An Overview of Implicit Surfaces

Agata Opalach
Steve Maddock

Department of Computer Science, The University of Sheffield,
Regent Court, 211 Portobello Street, Sheffield, S1 4DP, UK
Tel: +44 114 282 5577 Fax: +44 114 278 0972
E-mail: a.opalach@dcs.shef.ac.uk

Abstract

This tutorial paper gives an overview of the area of implicit surfaces. The history of the technique is presented and various approaches categorised and compared. Existing problems with modelling, rendering and animation of implicit surfaces are discussed and available solutions evaluated. The paper provides a comprehensive introduction for novice and experienced users of implicit surfaces.

Keywords: Implicit Surfaces, Deformable Objects, Modelling Techniques, Animation Control, Review

1. Introduction

The most common methods of modelling the surface of an object are to explicitly represent it as a set of polygons or parametric patches. Many Computer-Aided Design (CAD) and computer graphics systems utilise these modelling techniques. However, they have their limitations, especially when smooth, deformable objects need to be represented and animated. Then, it is time for implicit surfaces to step into the limelight. As their name suggests the surface of an object is not modelled explicitly. Instead, equations are used to represent the surface.

Consider the equation for a circle: $x^2 + y^2 - r^2 = 0$. This describes the infinite number of (x,y) pairs that lie on the circumference of a circle of radius r with its centre positioned at the origin of a 2D Cartesian coordinate system. The circle circumference is implicitly represented by the equation. The same thing can be done in 3D where we can use $x^2 + y^2 + z^2 - r^2 = 0$ to represent all the (x,y,z) points on the surface of a sphere. Correspondingly, any other (x,y,z) point will give a non-zero result when inserted into the equation. This result can be used to trivially determine whether the point is inside or outside an implicit surface. The equation of the sphere can be rewritten as

$$F(x, y, z) - Iso = 0 \tag{1}$$

Now, by changing the value of *Iso*, different surfaces (depending on the function F) can be visualised by picking out all those points (x, y, z) where the equation gives a value of zero. Each such surface is called an *iso-surface* (or contour surface in analogy with geographical contours on 2D maps) and the function F is called the *scalar field function*. This is because for each point (x, y, z) the function gives a scalar value. Different functions F can be used to describe the field around a point and more than one point source can be specified with the resulting surface being a blend of the simpler surfaces surrounding each point. Complex surfaces can be modelled by using relatively few points. An often used analogy is with heat sources. A set of point heat sources each emit a spherical heat field. When placed in close proximity to each other, they create a more complex heat distribution surface describing all those points with the same scalar heat value. An important point that should be noted here is that it is easy to guess the approximate shape of this surface simply by overlaying each individual spherical field.

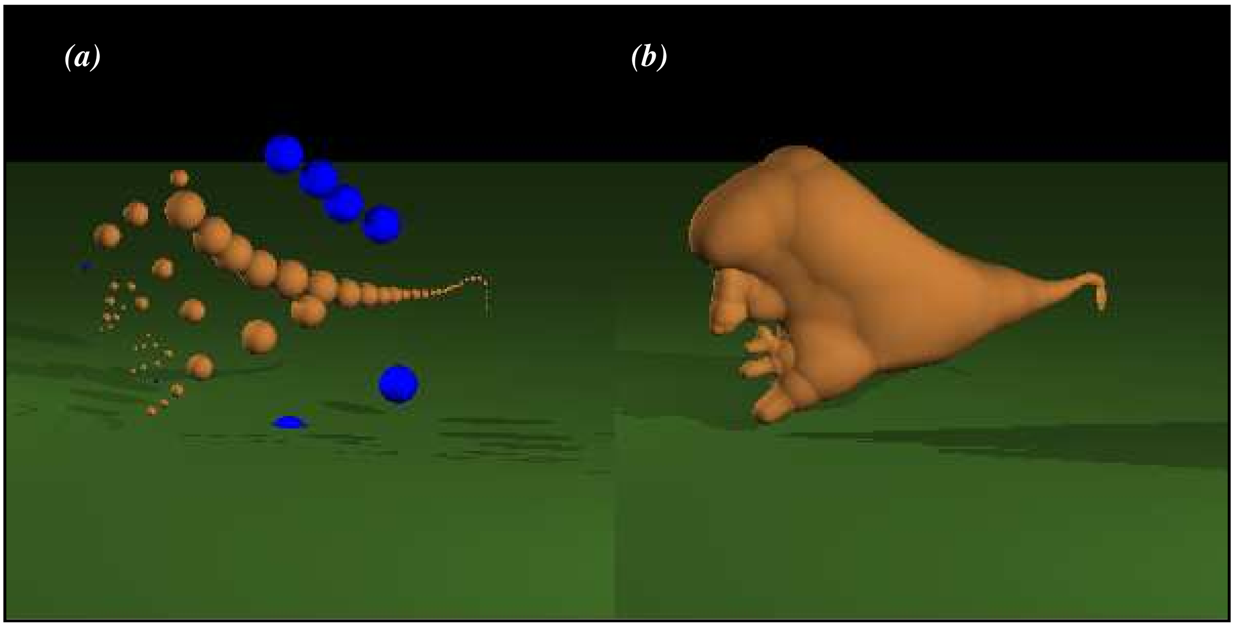

Figure 1. *An implicit surface inspired by Salvador Dalí. Brown (light) spheres represent positive primitives, blue (dark) spheres represent negative primitives, the radius of each sphere is proportional to the radius of influence of the corresponding primitive.*

Figure 1 shows an interesting implicit surface, inspired by Salvador Dalí's *Dream*. The head (Figure 1b) was modelled using 66 primitives, pictured as spheres in Figure 1a.

