to develop a grammar-based model.

## **4.2. Particle Systems**

A *particle system* is defined as a collection of simple primitive objects (particles) that evolve over time [FDF96, AMH02] and that are processed as a group in order to represent an object [MBG99]. The evolution is determined by applying certain algorithms, usually in form of probabilistic rules to the particles [FDF96, AMH02]. The characteristics of these objects, such as shape, size, position, color, and the lifetime of the particle itself, can be changed dynamically using a set of parameters. Stochastic processes that randomly select each particle's appearance and movement are constrained by these parameters. In general, each parameter specifies a range in which a particle's value must lie [RTS83]. If these parameters of the particles are coordinated, the collection of particles can represent an object.

#### *Advantages*

Advantages over classical surface-oriented techniques are presented in [RTS83, RTB85]. One advantage is that a particle is a much simpler primitive than a polygon, the simplest of the surface representations. Therefore, in the same amount of computation time one can process more of the basic primitives and produce a more complex image. Moreover, the model definition is procedural and stochastic. Therefore, obtaining a highly detailed model does not necessarily require a great deal of human design time as is often the case with existing surface-based systems. Because it is procedural, a particle system can adjust its level of detail to suit a specific set of viewing parameters. As with fractal surfaces, zooming in on a particle system can reveal more

and more detail. Last, but not least, particle systems can be used to model dynamic objects which are more difficult to represent with surface-based modeling techniques.

#### *Disadvantages*

Particles of changing sizes can lead to performance penalties. Moreover, particle systems are hard to integrate seamlessly into a complex scene [MBG99].

#### *Applications*

Several applications include modeling solid objects [AEP03] (a deformable solid is modeled as a 3D array of particles that are held together by springs). When the object is subjected to external forces, the particles move and their positions approximate the shape of the solid object), simulating fire [RTS83], smoke, fog, explosions [RTS83], fireworks [RTS83], water flows, trees [RTB85], grass [RTB85], whirling galaxies, flocking behaviour of birds and other natural phenomena. Particles are also a method of animation. In this context, the main exploited idea is that particle systems are provided with control means for creating, moving, changing, and deleting particles during their lifetime.

#### **References**

- [AHD] \*\*\* *The American Heritage Dictionary of the English Language*, <http://www.bartleby.com/61/>
- [HLL91] R. Hall, "Supporting Complexity and Conceptual Design in Modeling Tools", in *State of the Art in Computer Graphics: Visualisation and Modeling*, Eds. D.F. Rogers, and R.A. Earnshaw, 1991 – quotation from R. Sproull, keynote speech, SIGGRAPH 1990
- [BGS97] N. I. Badler, and A. S. Glassner, "3D Object Modeling", *SIGGRAPH 97 Introduction to Computer Graphics Course Notes*, 1997.
- [PTR03] E.M. Petriu, "Neural Networks for Measurement and Instrumentation in Virtual Environments," in *Neural Networks for Instrumentation, Measurement and Related Industrial Applications*, Eds. S. Ablameyko, L. Goras, M. Gori, V. Piuri, NATO Science Series, Series III: Computer and System Sciences vol. 185, IOS Press, 2003, pp. 273-290.
- [FDF96] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, *Computer Graphics. Principles and Practice*, Addison-Wesley, 1996
- [FLM99] B. Fleming, *3D Modeling & Surfacing*, Morgan Kaufmann, 1999
- [GDM] A. Goodman, Lecture Notes SCC308 – Computer Graphics, Deakin University, Australia, <http://www.deakin.edu.au/~agoodman/scc308/topic4.html>
- [AMH02] T. Akenine-Moller, and E. Haines, *Real-Time Rendering*, A K Peters, 2002
- [MEM97] M. E. Mortenson, *Geometric Modeling*, Second Edition, Wiley Computer Publishing, 1997
- [AEP03] E. Angel, *Interactive Computer Graphics: A Top-Down Approach Using OpenGL*, Third Edition, Pearson Education, 2003