The types of functions that have been used to model implicit surfaces and how they are blended will be covered in more detail in section Section 3. However, first some history.

2. History and applications

As with many other techniques, it was Blinn who first introduced implicit surfaces into computer graphics when he used them to visualise electron density fields using *blobby molecules* [2]. At the same time Omura and his computer graphics research group in Japan developed *metaballs* [25]. The general principle of this technique is to model an equi-potential surface around point charges in a 3D space. Then *soft objects* were introduced in Canada [40, 41]. They are modelled as keypoints with scalar fields defined around them. More complex field functions, e.g. ellipsoids, superellipsoids or negative primitives, have also been suggested for the above approaches. The technique was later generalised by Bloomenthal for arbitrary skeletal elements, e.g. curves or polygons [4, 5] and has found many practical applications in both the commercial and the academic worlds.

The academic use of implicit surfaces includes a wide variety of applications. For instance, Tatsumi *et al.* present the use of metaballs in the process of modelling a certain type of cells in a cod's liver [33], Fujita *et al.* represent splashing water using metaballs [10], Muraki uses a blobby model for volumetric shape description (including human face modelling) [23], Max and Wyvill render corals as soft objects [20], Payne and Toga model a rat's brain using distance fields [29], Reed and Wyvill use soft objects to visualise lightning [30] and in Japan, metaballs have been widely used for modelling and animation of shapes as complex as human anatomy parts [13].

Implicit surfaces have also been appreciated in the fields of computer simulation and physically based animation. They have been used to visualise deformable material modelled as particles [22, 35] or mass points [34]. Modelling collision detection and response with a precise contact surface in a physically based environment has also been simulated [11]. Finally, realistic simulation of highly deformable, amorphous material (e.g. mud or dough) capable of fractures, fusion and maintaining nearly constant volume has been produced [7, 9].

When used in a keyframe animation environment, implicit surfaces offer the advantage of guaranteed smoothly blended and continuous surface. Metamorphosis can also be performed by interpolating between matching parts of two objects [44]. Successful animations like letters descending a staircase [46], a soft train rubbing a bank [47] or a blobby guitarist emerging from sparkling toothpaste [1] are very successful at conveying the story line to the audience.

Implicit surfaces can also serve as skin covering an articulated figure built by imposing a hierarchy on primitives. [26] describes the ABC of using this idea in animation: maintaining Appearance, preventing unwanted Blending and preserving Coherence. Some interesting animation effects can be achieved using this approach, including automatic traditional animation tricks, e.g. squash and stretch, follow through or exaggeration [27]. The balance between tedious manual motion specification and automatically generated movement has to be maintained [28].

In the commercial world, the metaball technology [13] proves most successful. The development of MetaEditor simplified the process of modelling with metaballs. Since then, numerous companies have included metaballs in their modelling and animation packages. Example companies include Meta Corporation (Japan), Japan LINKS Corporation, SoftImage (Quebec) which incorporated V-Clay implemented by Magic Box of Beverly Hills (California), Thomson Digital Image and Autodesk Inc. in 3D Studio, a popular modelling, rendering and animation package. Blobby molecules have been used by Pacific Data Images (California) in much of their commercial animation work and they have also been incorporated in Rayshade [17] and POV-Ray [51], freeware raytracing packages.

2.1 Summary

Modelling with implicit surfaces guarantees a continuous and smoothly blended surface which is relatively easy to deform and intuitive to specify. However, since visualisation of implicit surfaces is expensive, real-time modelling is difficult to achieve. This problem will be covered in more detail in Section 4.

Implicit surfaces are capable of topology changes without losing surface continuity. Useful in some applications, e.g. modelling amorphous material like dough or mud, it creates problems in character animation. When two primitives from the set composing a character are placed too far from each other, they will disconnect causing coherence loss (“scrambling” [44]), an often undesired effect. The opposite to coherence loss is the problem of unwanted blending, which occurs when two unblendable parts of an object (e.g. hands or legs) are placed too close to each other and blended. The coherence loss and unwanted blending occur most frequently during animation. We will discuss them in more detail in Section 5.

3. Modelling

There is a number of approaches to modelling using implicit surfaces. They will be categorised in this section into two groups: *simple primitives* and *skeletal primitives* (*skeletons*). The first category, simple primitives, groups all techniques which create models from primitives defined around a single point (e.g. a sphere or an ellipsoid). The second category, skeletal primitives, contains approaches which use primitives defined around rigid skeletons, e.g. lines or polygons. A point can also be considered a skeleton, hence the second category is a generalisation of the first one. An additional category, *complex primitives*, have been created to describe one more system which offers unconventional ways of defining implicit surfaces.

3.1 Simple primitives

Blobby Molecules. Blinn uses the following density function around a single atom (based on the behaviour of hydrogen atoms):

$$D(P) = be^{-ar^2} \tag{2}$$

where r is the distance from the point P to the centre of the atom.

For multiple atoms the overall density at a given point is calculated using summation:

$$D(P) = \sum_i b_i e^{-a_i r_i^2} = T \tag{3}$$

where r_i is the distance from the point P to the centre of the i-th atom. The expotential term is a Gaussian bump centred at r_i, with height a_i and standard deviation b_i. The "blobbiness" of a model can be controlled by adjusting the parameters a_i and b_i. The implicit surface is defined as all points where the density is equal to some threshold value T.

Metaballs. The density distribution function used in the metaball technology is as follows:

$$w(P) = \begin{cases} d_i\left(1 - 3\left(\frac{r_i}{b_i}\right)^2\right) & 0 \le r_i \le \frac{b_i}{3} \\ \frac{3d_i}{2}\left(1 - \frac{r_i}{b_i}\right)^2 & \frac{b_i}{3} \le r_i \le b_i \\ 0 & b_i \le r_i \end{cases} \tag{4}$$

where r_i is the distance from P to the centre of i-th metaball, d_i is the weight of i-th metaball and b_i is the radius of i-th metaball. For multiple primitives contributions from all influencing points are added:

$$w(P) = \sum_{m_i \in M} w_i(P) \ge C \tag{5}$$

where m_i is i-th metaball, M is a fusion cluster and C is an arbitrary constant. Points with the density value greater than or equal to C form a closed region in space, called a *fusion cluster.*

Soft Objects. In this approach Blinn's expotential distribution of density is approximated by polynomials. Additionally, the scalar field value is truncated to zero at a certain distance (radius of influence):

$$C(r) = \begin{cases} -\frac{4}{9}\frac{r^6}{R^6} + \frac{17}{9}\frac{r^4}{R^4} - \frac{22}{9}\frac{r^2}{R^2} & r \le R \\ 0 & r \ge R \end{cases} \tag{6}$$

where r is the distance from P to the considered keypoint and R is the radius of influence of the keypoint.

Two primitives are blended together by summing their contributions in the areas influenced by both of them:

$$C(P) = \sum_{i=1}^{n} C(r_i) = magic \tag{7}$$

where r_i is the distance from P to the i-th keypoint and *magic* is a constant for extracting an iso-surface. All points with the scalar field value equal to *magic* create the resulting implicit surface.

The description of blobby molecules, metaballs and soft objects presented above only shows the basic principles of these methods. They are very similar, nevertheless we can point out a few important differences between the techniques.

The density function chosen by Blinn influences all points of space. Hence, to evaluate the density value at a certain point all atoms have to be considered. This is computationaly very expensive when a large number of atoms has to be modelled. To solve this problem Blinn encloses each atom in a sphere neglecting the influence of an atom outside the corresponding sphere. For metaballs and soft objects this problem is avoided by choosing scalar field functions which drop to zero at a certain distance (Eq. 4) and (Eq. 6). Hence, points influence only finite region around them. Local surface changes are possible by adjusting characteristics of a single primitive without affecting the entire space.

Also, for metaballs and soft objects polynomials are used in the scalar field functions. Polynomials are computationaly cheaper than Blinn's expotential function. Soft objects have a further advantage over metaballs because their scalar field function only uses the value of the radius squared of a primitive. Therefore, an expensive operation of square root is avoided.

All three techniques have been extended to include more complex primitives. For instance, negative primitives were added. They interact with positive objects creating dents in them. General quadrics (ellipsoids, cylinders, planes) and superellipsoids as primitives were suggested by Blinn [2]. Non-spherical metaballs were added to the metaballs technology [19]. Finally, superellipsoidal primitives were introduced to the soft object technique in [43].

3.2 Skeletal primitives

Skeletons used for defining implicit surfaces consist of lines, curves, polygons or points. The shape of a primitive follows the shape of its skeleton. The previous section described implicit surfaces created solely from simple primitives. One disadvantage of this approach is that flat surfaces can only be approximated. The use of skeletons is one possible solution to this problem.

To create a skeletal primitive we need to find a way of calculating the scalar field value (the potential) for a given point in space. There exist two approaches to do this: *distance surfaces*, where the potential is calculated from the distance to the nearest point on the skeleton and *convolution surfaces*, where it is found from all points on the skeleton by integration.

Distance Surfaces: Their use as a way of generating an implicit surface around a skeleton are presented in [5]. The following method for calculating the potential at a point is used:

$$f(S, \boldsymbol{p}) = max_{s \in S} exp\left(\frac{-\|\boldsymbol{s} - \boldsymbol{p}\|^2}{2}\right) \tag{8}$$

where S is the skeleton, $\boldsymbol{p}$ is the point for which the potential is calculated and $\boldsymbol{s}$ is a point on the skeleton.