- [HLP93] J. Hoschek, and D. Lasser, *Fundamentals of Computer Aided Geometric Design*, A K Peters, 1993
- [WFH02] H. J. Wolters, "Rational Techniques", in *Handbook of Computer Aided Geometric Design*, Eds. G. Farin, J. Hoschek, and M. S. Kim, Elsevier, 2002
- [FRN02] G. Farin, *Curves and Surfaces for CAGD. A Practical Guide*, Fifth edition, Morgan Kaufmann, 2002
- [KBT84] D. H. U. Kochanek, R. H. Bartels, "Interpolating Splines with Local Tension, Continuity, and Bias Control", *Proceedings of the 11<sup>th</sup> annual International Conference on Computer Graphics and Interactive Techniques*, 1984, pp. 33 - 41
- [WSS] *Eric Weisstein's World of Mathematics*. A Wolfram Web Resource, <http://mathworld.wolfram.com/>
- [TBS99] G. Turk, and J. F. O'Brien, "Shape Transformation Using Variational Implicit Surfaces", *SIGGRAPH'99*, pp. 335–342
- [EAN] \*\*\* "Easily Achieving NURBs Patch Continuity", <http://www.swanimator.com/tipstutorials/patchModelling/patchmodelling.htm>.
- [YTG99] G. Yngve, and G. Turk, "Creating Smooth Implicit Surfaces from Polygonal Meshes", *Technical Report GIT-GVU-99-42*, Georgia Institute of Technology, 1999.
- [BCX95] C.L. Bajaj, J. Chen, and G. Xu, "Modeling with Cubic A-Patches", *ACM Transactions on Computer Graphics*, 1995, vol.14, issue 2, pp. 103–133.
- [KSA01] T. Karkanis, A. James Stewart, "Curvature Dependent Triangulation of Implicit Surfaces", *IEEE Computer Graphics and Applications*, 2001, vol. 22, no.2, pp. 60-69.
- [HSF90] F. S. Hill, *Computer Graphics*, Macmillan, 1990
- [BAH84] A.H. Barr, "Local and Global Deformations of Solid Primitives", *Computer Graphics*, 1984, vol. 18, nr.3.
- [SBT90] F. Solina and R. Bajcsy, "Recovery of Parametric Models from Range Images: The Case for Superquadrics with Global Deformations", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990, vol.12, issue 2, pp. 131–147.
- [CFT01] R. J. Campbell, P. J. Flynn, "A Survey Of Freeform Object Representation and Recognition Techniques", *Computer Vision and Image Understanding*, 2001, vol. 81, no. 2, pp. 166-210.
- [RGA80] A. G. Requicha, "Representations for Rigid Solids: Theory, Methods, and Systems", *ACM Computing Surveys*, 1980, vol. 12, issue 4, pp. 437 - 464
- [SFH02] V. Shapiro, "Solid Modeling", in *Handbook of Computer Aided Geometric Design*, Eds. G. Farin, J. Hoschek, and M. S. Kim, Elsevier, 2002
- [OCR] \*\*\* "Object-Centered Representations", <http://www.netnam.vn/unescocourse/computervision/832.htm>
- [SHS84] H. Samet, "The Quadtree and Related Hierarchical Data Structures", *ACM Computing Surveys*, 1984, vol. 16, issue 2, pp.187-260.
- [PSM] \*\*\* "Point-Set Models of Solid", <http://weld.arc.cmu.edu/48-745/Lectures-handouts/>
- [LDT91] D. Libes, "Modeling Dynamic Surfaces with Octrees", *National Institute of Standards and Technology*, 1991

- [SHJ00] J. Sinnott, and T.L.J. Howard, "SQUIDS: Interactive Deformation of Superquadrics for Model Matching in Virtual Environments", *Proceedings Eurographics*, 2000, pp. 73-80.
- [SRA84] A. R. Smith, "Plants, Fractals, and Formal Languages", *Computer Graphics*, 1984, vol. 18, no. 3, pp. 1-9.
- [PHM] P. Prusinkiewicz, M. Hammel, R. Miech, and J. Hanan, "The Artificial Life of Plants", [http://www.siggraph.org/education/materials/HyperGraph/modeling/procedural\\_modeling/procedural\\_modeling.htm](http://www.siggraph.org/education/materials/HyperGraph/modeling/procedural_modeling/procedural_modeling.htm)
- [MFS01] F. K. Musgrave, "Fractal Models of Natural Phenomena", *SIGGRAPH 01 "Simulating Nature" Course*, [https://www.internal.pandromeda.com/engineering/musgrave/unsecure/S01\\_Course\\_Notes.html](https://www.internal.pandromeda.com/engineering/musgrave/unsecure/S01_Course_Notes.html)
- [RTS83] W.T. Reeves, "Particle Systems – A Technique for Modeling a Class of Fuzzy Objects", *ACM SIGGRAPH Computer Graphics*, 1983, vol. 17, no.3
- [SNC00] N. Sala, "Fractal Models in Architecture: A Case of Study", *Proceedings International Conference on "Mathematics for Living"*, 2000, pp. 266-272
- [PHM01] Y. I. H. Parish, and P. Müller, "Procedural Modeling of Aities", *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001
- [MBG99] T. McReynolds, D. Blythe, B. Grantham, and S. Nelson, *Siggraph 99 Advanced Graphics Programming Techniques Using OpenGL*, Course Notes, <http://www.opengl.org/developers/code/sig99/advanced99/notes/node335.html>
- [RTB85] W. T. Reeves, R. Blau, "Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems", *ACM SIGGRAPH Computer Graphics*, 1985, vol.19 nr.3, p.313-322.