One disadvantage of distance surfaces is that bulges and creases may occur in the area where the skeletons meet. This effect is often undesirable. A solution to this problem was proposed by Wyvill [45]. He applied a different way of blending distance surfaces using the field function for soft objects (Eq. 6). He calculates the scalar field value as follows:

$$F_{total}(P) = \sum_{i=1}^{n} c_i F_i(r_i) \tag{9}$$

where c_i is the weight (positive or negative), F_i is the blending function (Eq. 6) and r_i is the distance from P to the nearest point on the i-th element.

Convolution Surfaces: These also solve the problem of unwanted bulges and creases [5]. A point in space has the corresponding potential value calculated from all points of the skeleton by integration:

$$f(S, \boldsymbol{p}) = \int_S \left(exp\left(\frac{-\|\boldsymbol{s} - \boldsymbol{p}\|^2}{2} \right) \right) d\boldsymbol{s} \tag{10}$$

where S is the skeleton, $\boldsymbol{p}$ is the point for which the potential is calculated and $\boldsymbol{s}$ is a point on the skeleton.

More complex convolutions can be achieved by adding weights along a skeleton, e.g. a tapered shape will be created by decreasing the weight along a line segment. Models as complex as complex a human hand can be created using convolution surfaces with triangles and line segments as skeletons [6].

3.3 Complex primitives

The last approach to modelling using implicit surfaces is taken from [21]. Three types of primitives are used: algebraic, procedural and empirical. Algebraic primitives are these which can be defined by an algebraic expression, e.g. a sphere, a quadratic surface, a torus or a sine wave. Procedural primitives can include conditions or loops in their definition, e.g. a plane defined as a sheet between two half-spaces (such a plane has thickness), a cube defined using six half-spaces or a noise which defines semi-random displacement vectors in the entire 3D space. Experimental primitives are data gathered using physical devices, e.g. medical scans.

All these primitives can be blended together preserving their C^1-continuity. First, the most suitable blending function is chosen, e.g. addition, subtraction, minimum (intersection), maximum (union). A smoothing function can be applied afterwards if required to smooth sharp edges.

4. Rendering

Implicit surfaces suffer from a serious drawback - they cannot be easily rendered at interactive speeds. Two techniques that have been most widely used for rendering them are polygonisation and direct ray-tracing. Since they cannot be so far be performed in real-time, the usual trade-off between high quality and speed occurs. Hence, alternative techniques which display the approximation of the surface have been developed. In this section we will first describe the approximation techniques, then discuss polygonisation and direct ray-tracing and finish the section by showing how to achieve further realism during rendering by texturing implicit surfaces.

4.1 Surface approximation

Bloomenthal and Wyvill [4] suggest several techniques which offer real-time or near real-time display of implicit surfaces: octree display, 2d slices, image thresholding or scattering. The idea of scattering is to display a cloud of seeds resting on an implicit surface. When an implicit surface moves, the seeds follow it until they reach the surface again. In [4] initial seeds are computed from cross-sections of a skeleton. They then migrate towards the skeleton in the gradient direction. It is important to maintain uniform seed distribution at each time step.

Scattering has been further developed by Witkin and Heckbert [39] and Desbrun *et al.* [8]. Witkin and Heckbert [39] use a set of particles which attract and repel each other to achieve an even sampling of an implicit surface. Fission and death of particles occur when particle density in an area is too low or too high respectively.

In [8], each primitive sends a set of particles in fixed directions. Validation tests are performed to determine which of the sent seeds are to be used for sampling of the implicit surface. The resulting set of valid seeds is used for real-time visualisation and for collision detection between implicit solids.

4.2 Polygonisation

To polygonise an implicit surface we need to divide a 3D space into grid cells. Then the polygonisation process consists of two stages: first, find all cells which are intersected by the surface (*boundary cells*) and, second, produce polygons in each boundary cell according to the scalar field values in the corners of the cell.

There are many algorithms for polygonisation of implicitly defined shapes [18, 32, 37, 38, 36]. They are mostly used for scientific visualisation. Wyvill et al. [40] propose a polygonisation algorithm specifically designed for the implicit surfaces modelling technique. The boundary cells are found by examining all grid cells starting at a keypoint and going up, down, left or right in the grid. When a boundary cell is encountered, its neighbouring cells are tested if they are not boundary as well. This is recursively repeated until all boundary cells are found. The polygonisation of a cubical cell is decided on a basis of hot/cold property of each corner of the cell (where hot means inside, cold outside the surface). Because of ambiguous polygonisation cases, the rendered surface may contain ‘cracks’ if two neighbour cells were polygonised inconsistently. Also, fine details can be missed if the cell size is too large.

Bloomenthal [3] suggests a faster polygonisation algorithm based on the octree and adaptive octree subdivision. He uses tetrahedronal cells avoiding the ambiguous polygonisation problem. Adaptive subdivision also ensures that fine details of the surface are found.

Wyvill and Jevans [48] propose an algorithm based on look-up tables. Tables store 256 cases of the possible hot/cold characteristics of the corners of an individual cell. Polygonisation of cells is therefore fast and automatic. However, this algorithm can miss some surface details because it does not handle adaptive subdivision.

Ning and Bloomenthal [24] provide a comprehensive review of implicit surface polygonisers.

4.3 Direct ray tracing

Ray tracing can be used for visualisation of any type of model as long as ray-surface intersection points and normal vectors to the surface can be calculated. For implicit surfaces the initial guesses for the intersection points are estimated by intersecting a ray with a sphere which bounds a primitive. More detailed numerical calculations have to be performed in order to obtain the precise intersection point.

The normal vector at a given point is calculated as partial derivatives of the scalar field function:

$$N = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}\right) \tag{11}$$

Blinn [2] and Nishimura et al. [25] visualise their models using ray tracing. More general ray tracing algorithms for implicit surfaces have also been developed. Kalra and Barr [16] propose an algorithm which guarantees ray-surface intersections even with the finest details on an implicit surface. Wyvill and Trotman [49] suggest an algorithm which is less general but finds all intersections along a ray, not only the closest one, a feature useful for CSG operations on implicit surfaces. Hart [15] presents a good review of ray tracing applied to the implicit surfaces modelling technique. More recently, Gascuel [12] developed a robust algorithm for ray-tracing implicit surfaces built using skeletal primitives.

4.4 Texturing Implicit Surfaces

Texturing general models built with implicit surfaces is considered an open research problem. The possibility of constant changes of the surface of a model (including topology changes) is one of the main advantages of the implicit surfaces modelling technique. However, this causes problems with texturing such models. This section will present three existing techniques for texturing implicit surfaces: the use of colour, two-dimensional texture mapping and three-dimensional texture mapping.

Colour. Colour is the simplest "texture" which can be applied to an implicit surface. It has been used in all implementations of implicit surfaces. The most common approach is to apply the weighted sum of colour contributions from all primitives to find the colour of the areas of multiple influence [2, 25, 41, 21, 23]. This results in a smooth colour transition in the blending area.

McPheeters [21] summarises possible colouring metaphors for implicit surfaces. He suggests three simple approaches: all primitives change colour to the blended colour (primitives are like drops of coloured water), the division between primitives is visible (primitives are made of clay-like solids) or there is a smooth transfer of colour (semi-solid primitives which exhibit limited mixing). He also mentions a more complex metaphor: the effect of sculpting an object from differently coloured pieces of clay, the result being streaks of two initial colours and shades of the blended colour around the connection area.

2D texture mapping (parametrisation). Two-dimensional (2D) texture mapping finds a texture value for a surface point from the parametric representation of the surface. The values of parameters are passed to a 2D function which returns the required texture value. Implicit surfaces are not easily parametrised. Nevertheless parametrisation can be obtained in some cases. Bloomenthal [3] parametrises implicit surfaces using their boundary representation (polygons). McPheeters [21] uses direct parametrisation of primitives applying complex mathematical procedures. Smets [31] observes that parametrisation is not essential for applying 2D texture mapping. What is sufficient is a homeomorphism between instances of a moving and deforming implicit surface. He proposes an algorithm (in the 2D case) for finding such a homeomorphism. The resulting texture 'sticks' to the surface.

3D texture mapping (solid texturing). Solid texturing uses the three-dimensional (3D) co-ordinates of a surface point to determine the texture value. The texture value is found in a texture space which has a 3D pattern defined in it. Problems arise when a textured object moves. If we make the texture space static the texture on the object will change over time. This is usually undesirable. A simple solution is to move the texture space with the object. However, for implicit surfaces the movement may involve changes in the topology of the object. Moving the texture space is therefore insufficient (in fact, sometimes it is impossible to perform, e.g. when two primitives are dragged apart, which way would the texture space go?). Wyvill et al. [42] give a solution to this problem by attaching a separate texture space to each keypoint and then interpolating the final texture using local texture values for keypoints. The texture value at a surface point is a weighted sum of all texture contributions from the influencing primitives. The texture is distorted when two primitives blend with each other but it gives the impression of the correct appearance.

5. Animation

Implicit surfaces offer a compact representation of a model (a set of primitives, their scalar field functions, blending method and an iso-value). To create an animation of such a model, we need to know the way this specification changes over time. For each time-step, this specification will be used to extract the iso-surface at this time. Various animation techniques can be used to specify the change of an implicit surface over time. In this section we present three approaches: keyframing (including metamorphosis), physically based simulation and a hybrid method used for character animation which combines keyframing and automatic motion generation.

5.1 Keyframe animation

For keyframe animation, a set of keyframes of a model is given for certain time instants together with a time curve which describes the rate of frames generated between keyframes. The frames in between the key positions (inbetweens) are generated automatically. Keyframing requires considerable effort and artistic ability to create a successful animation.

For implicit surfaces, each keyframe is given as a specification of the current configuration of a model (a set of primitives, their scalar field functions, blending method and an iso-value). Inbetweens are generated and the result is a C^1 continuous surface although surface topology may change. Successful animation sequences have been created using this method [1, 41].

Metamorphosis can also be performed using keyframes. Two methods for performing metamorphosis described in [44] are *surface matching* or *cellular matching*. Surface inbetweening starts from both source and target objects and weights their influence to the final image throughout the metamorphosis process. At the beginning the source has the weight 1 and the target 0. In the end the source has the weight 0 and the target 1. This method often causes "scrambling" of an object since new primitives appear suddenly. Surface inbetweening is also used by [21].

Cellular matching matches parts of the source and the target according to the space they occupy. Essentially primitives from the same cells in a 3D grid in both the source and the target are matched. Such matching may result in the need for some primitives to grow or fade away. Yet the coherence of the model is better preserved than in the surface inbetweening method.

5.2 Physically based simulation

Primitives can be treated as rigid objects and Newtonian equations of motion can be applied to simulate their motion [11, 50]. Gascuel [11] uses the implicit surface generated by primitives in a physically based environment for fast collision detection using the inside/outside test of one object on the sampling points of another. During collisions, field functions are locally deformed by introducing deformation terms in the interpenetration area and to propagate the deformation outside the contact area. Both linear and non-linear stiffness of solids can be simulated.

Wyvill [50] achieves deformation during collisions by space warping. Various regions of space have deformation terms associated with them. When an object enters a particular region, its field function changes, i.e. it undergoes deformation. For instance, a ball squashes when it enters the region of space closer to the ground and stretches when it bounces back up.

Physical simulation in a particle system can also be used to animate implicit surfaces. It gives particularly good results for modelling inanimate materials, e.g. mud, clay or dough. In [7] a highly deformable material is simulated in a generalised particle system with Lennard-Jones attraction/repulsion interaction forces between primitives. Such material can undergo fractures and its chunks can join back together. Collision detection and response between unblendable materials can be modelled and the progressive fusion between chunks occurs under influence of sufficient force. By using a very soft blending function, the simulation requires fewer particles than in [22, 34, 35] where implicit surfaces are only used to visualise results obtained from a particle system. An important issue addressed in [9] is volume conservation of implicit material modelled this way. During animation, it can undergo considerable (up to 50%) volume variations. By locally controlling the volume in [9], this is decreased to under 3%, hence ensuring realistic simulation.

5.3 Hybrid animation control for character animation

Keyframe generation for implicit surfaces for character animation requires good artistic skills from an animator in order to produce an interesting animation sequence. Care has to be taken not to move certain primitives too close to each other (unwanted blending) or too far from each other (coherence loss). For keyframing, the motion control is left entirely up to the animator. This process may prove very tedious, less skilled animators may even give up altogether. On the other hand however, animation control cannot be taken completely out of the animator's hands and automatised, since it would not normally produce the exact motion required by the animator. Assistance with motion generation is desired but fine tuning of detail should be left to the animator.

In [26] the ABC of implicit surfaces for character animation is proposed: automatically generated motion has to maintain Appearance, prevent unwanted Blending and preserve Coherence. These principles were used in [27] to create traditional animation effects, e.g. squash & stretch, follow through and exaggeration. A character in [27] is defined as a layered construction (an articulated figure clothed with implicit flesh). The motion of the articulated figure is keyframed and the motion of flesh is automatically generated. In [28] this approach was extended to give an animator more control over tuning the generated motion to their requirements.

5.4 Summary

During animation, the problems of unwanted blending, coherence loss and volume variation may occur. Several solutions have been suggested, below we present their summary.

Unwanted Blending: To solve this problem the metaball technology offers both fusible and non-fusible connections between clusters. In the former case clusters blend to form a new cluster while they do not affect each other in the latter. Eccentric metaballs (as opposed to concentric spheres) are introduced [19]. Two metaballs will blend (or fuse) only if their centres of density are close enough. Eccentric metaballs have their centres of field shifted which helps to avoid blending (called meta-fusion). Unwanted blending will still occur in some cases. [43] solves the unwanted blending problem for soft objects by hierarchical clothing of a scene, rendering groups of blendable primitives in separate runs of a renderer. However, the unblendable primitives can only intersect each other, they do not interact. In [11, 26] similar solutions are proposed which modify the scalar field function in order to achieve deformation between unblendable primitives. The case of blendable and unblendable primitives (e.g. two fingers blended with a palm on one end but not allowed to blend with it on the other end) is handled in [26] but the method may introduce surface discontinuities. [8, 14] further develop this algorithm to improve its performance in these cases, however satisfactory solution has not yet been proposed.

Coherence: This occurs when one primitive is moved too far apart from the others. The most popular method of preventing coherence loss is to calculate molecular Lennard-Jones attraction/repulsion forces between primitives [35, 22, 7, 27]. Primitives can also be connected by springs and dampers [34].

Volume Variation: When model undergoes considerable volume variation, the simulation is not successful. This has been controlled in [8, 9] by associating local volume with each primitive and maintaining it during animation. Volume variation has been decreased to under 3%.

6. Conclusions

This paper has presented a review of implicit surfaces. They are particularly suitable for modelling smoothly blended, “plasticine” objects which deform during motion and can undergo topology changes. Such objects change their shape due to external forces, deform during collisions with other objects or conform to their environment. Deformation of an object can be achieved by simply altering the spacial relationship between primitives. A closed, smoothly blended surface is always guaranteed, although it may consist of multiple, disconnected parts.

Implicit surfaces have been widely used in computer graphics and animation in both commercial and academic worlds. The commercial world adapted the simple version of the technique based on field functions around point sources. In the academic world, these have been extended to more general skeletons (points, lines, curves, polygons or solids).

The main problems with the technique are unwanted blending, coherence loss, volume variation and lack of real-time rendering. However, solutions to these problems have been proposed and implicit surfaces are growing in popularity since they offer an attractive alternative to parametric surfaces.

7. References

1. Beier, T. “Practical Uses for Implicit Surfaces in Animation”, Siggraph 90 Course Notes No 23, “Modelling and Animating With Implicit Surfaces”, pp 20.1-20.11
2. Blinn, J. F., “A Generalisation of Algebraic Surface Drawing”, ACM Trans. Graphics, Vol. 1, No 3, July 1982, pp 135-256
3. Bloomenthal, J., “Polygonization of Implicit Surfaces”, Computer Aided Geometric Design, Vol5, No 4, November 1988, pp 341-355
4. Bloomenthal, J. and Wyvill, B., “Interactive Techniques for Implicit Modelling”, Computer Graphics, Vol. 24, No 2, 1990, pp 109-116
5. Bloomenthal, J. and Shoemake, K., “Convolution Surfaces”, Computer Graphics, Vol. 25, No 4, July 1991, pp 251-256
6. Bloomenthal, J., “Hand crafted”, Proc. of the 4th Annual Western Computer Graphics Symposium, Banff, Alberta, April 1992; also in Siggraph 93 Course Notes No 25, “Modelling, Visualizing and Animating Implicit Surfaces”
7. Desbrun, M. and Gascuel, M-P.,”Highly Deformable Material for Animation and Collision Processing”, Proc. 5th Eurographics Workshop on Animation and Simulation, Oslo, September 94.
8. Desbrun, M., N. Tsingos and Gascuel, M-P. “Adaptive Sampling of Implicit Surfaces for Interactive Modelling and Animation”, Proc. 1st International Eurographics Workshop on Implicit Surfaces (Grenoble, 17-18th April, 1995)
9. Desbrun, M. and Gascuel, M-P.,”Animating Soft Substances with Implicit Surfaces”, Computer Graphics Proc. Siggraph’95, 1995.
10. Fujita, T., Hirota, K. and Murakami, K., “Representation of Splashing Water using Metaball Model”, Fujitsu, 1990, Vol. 41, part 2, pp 159-165, in Japanese
11. Gascuel, M.-P., “An Implicit Formulation for Precise Contact Modelling between Flexible Solids”, Proceedings of SIGGRAPH 93 (Anaheim, California, August 1-6, 1993). In Computer Graphics Proceedings, Annual Conference Series, 1993, ACM SIGGRAPH, New York, pp 313-320
12. Gascuel, J.-D., “Implicit Patches: An optimised and powerful ray intersection algorithm”, Proc. 1st International Eurographics Workshop on Implicit Surfaces (Grenoble, 17-18th April, 1995)
13. Graves, G., “The Magic of metaballs”, Computer Graphics World, May 1993, pp 27-32
14. Guy, A. and B. Wyvill, “Controlled Blending for Implicit Surfaces”, Proc. 1st International Eurographics Workshop on Implicit Surfaces (Grenoble, 17-18th April, 1995)
15. Hart, J., “Ray Tracing Implicit Surfaces”, Siggraph 93 Course Notes No 25, “Modelling, Visualizing and Animating Implicit Surfaces”
16. Karla, D. and Barr, A., “Guaranteed Ray Intersections with Implicit Surfaces”, Computer Graphics, Vol. 23, No 3, 1989, pp 297-306
17. Kolb, C. and R. Bogart, Rayshade 4.0. Available by ftp from cs.princeton.edu, 1991.
18. Lorensen W. and H. Cline, “Marching cubes: a high resolution 3D surface construction algorithm”, Computer Graphics, Vol. 21, No. 4, July 1987, pp 163-169.
19. MetaEditor, “Operation guide”, personal communication from Bruno Tsuchiya, META Corporation, Japan, July 1993
20. Max, N. L. and Wyvill, B., “Shapes and Textures for Rendering Coral”, Scientific Visualisation of Physical Phenomena, N. M. Patrikalakis (Ed.), Springer-Verlag 1991, pp 333-343
21. McPheeters, C. W., “Isosurface Modelling of Soft Objects in Computer Graphics”, 1990, National Centre for Computer Animation, Department of Communication and Media, Dorset Institute, PhD thesis
22. Miller, G. and Pearce, A., “Globular Dynamics: A Connected Particle System for Animating Viscous Fluids”, Vol. 13, No 3, 1989, pp 305-309

23. Muraki, S., “Volumetric Shape Description of Range Data Using “Blobby Model”Model””, Computer Graphics, Vol. 25, No 4, July 1991, pp 227-235

24. Ning, P. and Bloomenthal, J, “Evaluation of Implicit Surface Tilers”, Siggraph 93 Course Notes No 25, “Modelling, Visualizing and Animating Implicit Surfaces”

25. Nishimura, H., Hirai, M., Kawai, T., Kawata, T., Shirakawa, I. and Omura, K., “Object Modelling by Distribution Function and a Method of Image Generation”, The Transactions of the Institute of Electronics and Communication Engineers of Japan, 1985, Vol. J68-D, Part 4, pp. 718-725, in Japanese, translated into English by Takao Fujuwara

26. Opalach, A. and S. Maddock, “Implicit Surfaces: Appearance, Blending and Consistency”, proc. of 4th Eurographics Workshop on Animation and Simulation, Barcelona, September 1993, pp 233-245

27. Opalach, A. and Maddock, S., “Disney Effects Using Implicit Surfaces”, Proc. 5th Eurographics Workshop on Animation and Simulation (Oslo, 17-18th September, 1994)

28. Opalach, A. and Maddock, S., “High Level Control of Implicit Surfaces for Character Animation”, Proc. 1st International Eurographics Workshop on Implicit Surfaces (Grenoble, 17-18th April, 1995)

29. Payne, B. A. and Toga, A. W., “Distance Field Manipulation of Surface Models”, IEEE Computer Graphics and Applications, January 1992, pp 65-71

30. Reed, T. and B. Wyvill, “Visual Simulation of Lightning”, Computer Graphics, Proceedings Siggraph’94, 1994, pp 359-364.

31. Smets, J-P, “Surfacic Textures for Animated Implicit Surfaces: the 2D case”, proc. of 4th Eurographics Workshop on Animation and Simulation, Barcelona, September 1993, pp 221-232

32. Suffern, K., “Recursive Space Subdivision Techniques for Rendering Implicit Surfaces”, Proc. of The Australian Conference on Computer Graphics 1989, pp 239-250

33. Tatsumi, H., Takaoki, E., Omura, K. and Fujito, H., “A new method for 3D reconstruction from serial sections by computer graphics using “meta-balls”: reconstruction of “Hepatoskeletal System” formed by Ito Cells in the Cod Liver”, Computers and Biomedical Research, 1990, Vol. 23, Part 1, pp 37-45

34. Terzopoulos, D., Platt, J. and Fleisher, K. “Heating and melting deformable models (from goop to glop)”, Graphics Interface'89, June 1989, pp 219-226

35. Tonnesen, D., “Modelling liquids and solids using thermal particles”, Proceedings of the Graphics Interface 1991, pp. 255-262

36. Turk, G., “Re-Tiling Polygonal Surfaces”, Computer Graphics, Vol26, No 2, July 1992, pp 55-64

37. Velho, L., “Adaptive polygonization of implicit surfaces using simplicial decomposition and boundary constraints”, Eurographics 90, C.E. Vandoni and D.A. Duce (Eds)

38. Wilhelms, J. and van Gelder, A., “Topological Considerations in Isosurface Generation Extended Abstract”, Computer Graphics, Vol. 24, No 5, November 1990, pp 79-86

39. Witkin, A. and P. Heckbert, “Using Particles to Sample and Control Implicit Surfaces”, Computer Graphics, Proceedings Siggraph’94, July 1994, pp. 269-278.

40. Wyvill, G. and McPheeters, C. and Wyvill, B., “Data Structure for Soft Objects”, The Visual Computer, Vol. 2, No 4, August 1986a, pp 227-234

41. Wyvill, G. and McPheeters, C. and Wyvill, B., “Animating Soft Objects”, The Visual Computer, Vol. 2, No 4, August 1986b, pp 235-242

42. Wyvill, G. and McPheeters, C. and Wyvill, B., “Solid Texturing of Soft Objects”, IEEE Computer Graphics and Applications, Vol7, No 12, December 1987, pp 20-26

43. Wyvill, B. and Wyvill, G., “Field Functions for Implicit Surfaces”, Visual Computer, Vol. 5, 1989, pp 75-82

44. Wyvill, B., “Metamorphosis of Implicit Surfaces”, Siggraph 90 Course Notes 23, “Modelling and Animating with Implicit Surfaces”

45. Wyvill, B., “Implicit surface Tiling Techniques”, Siggraph 93 talk, Course No 25, “Modelling, Visualizing and Animating Implicit Surfaces”

46. Wyvill, B., “Soft”, Siggraph 86 Electronic Theatre and Video Review, Issue 24, 1986.

47. Wyvill, B., “The Great Train Rubbery”, Siggraph 88 Electronic Theatre and Video Review, Issue 26, 1988.

48. Wyvill, B. and Jevans, D., “Table Driven Polygonization”, Siggraph 93 Course Notes No 25, “Modelling, Visualizing and Animating Implicit Surfaces”

49. Wyvill, G. and Trotman, A., “Ray-Tracing Soft Objects”, proc. CG International, 1990, pp 469-475

50. Wyvill, B., “Warping Implicit Surfaces”, Proc. Western Computer Graphics Symposium (SKIGRAPH 92), 1992, pp 55-63

51. Young, P. (POV-Team Coordinator) et al., POV-Ray v2.2. Available by ftp from alfred.ccs.carleton.ca.